Robert Schaefer
Carlos Cotta
Joanna Kołodziej
Günter Rudolph (Eds.)

LNCS 6238

Parallel Problem Solving from Nature – PPSN XI

11th International Conference
Kraków, Poland, September 2010
Proceedings, Part I

Part I

Springer

Lecture Notes in Computer Science 6238

Robert Schaefer   Carlos Cotta
Joanna Kołodziej   Günter Rudolph (Eds.)

Parallel Problem Solving from Nature – PPSN XI

11th International Conference
Kraków, Poland, September 11-15, 2010
Proceedings, Part I

Springer

Volume Editors

Robert Schaefer
AGH University of Science and Technology
Faculty of Electrical Engineering, Automatics, Computer Science
and Electronics, Department of Computer Science
Mickiewicza 30, 30-059 Kraków, Poland
E-mail: schaefer@agh.edu.pl

Carlos Cotta
Universidad de Málaga
Departamento Lenguajes y Ciencias de la Computación
ETSI Informática, Campus Teatinos, 29071 Málaga, Spain
E-mail: ccottap@lcc.uma.es

Joanna Kołodziej
University of Bielsko-Biala
Department of Mathematics and Computer Science
Willowa 2, 43-309 Bielsko-Biala, Poland
E-mail: jkolodziej@ath.bielsko.pl

Günter Rudolph
Technische Universität Dortmund, Fakultät für Informatik
44221 Dortmund, Germany
E-mail: guenter.rudolph@tu-dortmund.de

Library of Congress Control Number: 2010934114

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

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

ISSN 0302-9743
ISBN-10 3-642-15843-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-15843-8 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

Preface

We are very pleased to present to you this LNCS volume, the proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN 2010). PPSN is one of the most respected and highly regarded conference series in evolutionary computation, and indeed in natural computation as well. This biennial event was first held in Dortmund in 1990, and then in Brussels (1992), Jerusalem (1994), Berlin (1996), Amsterdam (1998), Paris (2000), Granada (2002), Birmingham (2004), Reykjavik (2006) and again in Dortmund in 2008.

PPSN 2010 received 232 submissions. After an extensive peer review process involving more than 180 reviewers, the program committee chairs went through all the review reports and ranked the papers according to the reviewers' comments. Each paper was evaluated by at least three reviewers. Additional reviewers from the appropriate branches of science were invoked to review into disciplinary papers. The top 128 papers were finally selected for inclusion in the proceedings and presentation at the conference. This represents an acceptance rate of 55%, which guarantees that PPSN will continue to be one of the conferences of choice for bio-inspired computing and metaheuristics researchers all over the world who value the quality over the size of a conference.

The papers included in the proceedings volumes cover a wide range of topics, from evolutionary computation to swarm intelligence, from bio-inspired computing to real-world applications. Machine learning and mathematical games supported by evolutionary algorithms as well as memetic, agent-oriented systems are also represented. They all are the latest and best in natural computation. The proceedings are composed of two volumes divided into nine thematic sections.

In accordance with the PPSN tradition, all papers at PPSN 2010 were presented as posters. There were nine sessions of posters. Each session consisted of around 15 papers. For each session, we covered as wide a range of topics as possible so that participants with different interests could find some relevant papers at every session.

PPSN 2010 featured three distinguished keynote speakers: John Garibaldi, Zbigniew Michalewicz and Darrell Whitley who delivered lectures entitled: Ensemble Fuzzy Reasoning, Some Thoughts on Wine Production, and Elementary Landscapes Made Easy, respectively.

PPSN 2010 also included eight interesting tutorials. These covered the wide area of natural computing science. The first of them "A Rigorous Theoretical Framework for Measuring Generalization of Co-evolutionary Learning" (X. Yao) was devoted to the genetic algorithm theory while the following two "Foundations of Evolutionary Multi-objective Optimization" (F. Neumann, T. Friedrich) and "Hybrid Optimization Approaches" (G. Raidl) introduced important groups of algorithms inspired by nature. The next tutorials, "Natural Computing and

Finance" (T. Brabazon, M. O'Neill), "Heuristic and Meta-heuristic Approaches for Scheduling in Large Scale Distributed Computing Environments" (F. Xhafa) and "Artificial Immune Systems in Optimization and Classification Problems with Engineering and Biomedical Applications" (T. Burczyński, M. Bereta, W. Kuś), focused on important engineering, business and medical applications. Finally, "Learning to Play Games" (S. M. Lucas) and "The Complexity of Elections: New Domain for Heuristic Computations" (P. Faliszewski) concerned games and social problems.

PPSN 2010 also included four workshops. They made an excellent start to the five-day event. The workshops offered an ideal opportunity for participants to explore specific topics in natural computation in an informal setting. They sowed the seeds for the future growth of natural computation. The first of them "Self-tuning, Self-configuring and Self-generating Search Heuristics (Self* 2010)" (G. Ochoa, M. Schoenauer, D. Whitley) focused on developing automated systems to replace the role of a human expert in the design, tuning and generation of search heuristics. The next pair of workshops "Understanding Heuristics: How Do We Get the Best of Both Theory and Empirical Methods?" (E. Ozcan, A. Parkes, J. Rowe) and "Experimental Methods for the Assessment of Computational Systems (WEMACS)" (T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss) concerned two complementary theoretical and experimental approaches to the analysis of heuristic and meta-heuristic algorithms. The last one "Workshop on Parallel and Cooperative Search Methods" (D. Ouelhadj, E. Ozcan, M. Toulouse) dealt with cooperative parallel searches improving performance, especially when dealing with large scale combinatorial optimization problems.

The success of any conference depends on its authors, reviewers and organizers. PPSN 2010 was no exception. We are grateful to all the authors who submitted their papers and to all the reviewers for their outstanding work in refereeing the papers on a very tight schedule. We relied heavily on a team of volunteers, especially those in Kraków, to keep the PPSN 2010 wheel turning.

PPSN XI would not have been possible without the support of Microsoft Poland, Intel and HP.

September 2010

Robert Schaefer
Carlos Cotta
Joanna Kołodziej
Günter Rudolph
Juan J. Merelo
Hans-Paul Schwefel

Organization

PPSN XI was organized and hosted by the Intelligent Information Systems Group of the Department of Computer Science, Faculty of Electrical Engineering, Automatics, Computer Science and Electronics, AGH University of Science and Technology, Poland. The conference took place in the AGH Conference and Teaching Center U-2, Kraków.

Steering Committee

David W. Corne	Heriot-Watt University Edinburgh, UK
Kenneth De Jong	George Mason University, USA
Agoston E. Eiben	Vrije Universiteit Amsterdam, Netherlands
Juan Julián Merelo Guervós	Universidad de Granada, Spain
Günter Rudolph	Technische Universität Dortmund, Germany
Thomas P. Runarsson	University of Iceland, Iceland
Marc Schoenauer	Université Paris Sud, France
Xin Yao	University of Birmingham, UK

Conference Committee

General Chair	Robert Schaefer (AGH Kraków, Poland)
Honorary Chair	Hans-Paul Schwefel (Technische Universität Dortmund, Germany)
Program Chairs	Carlos Cotta (University of Málaga, Spain)
	Joanna Kołodziej (ATH Bielsko-Biała, Poland)
	Günter Rudolph (Technische Universität Dortmund, Germany)
Workshop Chair	Aleksander Byrski (AGH Kraków, Poland)
Tutorial Chair	Krzysztof Cetnarowicz (AGH Kraków, Poland)
E-proceedings Chair	Juan Julián Merelo Guervós (Universidad de Granada, Spain)
E-publicity Chair	Bartłomiej Śnieżyński (AGH Kraków, Poland)
Technical Editorial Chair	Jarosław Koźlak (AGH Kraków, Poland)
Financial Manager	Leszek Siwik (AGH Kraków, Poland)
Local Organization	Jacek Dajda (AGH Kraków, Poland)
	Ewa Olejarz (AGH, Kraków, Poland)
	Rafał Dreżewski (AGH Kraków, Poland)
	Piotra Gawor (ATH Bielsko-Biała, Poland)
	Dawid Kotrys (ATH Bielsko-Biała, Poland)
	Anna Prochownik (ATH Bielsko-Biała, Poland)

Workshops

Self-tuning, Self-configuring and Self-generating Search Heuristics (Self* 2010)
Gabriela Ochoa, Marc Schoenauer and Darrell Whitley

Understanding Heuristics: How Do We Get the Best of Both Theory and Empirical Methods?
Ender Özcan, Andrew Parkes and Jonathan Rowe

Experimental Methods for the Assessment of Computational Systems (WEMACS)
Thomas Bartz-Beielstein, Marco Chiarandini, Luis Paquete and Mike Preuss

Workshop on Parallel and Cooperative Search Methods
Djamila Ouelhadj, Ender Özcan and Michel Toulouse

Tutorials

A Rigorous Theoretical Framework for Measuring Generalization of Co-evolutionary Learning
Xin Yao

Foundations of Evolutionary Multi-objective Optimization
Frank Neumann and Tobias Friedrich

Hybrid Optimization Approaches
Günther Raidl

Natural Computing and Finance
Tony Brabazon and Michael O'Neill

Heuristic and Meta-heuristic Approaches for Scheduling in Large Scale Distributed Computing Environments
Fatos Xhafa

Artificial Immune Systems in Optimization and Classification Problems with Engineering and Biomedical Applications
Tadeusz Burczyński, Michał Bereta and Wacaw Kuś

The Complexity of Elections: New Domain for Heuristic Computations
Piotr Faliszewski

Learning to Play Games
Simon M. Lucas

Program Committee

Alex Agapie
Uwe Aickelin
Enrique Alba
Anna I. Esparcia Alcázar
Jhon Edgar Amaya
Jarosław Arabas
Dirk Arnold
Anne Auger
Dariusz Badura
Thomas Bartz-Beielstein
Peter Bentley
Nicola Beume
Marenglen Biba
Mark Bishop
Christian Blum
Yossi Borenstein
Urszula Boryczka
Peter Bosman
Pascal Bouvry
Anthony Brabazon
Juergen Branke
Dimo Brockhoff
Larry Bull
Tadeusz Burczyński
Juan Carlos Burguillo-Rial
Aleksander Byrski
Erick Cantú-Paz
Uday Chakraborty
Ying-ping Chen
Sung-Bae Cho
Siang-Yew Chong
Carlos Coello Coello
Pierre Collet
David W. Corne
Luis Correia
Carlos Cotta
Peter Cowling
Valentin Cristea
Kenneth DeJong
Benjamin Doerr
Marco Dorigo
Rafał Dreżewski
Stefan Droste
Gusz Eiben
Talbi El-Ghazali
Michael Emmerich
Andries Engelbrecht
Thomas English
Antonio J. Fernández
Jonathan Fieldsend
Carlos M. Fonseca
Tobias Friedrich
Marcus Gallagher
Jos Enrique Gallardo
Jonathan M. Garibaldi
Mario Giacobini
Kyriakos Giannakoglou
Jens Gottlieb
Roderich Gross
Juan Julian Merelo Guervos
Steven Gustafson
Hisashi Handa
Nikolaus Hansen
Emma Hart
Philip Hingston
Jeff Horn
Eyke Huellermeier
Christian Igel
Christian Jacob
Wilfried Jakob
Mark Jelasity
Yaochu Jin
Bob John
Bryant A. Julstrom
Iwona Karcz-Dulęba
Andy Keane
Graham Kendall
Joshua Knowles
Joanna Kołodziej
Wolfgang Konen
Jozef Korbicz
Witold Kosinski
Jarosław Koźlak
Oliver Kramer
Krzysztof Krawiec
Halina Kwaśnicka
Pier Luca Lanzi
Pedro Larranaga
Per Kristian Lehre
Jose A. Lozano
Simon Lucas
Evelyne Lutton
Krzysztof Malinowski
Jacek Mańdziuk
Elena Marchiori
Barry McCollum
Jorn Mehnen
Peter Merz
Silja Meyer-Nieberg
Zbigniew Michalewicz
Ralf Mikut
Alberto Moraglio
Boris Naujoks
Ferrante Neri
Frank Neumann
Ewa Niewiadomska-Szynkiewicz
Lars Nolle
Shigeru Obayashi
Andrzej Obuchowicz
Pietro Oliveto
Ben Paechter
Luis Paquete
Mario Pavone
Pawel Pawlewski
Martin Pelikan
Florin Pop
Mike Preuss
Christian Prins
Domenico Quagliarella
Günther Raidl
Robert Reynolds
Philipp Rohlfshagen
Andrea Roli
Jonathan Rowe
Günter Rudolph

Thomas Runarsson	Moshe Sipper	Heike Trautmann
Thomas A. Runkler	Leszek Siwik	Krzysztof Trojanowski
Leszek Rutkowski	Jerzy Stefanowski	Andrew Tuson
Conor Ryan	Thomas Stibor	Massimiliano Vasile
Michael Sampels	Catalin Stoean	Igor Vatolkin
Ruhul Sarker	Thomas Stützle	Pedro Isasi Viñuela
Jayshree Sarma	Dirk Sudholt	R. Paul Wiegand
Robert Schaefer	Ponnuthurai Suganthan	Slawomir Wierzchoń
Robert Scheffermann	Ryszard Tadeusiewicz	Carsten Witt
Marc Schoenauer	Kay Chen Tan	Janusz Wojtusiak
Oliver Schütze	Alexander Tarakanov	Fatos Xhafa
Bernhard Sendhoff	Olivier Teytaud	Xin Yao
Franciszek Seredyński	Lothar Thiele	Tina Yu
Marc Sevaux	Dirk Thierens	Ivan Zelinka
Jonathan Shapiro	Jon Timmis	Qingfu Zhang
Ofer Shir	Julian Togelius	

Sponsoring Institutions

Hewlett-Packard Polska
Intel Corporation
Microsoft

Table of Contents – Part I

Theory of Evolutionary Computing (I)

Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem 1
Süntje Böttcher, Benjamin Doerr, and Frank Neumann

Mirrored Sampling and Sequential Selection for Evolution Strategies 11
Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V. Arnold, and Tim Hohm

Optimisation and Generalisation: Footprints in Instance Space 22
David W. Corne and Alan P. Reynolds

Adaptive Drift Analysis 32
Benjamin Doerr and Leslie Ann Goldberg

Optimizing Monotone Functions Can Be Difficult 42
Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges

Log-Linear Convergence of the Scale-Invariant $(\mu/\mu_w, \lambda)$-ES and Optimal μ for Intermediate Recombination for Large Population Sizes 52
Mohamed Jebalia and Anne Auger

Exploiting Overlap When Searching for Robust Optima 63
Johannes Kruisselbrink, Michael Emmerich, André Deutz, and Thomas Bäck

Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis 73
Olaf Mersmann, Mike Preuss, and Heike Trautmann

One-Point Geometric Crossover 83
Alberto Moraglio

When Does Dependency Modelling Help? Using a Randomized Landscape Generator to Compare Algorithms in Terms of Problem Structure 94
Rachael Morgan and Marcus Gallagher

First-Improvement vs. Best-Improvement Local Optima Networks of NK Landscapes 104
Gabriela Ochoa, Sébastien Verel, and Marco Tomassini

Differential Mutation Based on Population Covariance Matrix 114
Karol Opara and Jarosław Arabas

General Lower Bounds for the Running Time of Evolutionary Algorithms 124
Dirk Sudholt

A Binary Encoding Supporting Both Mutation and Recombination 134
Karsten Weicker

Towards Analyzing Recombination Operators in Evolutionary Search . . . 144
Yang Yu, Chao Qian, and Zhi-Hua Zhou

Theory of Evolutionary Computing (II)

Bidirectional Relation between CMA Evolution Strategies and Natural Evolution Strategies 154
Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi

A Fine-Grained View of GP Locality with Binary Decision Diagrams as Ant Phenotypes 164
James McDermott, Edgar Galván-Lopéz, and Michael O'Neill

Drift Analysis with Tail Bounds 174
Benjamin Doerr and Leslie Ann Goldberg

More Effective Crossover Operators for the All-Pairs Shortest Path Problem 184
Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Frank Neumann, and Madeleine Theile

Comparison-Based Adaptive Strategy Selection with Bandits in Differential Evolution 194
Álvaro Fialho, Raymond Ros, Marc Schoenauer, and Michèle Sebag

Fixed Parameter Evolutionary Algorithms and Maximum Leaf Spanning Trees: A Matter of Mutation 204
Stefan Kratsch, Per Kristian Lehre, Frank Neumann, and Pietro Simone Oliveto

An Archive Maintenance Scheme for Finding Robust Solutions 214
Johannes Kruisselbrink, Michael Emmerich, and Thomas Bäck

Experimental Supplements to the Theoretical Analysis of Migration in the Island Model 224
Jörg Lässig and Dirk Sudholt

General Scheme for Analyzing Running Times of Parallel Evolutionary Algorithms 234
Jörg Lässig and Dirk Sudholt

Negative Drift in Populations 244
Per Kristian Lehre

Log(λ) Modifications for Optimal Parallelism 254
Fabien Teytaud and Olivier Teytaud

The Linkage Tree Genetic Algorithm 264
Dirk Thierens

An Analysis of the XOR Dynamic Problem Generator Based on the Dynamical System 274
Renato Tinós and Shengxiang Yang

The Role of Degenerate Robustness in the Evolvability of Multi-agent Systems in Dynamic Environments 284
James M. Whitacre, Philipp Rohlfshagen, Axel Bender, and Xin Yao

Machine Learning, Classifier Systems, Image Processing

Evolutionary Learning of Technical Trading Rules without Data-Mining Bias 294
Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon

Using Computational Intelligence to Identify Performance Bottlenecks in a Computer System 304
Faraz Ahmed, Farrukh Shahzad, and Muddassar Farooq

Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation 314
Bernd Bischl, Igor Vatolkin, and Mike Preuss

Globally Induced Model Trees: An Evolutionary Approach 324
Marcin Czajkowski and Marek Krętowski

Open-Ended Evolutionary Robotics: An Information Theoretic Approach 334
Pierre Delarboulas, Marc Schoenauer, and Michèle Sebag

A Novel Similarity-Based Crossover for Artificial Neural Network Evolution 344
Mauro Dragoni, Antonia Azzini, and Andrea G.B. Tettamanzi

Indirect Encoding of Neural Networks for Scalable Go 354
Jason Gauci and Kenneth O. Stanley

Comparison-Based Optimizers Need Comparison-Based Surrogates 364
Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag

A Cooperative Coevolutionary Approach to Partitional Clustering 374
Mitchell A. Potter and Christine Couldrey

Feature Selection for Multi-purpose Predictive Models: A Many-Objective Task 384
Alan P. Reynolds, David W. Corne, and Michael J. Chantler

Incorporating Domain Knowledge into Evolutionary Computing for Discovering Gene-Gene Interaction 394
Stephen D. Turner, Scott M. Dudek, and Marylyn D. Ritchie

The Application of Pittsburgh-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies 404
Ryan J. Urbanowicz and Jason H. Moore

Threshold Selection, Mitosis and Dual Mutation in Cooperative Co-evolution: Application to Medical 3D Tomography 414
Franck P. Vidal, Evelyne Lutton, Jean Louchet, and Jean-Marie Rocchisani

Comparative Analysis of Search and Score Metaheuristics for Bayesian Network Structure Learning Using Node Juxtaposition Distributions 424
Yanghui Wu, John McCall, and David Corne

Analyzing the Credit Default Swap Market Using Cartesian Genetic Programming 434
Laleh Zangeneh and Peter J. Bentley

Memetic Algorithms, Hybridized Techniques, Meta and Hyperheurisics

A Memetic Cooperative Optimization Schema and Its Application to the Tool Switching Problem 445
Jhon Edgar Amaya, Carlos Cotta, and Antonio J. Fernández Leiva

Ownership and Trade in Spatial Evolutionary Memetic Games 455
Juan C. Burguillo and Ana Peleteiro

A Hyper-Heuristic Approach to Strip Packing Problems 465
Edmund K. Burke, Qiang Guo, and Graham Kendall

Asymptotic Analysis of Computational Multi-Agent Systems 475
Aleksander Byrski, Robert Schaefer, Maciej Smołka, and Carlos Cotta

Path-Guided Mutation for Stochastic Pareto Local Search Algorithms 485
Madalina M. Drugan and Dirk Thierens

Scheduling English Football Fixtures over the Holiday Period Using Hyper-heuristics 496
Jonathon Gibbs, Graham Kendall, and Ender Özcan

Graph Clustering Based Model Building 506
David Iclănzan and Dumitru Dumitrescu

How to Choose Solutions for Local Search in Multiobjective Combinatorial Memetic Algorithms 516
Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Yoshihiko Wakamatsu, and Yusuke Nojima

Secure and Task Abortion Aware GA-Based Hybrid Metaheuristics for Grid Scheduling 526
Joanna Kołodziej, Fatos Xhafa, and Marcin Bogdański

A Memetic Algorithm for the Pickup and Delivery Problem with Time Windows Using Selective Route Exchange Crossover 536
Yuichi Nagata and Shigenobu Kobayashi

Ant Based Hyper Heuristics with Space Reduction: A Case Study of the p-Median Problem 546
Zhilei Ren, He Jiang, Jifeng Xuan, and Zhongxuan Luo

A Study of Multi-parent Crossover Operators in a Memetic Algorithm 556
Yang Wang, Zhipeng Lü, and Jin-Kao Hao

A Hybrid Genetic Algorithm for the Traveling Salesman Problem Using Generalized Partition Crossover 566
Darrell Whitley, Doug Hains, and Adele Howe

A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-model 576
Saúl Zapotecas Martínez and Carlos A. Coello Coello

Multiobjective Optimization, Theoretical Aspects

Theoretically Investigating Optimal μ-Distributions for the Hypervolume Indicator: First Results for Three Objectives 586
Anne Auger, Johannes Bader, and Dimo Brockhoff

Convergence Rates of (1+1) Evolutionary Multiobjective Optimization Algorithms 597
Nicola Beume, Marco Laumanns, and Günter Rudolph

Tight Bounds for the Approximation Ratio of the Hypervolume Indicator 607
Karl Bringmann and Tobias Friedrich

Evolutionary Multiobjective Optimization Algorithm as a Markov System 617
Ewa Gajda, Robert Schaefer, and Maciej Smołka

A Natural Evolution Strategy for Multi-objective Optimization 627
Tobias Glasmachers, Tom Schaul, and Jürgen Schmidhuber

Solving Multiobjective Optimization Problem by Constraint Optimization 637
He Jiang, Shuyan Zhang, and Zhilei Ren

Enhancing Diversity for Average Ranking Method in Evolutionary Many-Objective Optimization 647
Miqing Li, Jinhua Zheng, Ke Li, Qizhao Yuan, and Ruimin Shen

Objective Space Partitioning Using Conflict Information for Many-Objective Optimization 657
Antonio López Jaimes, Hernán Aguirre, Kiyoshi Tanaka, and Carlos A. Coello Coello

How Crossover Speeds Up Evolutionary Algorithms for the Multi-criteria All-Pairs-Shortest-Path Problem 667
Frank Neumann and Madeleine Theile

Path Relinking on Many-Objective NK-Landscapes 677
Joseph M. Pasia, Hernán Aguirre, and Kiyoshi Tanaka

In Search of Equitable Solutions Using Multi-objective Evolutionary Algorithms 687
Pradyumn Kumar Shukla, Christian Hirsch, and Hartmut Schmeck

Stopping Criteria for Genetic Algorithms with Application to Multiobjective Optimization 697
Marcin Studniarski

Defining and Optimizing Indicator-Based Diversity Measures in Multiobjective Search 707
Tamara Ulrich, Johannes Bader, and Lothar Thiele

On Expected-Improvement Criteria for Model-Based Multi-objective Optimization 718
Tobias Wagner, Michael Emmerich, André Deutz, and Wolfgang Ponweiser

Parameter Tuning Boosts Performance of Variation Operators in Multiobjective Optimization 728
Simon Wessing, Nicola Beume, Günter Rudolph, and Boris Naujoks

Author Index 739

Table of Contents – Part II

Multiobjective Optimization, Models and Applications

A Novel Smart Multi-Objective Particle Swarm Optimisation Using Decomposition 1
Noura Al Moubayed, Andrei Petrovski, and John McCall

A Hybrid Scalarization and Adaptive ϵ-Ranking Strategy for Many-Objective Optimization 11
Hernán Aguirre and Kiyoshi Tanaka

pMODE-LD+SS: An Effective and Efficient Parallel Differential Evolution Algorithm for Multi-Objective Optimization 21
Alfredo Arias Montaño, Carlos A. Coello Coello, and Efrén Mezura-Montes

Improved Dynamic Lexicographic Ordering for Multi-Objective Optimisation 31
Juan Castro-Gutierrez, Dario Landa-Silva, and José Moreno Pérez

Privacy-Preserving Multi-Objective Evolutionary Algorithms 41
Daniel Funke and Florian Kerschbaum

Optimizing Delivery Time in Multi-Objective Vehicle Routing Problems with Time Windows 51
Abel Garcia-Najera and John A. Bullinaria

Speculative Evaluation in Particle Swarm Optimization 61
Matthew Gardner, Andrew McNabb, and Kevin Seppi

Towards Directed Open-Ended Search by a Novelty Guided Evolution Strategy 71
Lars Graening, Nikola Aulig, and Markus Olhofer

Consultant-Guided Search Algorithms with Local Search for the Traveling Salesman Problem 81
Serban Iordache

Many-Objective Test Problems to Visually Examine the Behavior of Multiobjective Evolution in a Decision Space 91
Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Noritaka Tsukamoto, and Yusuke Nojima

Preference-Based Multi-Objective Particle Swarm Optimization Using Desirabilities 101
Sanaz Mostaghim, Heike Trautmann, and Olaf Mersmann

GPGPU-Compatible Archive Based Stochastic Ranking Evolutionary Algorithm (G-ASREA) for Multi-Objective Optimization 111
Deepak Sharma and Pierre Collet

Hybrid Directional-Biased Evolutionary Algorithm for Multi-Objective Optimization 121
Tomohiro Shimada, Masayuki Otani, Hiroyasu Matsushima, Hiroyuki Sato, Kiyohiko Hattori, and Keiki Takadama

A Framework for Incorporating Trade-Off Information Using Multi-Objective Evolutionary Algorithms 131
Pradyumn Kumar Shukla, Christian Hirsch, and Hartmut Schmeck

Applications, Engineering and Economical Models

Topography-Aware Sensor Deployment Optimization with CMA-ES 141
Vahab Akbarzadeh, Albert Hung-Ren Ko, Christian Gagné, and Marc Parizeau

Evolutionary Optimization on Problems Subject to Changes of Variables 151
Richard Allmendinger and Joshua Knowles

On-Line Purchasing Strategies for an Evolutionary Algorithm Performing Resource-Constrained Optimization 161
Richard Allmendinger and Joshua Knowles

Parallel Artificial Immune System in Optimization and Identification of Composite Structures 171
Witold Beluch, Tadeusz Burczyński, and Wacław Kuś

Bioreactor Control by Genetic Programming 181
Dimitris C. Dracopoulos and Riccardo Piccoli

Solving the One-Commodity Pickup and Delivery Problem Using an Adaptive Hybrid VNS/SA Approach 189
Manar I. Hosny and Christine L. Mumford

Testing the Dinosaur Hypothesis under Empirical Datasets 199
Michael Kampouridis, Shu-Heng Chen, and Edward Tsang

Fractal Gene Regulatory Networks for Control of Nonlinear Systems 209
Jean Krohn and Denise Gorse

An Effective Hybrid Evolutionary Local Search for Orienteering and Team Orienteering Problems with Time Windows 219
Nacima Labadi, Jan Melechovský, and Roberto Wolfler Calvo

Discrete Differential Evolution Algorithm for Solving the Terminal Assignment Problem 229
Eugénia Moreira Bernardino, Anabela Moreira Bernardino, Juan Manuel Sánchez-Pérez, Juan Antonio Gómez-Pulido, and Miguel Angel Vega-Rodríguez

Decentralized Evolutionary Agents Streamlining Logistic Network Design 240
Stephan Otto and Tobias Bannenberg

Testing the Permutation Space Based Geometric Differential Evolution on the Job-Shop Scheduling Problem 250
Antonin Ponsich and Carlos A. Coello Coello

New Uncertainty Handling Strategies in Multi-objective Evolutionary Optimization 260
Thomas Voß, Heike Trautmann, and Christian Igel

Evolving a Single Scalable Controller for an Octopus Arm with a Variable Number of Segments 270
Brian G. Woolley and Kenneth O. Stanley

Multi-agent Systems and Parallel Approaches

An Island Model for the No-Wait Flow Shop Scheduling Problem 280
Istvan Borgulya

Environment-Driven Embodied Evolution in a Population of Autonomous Agents 290
Nicolas Bredeche and Jean-Marc Montanier

Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning 300
Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang

EvoShelf: A System for Managing and Exploring Evolutionary Data.... 310
Timothy Davison, Sebastian von Mammen, and Christian Jacob

Differential Evolution Algorithms with Cellular Populations 320
Bernabé Dorronsoro and Pascal Bouvry

Flocking in Stationary and Non-Stationary Environments: A Novel Communication Strategy for Heading Alignment 331
Eliseo Ferrante, Ali Emre Turgut, Nithin Mathews, Mauro Birattari, and Marco Dorigo

Evolution of XPath Lists for Document Data Selection 341
Pablo García-Sánchez, Juan J. Merelo Guervós, Pedro Ángel Castillo, Jesús González, Juan L. Jiménez Laredo, Antonio M. Mora García, and Maria I. García Arenas

PMF: A Multicore-Enabled Framework for the Construction of Metaheuristics for Single and Multiobjective Optimization 351
Deon Garrett

Parallel Evolutionary Approach of Compaction Problem Using MapReduce 361
Doina Logofătu and Dumitru Dumitrescu

Ant Colony Optimization with Immigrants Schemes in Dynamic Environments 371
Michalis Mavrovouniotis and Shengxiang Yang

Secret Key Specification for a Variable-Length Cryptographic Cellular Automata Model 381
Gina M.B. Oliveira, Luiz G.A. Martins, Giordano B. Ferreira, and Leonardo S. Alt

Variable Neighborhood Search and Ant Colony Optimization for the Rooted Delay-Constrained Minimum Spanning Tree Problem 391
Mario Ruthmair and Günther R. Raidl

Adaptive Modularization of the MAPK Signaling Pathway Using the Multiagent Paradigm 401
Abbas Sarraf Shirazi, Sebastian von Mammen, and Christian Jacob

Genetic Computing and Games

Experimental Comparison of Methods to Handle Boundary Constraints in Differential Evolution 411
Jarosław Arabas, Adam Szczepankiewicz, and Tomasz Wroniak

Entropy-Driven Evolutionary Approaches to the Mastermind Problem 421
Carlos Cotta, Juan J. Merelo Guervós, Antonio M. Mora García, and Thomas Philip Runarsson

Evolutionary Detection of New Classes of Equilibria. Application in Behavioral Games 432
Dumitru Dumitrescu, Rodica Ioana Lung, Réka Nagy, Daniela Zaharie, Attila Bartha, and Doina Logofătu

Design and Comparison of Two Evolutionary Approaches for Solving the Rubik's Cube 442
Nail El-Sourani and Markus Borschbach

Statistical Analysis of Parameter Setting in Real-Coded Evolutionary Algorithms 452
Maria I. García Arenas, Pedro Ángel Castillo Valdivieso, Antonio M. Mora García, Juan J. Merelo Guervós, Juan L. Jiménez Laredo, and Pablo García-Sánchez

Performance of Network Crossover on NK Landscapes and Spin Glasses 462
Mark Hauschild and Martin Pelikan

Promoting Phenotypic Diversity in Genetic Programming 472
David Jackson

A Genetic Programming Approach to the Matrix Bandwidth-Minimization Problem 482
Behrooz Koohestani and Riccardo Poli

Using Co-solvability to Model and Exploit Synergetic Effects in Evolution 492
Krzysztof Krawiec and Paweł Lichocki

Fast Grammar-Based Evolution Using Memoization 502
Martin Luerssen and David Powers

Evolution of Conventions and Social Polarization in Dynamical Complex Networks 512
Enea Pestelacci and Marco Tomassini

Evolving Strategies for Updating Pheromone Trails: A Case Study with the TSP 523
Jorge Tavares and Francisco B. Pereira

The Role of Syntactic and Semantic Locality of Crossover in Genetic Programming 533
Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, and Bob McKay

The *Layered Learning* Method and Its Application to Generation of Evaluation Functions for the Game of Checkers 543
Karol Walędzik and Jacek Mańdziuk

Author Index 553

Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem

Süntje Böttcher, Benjamin Doerr, and Frank Neumann

Algorithms and Complexity, Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. We reconsider a classical problem, namely how the (1+1) evolutionary algorithm optimizes the LeadingOnes function. We prove that if a mutation probability of p is used and the problem size is n, then the optimization time *is*

$$\frac{1}{2p^2}((1-p)^{-n+1}-(1-p)).$$

For the standard value of $p = 1/n$, this is approximately $0.86n^2$. As our bound shows, this mutation probability is not optimal: For $p \approx 1.59/n$, the optimization time drops by more than 16% to approximately $0.77n^2$.

Our method also allows to analyze mutation probabilities depending on the current fitness (as used in artificial immune systems). Again, we derive an exact expression. Analysing it, we find a fitness dependent mutation probability that yields an expected optimization time of approximately $0.68n^2$, another 12% improvement over the optimal mutation rate. In particular, this is the first example where an adaptive mutation rate provably speeds up the computation time. In a general context, these results suggest that the final word on mutation probabilities in evolutionary computation is not yet spoken.

1 Introduction

Evolutionary algorithms [1] are a class of randomized algorithms [2] that have found many applications in different problem domains. Understanding the behavior of this kind of algorithms is a challenging and difficult task due to different random components that are involved. Considering evolutionary algorithms as randomized algorithms from a theoretical point of view allows to analyze them with respect to their runtime behavior in a rigorous way.

Analyzing the runtime behavior of evolutionary algorithms has become a major branch in the theoretical analysis of these algorithms. Starting with results on simple pseudo-Boolean functions (see e. g. [3,4]), different results have been obtained for classical combinatorial optimization problems such as shortest paths, minimum spanning trees, or maximum matchings (see [5] for an overview).

Almost all results mentioned are asymptotic ones. In particular, most of the results give little information about the leading constants, which are of high interest when using such methods for realistic input sizes. There are only a few results that give at least give a bound on the leading constant.

It is folklore that the (1+1) evolutionary algorithm finds the optimum of the most simple test function OneMax, counting the number of 1-bits in the bit-string of length n, in time at most $(1+o(1))en\ln(n)$. This follows immediately from an elementary proof using coupon collector type arguments. However, the corresponding lower bound was only recently given in [6].

For linear functions, Jägersküpper was the first to give a bound including the constant, namely $(1+o(1))2.02en\ln(n)$. This was improved to $(1+o(1))1.39en\ln(n)$ in [7]. There also the $(1-o(1))en\ln(n)$ lower bound for OneMax was extended to all linear functions with non-zero coefficients.

In this paper, we present an *exact* analysis for the function LeadingOnes introduced by Rudolph [8]. This function is one of the classical test problems and has been extensively studied (see e. g. [3,9,10]). Our analysis yields an exact formula for the expected number of iterations (*expected optimization time*) needed to find the optimum of the LeadingOnes function. Let n be the problem size, that is, the length of the bit-strings forming the search space, and let p be a mutation probability. The typical choice for the mutation probability is $p = 1/n$ [3], but our analysis works for all possible values.

Given n and p, we show that the expected optimization time is

$$\tfrac{1}{2p^2}((1-p)^{-n+1}-(1-p)).$$

From this formula, we see that the standard value for the mutation probability of $p = 1/n$ leads to an expected optimization time of less than $\frac{1}{2}(e-1)n^2 \approx 0.86n^2$, where the first expression is tight up to terms of order n. This improves the best known upper bound of en^2 and lower bound of $n^2/6$ given in [3].

Our formula also allows to determine the optimal mutation probability. This in particular shows that the standard mutation probability of $1/n$ is not optimal. For $p \approx 1.59/n$, the expected optimization time drops by more than 16% to approximately $0.77n^2$ (for n sufficiently large).

We should add that, while $p = 1/n$ is generally the preferred good choice for the mutation probability, there are examples known where mutation probabilities even having a different order of magnitude like $\Theta(\log(n)/n)$ are much better than $1/n$, see [11]. However, these example functions look custom-tailored to demonstrate this effect.

Our method also allows to analyze mutation probabilities depending on the current fitness. Such ideas have been used under the name *artificial immune systems* in [12,13]. However, contrary to the common belief that one should start the optimization process with a larger mutation probability and then successively reduce it, none of these works could show that varying the mutation probability yields an improvement. This is different in our analysis. If we choose the mutation probability to be the reciprocal of the current fitness value (that is, the number of leading ones), then the expected optimization time again reduces to a number T, which satisfies $\frac{1}{4}en^2 - \frac{1}{4}en \le T \le \frac{1}{4}en^2 + \frac{1}{4}en$. We do have an exact expression for T as well.

2 Problem and Algorithms

Our aim is to study the optimization behavior of a simple randomized search heuristic for the well-known pseudo-Boolean function $f = \textsc{LeadingOnes} : \{0,1\}^n \to \mathbb{N}_0$ defined by

$$f(x) = \textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

We investigate a simple baseline evolutionary algorithm called (1+1) EA. It works with a population size of 1 and produces in each iteration one offspring by mutation. If not worse, this offspring forms the new population. As mutation, we use standard bit mutation, that is, each bit is flipped independently with probability p. Note that many authors implicitly assume $p = 1/n$, whereas we do allow all values for p.

Algorithm 1 ((1+1) EA with mutation probability p)

```
1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. Flip each bit of $x$ independently with probability $p$ to produce an offspring $y$.
3. If $f(y) \geq f(x)$ then $x := y$.
4. Go to 2.
```

For our theoretical investigations we consider the expected number of iterations until our algorithms produces an optimal solution for the LeadingOnes problem for the first time. This is called the *expected optimization time*. Note that the mutation rate p in Algorithm 1 is constant for given n, that is, it does not vary over time or with respect to the current fitness of the population.

It is a common belief that one can gain improvements by varying the mutation probability. In particular, it is said that in the early stages of the optimization process the mutation probability should be larger to faster approach the optimum, whereas at the end, it should be smaller to avoid large jumps that potentially lead away from the optimum.

Such a varying mutation probability could be implemented by making the mutation probability depend on the time the optimization process already lasts. A difficulty here is that one would need a good guess on the expected optimization time.

An alternative approach is to let the mutation probability depend on the current fitness value. Here, of course, one would need a guess on the maximum possible fitness. Still, this seems to be easier than guessing the expected optimization time.

To incorporate this concept into the (1+1) EA above, let $p : \{0,\dots,n\} \to (0,1)$ be a function. We think of $p_k := p(k)$ being the mutation probability used when we have a LeadingOnes-value of k. This leads to the following Algorithm 2, which we call Adaptive (1+1) EA.

Algorithm 2 (Adaptive (1+1) EA)

```
1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. Flip each bit of $x$ independently with probability $p_{f(x)}$ to produce an offspring $y$.
3. If $f(y) \geq f(x)$ then $x := y$.
4. Go to 2.
```

Algorithms such as artificial immune systems make use of such adaptive mutation rates. For some recent asymptotic results on the runtime of these algorithms, we refer to [12,13].

3 A Formula for the Expected Optimization Time

The key to our analysis (and the topic of this section) is noting that the expected optimization time can be fully described by the expected times needed to improve the fitness (conditional on a particular fitness level). The latter can easily be expressed via the (current) mutation probability.

3.1 Combining the States

We first exploit the fact that we start with a random individual. This allows a simplified view, namely that we do not have to regard the remaining expected optimization time for each possible individual, but only for a few random states.

Lemma 1. *Let x be a (random) individual produced by Algorithm 1 or 2 within a fixed number t of iterations. Let $f(x)$ denote its fitness. Then $x_{f(x)+1} = 0$ with probability one. For all $i > f(x)+1$, we have $\Pr[x_i = 1] = \Pr[x_i = 0] = \frac{1}{2}$ independent from all other bits.*

Proof. Simple induction over the time t.

3.2 Improvement Times

As discussed above, the base of our analysis is (later) noting that the expected optimization time can be expressed in terms of the waiting times for an improvement. We now use Lemma 1 to, very elementarily, determine these waiting times relative to the current fitness.

Lemma 2. *Let $0 \leq j < n$. Let $x \in \{0,1\}^n$ be random subject to $f(x) =$ LeadingOnes $= j$. Let y be the offspring of x obtained by mutation with some mutation probability p_j. Then the probability that y is strictly fitter than x, the improvement probability, is $\Pr[f(y) > f(x)] = (1-p_j)^j p_j$.*

Proof. Since $f(x) = j$, a mutation step increases the f-value if and only if the first j bits do not change and the $(j+1)$st bit does change. All other bits are irrelevant.

The actions of the different bits are independent. A bit flips with probability p_j and remains unchanged with probability $1-p_j$. Thus the probability of improving the f-value is $\Pr[f(y) > f(x)] = (1-p_j)^j p_j$.

Given this improvement probability, we can simply compute the waiting time for such an improvement. Denote by A_i the expected time needed to find an improvement given that the initial solution has a fitness of $n-i$ (that is, we are i levels from the optimum). We compute the A_i.

Theorem 1. *Let $x \in \{0,1\}^n$ be random with $f(x) < n$. Then the time $A_{n-f(x)}$ we need to wait for an improvement is*

$$A_{n-f(x)} = \frac{1}{\Pr[f(y) > f(x)]}.$$

Consequently, for all $1 \leq i \leq n$, we have

$$A_i = \frac{1}{(1-p_{n-i})^{n-i} p_{n-i}}.$$

Proof. Follows from elementary properties of the geometric distribution and the previous lemma.

3.3 Expected Optimization Time

We now express the expected optimization time in terms of the improvement times just determined. Note that since the latter depend on the (possibly varying) mutation probability, this immediately tells us how the mutation probability influences the expected optimization time.

We denote by T_{n-i} the expected number of steps needed to find the optimum starting from a random initial individual with f-value i. Obviously, at most $n-i$ improvements are necessary.

Lemma 3. *The time needed to find the optimum given a random solution with f-value $n-i$ is*

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j,$$

where A_i is the improvement time as defined in the previous subsection and $T_0 = 0$.

Proof. For $i = 0$, the time we need to finish obviously is $T_0 = 0$. For $0 < i \leq n$, the remaining time is given by the time for the next improvement, A_i, plus the expected remaining time conditional on this improvement. Let $\bar{x}$ denote the individual right after the improvement to an f-value of more than $n-i$. Since the bits $n-i+2, \ldots, n$ are still random, we have $\Pr[f(\bar{x}) = n-i+j] = 2^{-j}$ for $j = 1, \ldots, i-1$ and $\Pr[f(\bar{x}) = n] = 2^{-(i-1)}$. Since $T_0 = 0$, we may write

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j. \qquad \square$$

We now express the remaining optimization time T_i fully via the improvement times A_j.

Theorem 2. *The time needed to find the optimum given a random solution with f-value $n - i$ is*

$$T_i = A_i + \frac{1}{2}\sum_{j=1}^{i-1} A_j.$$

Proof. By induction we show that our claim

$$T_i = A_i + \sum_{j=0}^{i-1} 2^{j-i} T_j = A_i + \frac{1}{2}\sum_{j=1}^{i-1} A_j$$

holds for all $0 < i \leq n$.

For $i = 1$ we have $T_1 = A_1 + \frac{1}{2}\sum_{j=1}^{0} A_j$.

Assume that $T_k = A_k + \frac{1}{2}\sum_{j=1}^{k-1} A_j$ holds for all $0 < k \leq i$. Then

$$\begin{aligned} T_{i+1} &= A_{i+1} + \sum_{j=0}^{i} 2^{j-(i+1)} T_j = A_{i+1} + \frac{1}{2}T_i + \frac{1}{2}\sum_{j=0}^{i-1} 2^{j-i} T_j \\ &= A_{i+1} + T_i - \frac{1}{2}T_i + \frac{1}{2}\sum_{j=0}^{i-1} 2^{j-i} T_j = A_{i+1} + T_i - \frac{1}{2}A_i \\ &= A_{i+1} + A_i + \frac{1}{2}\sum_{j=0}^{i-1} A_j - \frac{1}{2}A_i \\ &= A_{i+1} + \frac{1}{2}\sum_{j=1}^{i} A_j. \end{aligned}$$

□

The bound of Theorem 2 matches our intuition that all improvements apart from the current one produce a waiting time only with probability 1/2.

4 The Optimal Fixed Mutation Rate

We first investigate the optimal choice for the mutation rate when working with a fixed mutation rate. Depending on the chosen mutation rate p, we compute the expected optimization time of the (1+1) EA. Remember that the time for an improvement for LeadingOnes is $A_i = \frac{1}{p}(1-p)^{i-n}$ and the overall expected optimization time is $T = \frac{1}{2}\sum_{i=1}^{n} A_i$. This leads to the following result.

Theorem 3. *The expected optimization time of (1+1) EA with fixed mutation rate p for* LeadingOnes *is*

$$T = \frac{1}{2p^2}\left[(1-p)^{1-n} - (1-p)\right].$$

Proof. Using our previous observation, we can calculate the expected optimization time directly and get

$$\begin{aligned}
T &= \frac{1}{2}\sum_{i=1}^{n} A_i = \frac{1}{2}\sum_{i=1}^{n}\frac{1}{p}(1-p)^{i-n} \\
&= \frac{1}{2p}\sum_{i=0}^{n-1}(1-p)^{i-n+1} = \frac{1}{2p}(1-p)^{1-n}\sum_{i=0}^{n-1}(1-p)^{i} \\
&= \frac{1}{2p}(1-p)^{1-n}\cdot\frac{1-(1-p)^n}{1-(1-p)} \\
&= \frac{1}{2p^2}\left[(1-p)^{1-n}-(1-p)\right]. \qquad \square
\end{aligned}$$

Based on the previous theorem, we can determine the optimal choice of p for (1+1) EA and LeadingOnes. To do this we compute the derivative of T with respect to p. In particular, we solve $\frac{d}{dp}T = 0$. This leads to

$$-\frac{1}{p^3}\left[(1-p)^{1-n}-(1-p)\right]+\frac{1}{2p^2}\left[-(1-n)(1-p)^{-n}+1\right]=0.$$

We cannot solve this equation in an algebraic way. Therefore, we provide a numerical approximation. This gives that the optimal mutation rate p converges to a value around $\frac{1.5936}{n}$, which implies an expected optimization time of $\approx 0.77201n^2$ for n sufficiently large. Compared to this, the expected optimization time is $\approx 0.85914n^2$ when choosing $p = 1/n$. Therefore, the optimal mutation rate leads to an improvement of 16.1% compared to the standard choice.

5 The Optimal Adaptive Mutation Rate

After having investigated the optimal mutation rate for (1+1) EA, we want to examine whether an adaptive mutation rate which depends on the fitness of the currently best solution can lead to further improvements.

With our general framework we can optimize the remaining time T and examine the Adaptive (1+1) EA in the following.

As A_i is a function in p_i, the overall optimization time $T = \frac{1}{2}\sum_{i=1}^{n} A_i$ is a function in $(p_1,\ldots,p_n)$. Let $(p_1^*,\ldots,p_n^*)$ minimize T, thus $\frac{\partial}{\partial p_i}T(p_1^*,\ldots,p_n^*) = 0$. Note that the indices are renamed for the sake of simplicity. Here, p_i is the mutation probability if the currently best solution has fitness $f(x) = n - i$. In order to determine the optimal choice of the optimal adaptive mutation rates, we compute the partial derivatives according to the overall expected optimization time T.

We get

$$\begin{aligned}\frac{\partial}{\partial p_i} T(p_1,\ldots,p_n) &= \frac{\partial}{\partial p_i} \frac{1}{2}\sum_{i=1}^{n} A_i \\ &= \frac{1}{2}\sum_{i=1}^{n} \frac{\partial}{\partial p_i} A_i\end{aligned}$$

In order to find the optimal value of p_i, we need to solve

$$\frac{1}{2}\sum_{i=1}^{n} \frac{\partial}{\partial p_i} A_i = 0.$$

Remember that A_i is the waiting time for one improvement, thus $A_i \geq 0$. Hence it follows that this equation holds if and only if $\frac{\partial}{\partial p_i} A_i = 0$ holds for all i. Thus we have to compute $\frac{\partial}{\partial p_i} A_i$.

Theorem 4. *Let $A_i = \frac{1}{p_i}(1-p_i)^{i-n}$. Then the optimal mutation rate is*

$$p_i = \frac{1}{n-i+1}.$$

Proof. We calculate the derivative

$$\frac{\partial}{\partial p_i} A_i = -\frac{1}{p_i^2}(1-p_i)^{i-n} - \frac{1}{p_i}(i-n)(1-p_i)^{i-n-1}$$

and get the optimal mutation rate by resolving

$$\begin{aligned}& -\frac{1}{p_i^2}(1-p_i)^{i-n} - \frac{1}{p_i}(i-n)(1-p_i)^{i-n-1} = 0 \\ \iff & -(1-p_i)^{i-n-1}\left[\frac{1-p_i}{p_i^2} + \frac{i-n}{p_i}\right] = 0 \\ \iff & \frac{1}{p_i^2} + \frac{i-n-1}{p_i} = 0 \\ \iff & 1 + p_i(i-n-1) = 0.\end{aligned}$$

Hence, we get $p_i = \frac{1}{n-i+1}$. □

Thus we have shown that $p_i = \frac{1}{n-i+1}$ is an optimal mutation rate for the Adaptive (1+1) EA for optimizing LeadingOnes. We use this mutation rate for determining the expected optimization time for Adaptive (1+1) EA. First, we consider the time to reach an improvement in dependence of p_i.

The expected waiting time for an improvement is given by

$$\begin{aligned}A_i = \frac{1}{p_i}(1-p_i)^{i-n} &= \frac{1}{\frac{1}{n-i+1}}\left(1-\frac{1}{n-i+1}\right)^{i-n} \\ &= \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1).\end{aligned}$$

Using this expression, we can compute almost matching upper and lower bounds on the expected optimization time.

Theorem 5. *Let $p_{f(x)} = \frac{1}{f(x)+1}$, then the expected optimization time of the Adaptive (1+1) EA is upper bounded by $\frac{e}{4}n^2 + \frac{e}{4}n$.*

Proof. Let $A_i = \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1)$. Then

$$\begin{aligned}
T &= \sum_{i=0}^{n} 2^{i-n-1} T_i \\
&= \frac{1}{2}\sum_{i=1}^{n} A_i \\
&= \frac{1}{2}\sum_{i=1}^{n} \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1) \\
&\le \frac{1}{2}\sum_{i=1}^{n} e(n-i+1) \\
&= \frac{e}{4}n^2 + \frac{e}{4}n.
\end{aligned}$$

□

Similarly we can compute a lower bound for the expected optimization time.

Theorem 6. *Let $p_{f(x)} = \frac{1}{f(x)+1}$, then the expected optimization time of the Adaptive (1+1) EA is lower bounded by $\frac{e}{4}n^2 - \frac{e}{4}n$.*

Proof. Let $A_i = \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1)$. Then, as above,

$$\begin{aligned}
T &= \frac{1}{2}\sum_{i=1}^{n} \left(\frac{n-i+1}{n-i}\right)^{n-i}(n-i+1) \\
&\ge \frac{1}{2}\sum_{i=1}^{n} \left(\frac{n-i}{n-i+1}\right) e(n-i+1) \\
&= \frac{e}{4}n^2 - \frac{e}{4}n.
\end{aligned}$$

□

Summarizing the results for the adaptive mutation rate, we get an improvement of 12.0% compared to the expected optimization time using the optimal constant mutation rate of approximately $p = \frac{1.59}{n}$ and an overall improvement of 20.9% to the expected optimization time with standard mutation rate $p = \frac{1}{n}$.

6 Conclusions

Most of the theoretical studies on the runtime behavior of evolutionary algorithms deal with asymptotic results. We have presented an exact analysis for the (1+1) EA and the test function LeadingOnes. This in particular showed that one may speed up the computation by 16% when using the optimal mutation rate compared to the standard mutation rate of $1/n$. Furthermore, our investigations on the adaptive mutation rate show that a further improvement of 12% is possible when using such an approach. We are optimistic that further exact investigations will allow a deeper understanding of the right parameter setting for evolutionary algorithms.

References

1. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2007)
2. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
3. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. 276, 51–81 (2002)
4. Jansen, T., Wegener, I.: Evolutionary algorithms—how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans. Evolutionary Computation 5, 589–599 (2001)
5. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4, 281–293 (2007)
6. Doerr, B., Fouz, M., Witt, C.: Quasirandom evolutionary algorithms. In: Proceedings of GECCO 2010. ACM, New York (to appear 2010)
7. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: Proceedings of CEC 2010. IEEE, Los Alamitos (to appear 2010)
8. Rudolph, G.: Convergence properties of evolutionary algorithms. Kovac, Hamburg (1997)
9. Witt, C.: Runtime analysis of the (μ+1) EA on simple pseudo-Boolean functions. Evolutionary Computation 14, 65–86 (2006)
10. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence 3, 35–68 (2009)
11. Jansen, T., Wegener, I.: On the choice of the mutation probability for the (1+1) EA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 89–98. Springer, Heidelberg (2000)
12. Zarges, C.: Rigorous runtime analysis of inversely fitness proportional mutation rates. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 112–122. Springer, Heidelberg (2008)
13. Zarges, C.: On the utility of the population size for inversely fitness proportional mutation rates. In: Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009), pp. 39–46. ACM, New York (2009)

Mirrored Sampling and Sequential Selection for Evolution Strategies

Dimo Brockhoff[1], Anne Auger[1], Nikolaus Hansen[1], Dirk V. Arnold[2], and Tim Hohm[3]

[1] TAO Team, INRIA Saclay, LRI Paris Sud University, 91405 Orsay Cedex, France
firstname.lastname@inria.fr
[2] Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada B3H 1W5
dirk@cs.dal.ca
[3] Department of Medical Genetics, University of Lausanne, Lausanne, Switzerland
tim.hohm@unil.ch

Abstract. This paper reveals the surprising result that a single-parent non-elitist evolution strategy (ES) can be locally faster than the (1+1)-ES. The result is brought about by *mirrored sampling* and *sequential selection*. With mirrored sampling, two offspring are generated symmetrically or *mirrored* with respect to their parent. In sequential selection, the offspring are evaluated sequentially and the iteration is concluded as soon as one offspring is better than the current parent. Both concepts complement each other well. We derive exact convergence rates of the $(1, \lambda)$-ES with mirrored sampling and/or sequential selection on the sphere model. The log-linear convergence of the ES is preserved. Both methods lead to an improvement and in combination the (1,4)-ES becomes about 10% faster than the (1+1)-ES. Naively implemented into the CMA-ES with recombination, mirrored sampling leads to a bias on the step-size. However, the (1,4)-CMA-ES with mirrored sampling and sequential selection is unbiased and appears to be faster, more robust, and as local as the (1+1)-CMA-ES.

1 Introduction

Evolution strategies (ESs) are robust stochastic search algorithms designed to minimize objective functions f that map a continuous search space $\mathbb{R}^d$ into $\mathbb{R}$. The $(1, \lambda)$-ES is a non-elitist and rather local search algorithm where λ candidate solutions, the offspring, are created from a single parent, $\boldsymbol{X}_k \in \mathbb{R}^d$. The λ offspring are generated by adding λ *independent* random vectors $(\boldsymbol{\mathcal{N}}_k^i)_{1 \leq i \leq \lambda}$ to $\boldsymbol{X}_k$. Then, the *best* of the λ offspring $\boldsymbol{X}_k + \boldsymbol{\mathcal{N}}_k^i$, i.e., the solution with the lowest objective function value, is *selected* to become the next parent $\boldsymbol{X}_{k+1}$. The elitist version of this algorithm, the $(1 + \lambda)$-ES, selects $\boldsymbol{X}_{k+1}$ as the best among the λ offspring *and* the parent $\boldsymbol{X}_k$.

The (1+1)-ES is arguably the most local, and the locally fastest, variant of an evolution strategy. In a local search scenario, the (1+1)-CMA-ES outperforms its non-elitist counterparts typically by a factor of about 1.5 [10]. Also in the BBOB-2009 benchmarking exercise[1], the (1+1)-CMA-ES, restarted many times, performed surprisingly well on two highly multi-modal functions with weak overall structure (f_{21} and f_{22}).

[1] http://coco.gforge.inria.fr/doku.php?id=bbob-2009

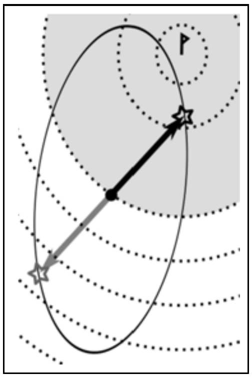
However, we regard elitist selection generally as less robust, as for instance witnessed by its poor performance on the BBOB-2009 noisy testbed [5] (a single outlier fitness measurement can survive for an arbitrarily long time) or its failure on the attractive sector function f_6. Therefore, we pursue the objective to construct local non-elitist ESs with a convergence speed competitive to the (1+1)-ES and without the disadvantages of elitist selection. This is achieved by derandomization of random samples and a greedy acceptance mechanism in the $(1, \lambda)$-ES with (very) small λ.

Derandomization of random numbers has been previously introduced as antithetic variables for isotropic samples [11] and for the CMA-ES by replacing the sequence of uniform random numbers used for sampling a multivariate normal distribution by scrambling-Halton and Sobol sequences [3, ref. [27]]. However, both approaches can introduce a bias on the step-size update as we will discuss later.

Objectives of this paper. In this paper we present the concepts of mirrored (derandomized, antithetic) sampling and sequential selection within evolution strategies. We derive theoretical results on their convergence rates. We discuss their implementation into CMA-ES, in particular with respect to the question of an unbiased step-size, and present some empirical performance results.

2 Mirrored Sampling and Sequential Selection

In this section, we present the concepts of mirrored samples and sequential selection, which we have recently benchmarked in the special case of the (1,2)- and the (1,4)-CMA-ES [3, ref. [3–10]]. Here, we describe both concepts for the $(1 \overset{+}{,} \lambda)$-ES.



```
given: $\boldsymbol{X}_k \in \mathbb{R}^d, j \in \mathbb{N}, \lambda \in \mathbb{N}^+, f : \mathbb{R}^d \to \mathbb{R}$
$i \leftarrow 0$
while $i < \lambda$ do
    $i \leftarrow i + 1, j \leftarrow j + 1$
    if mirrored sampling and $j \equiv 0 \pmod 2$ then
        $\boldsymbol{X}_k^i = \boldsymbol{X}_k - \boldsymbol{\mathcal{N}}_k^{i-1}$    use previous sample
    else
        $\boldsymbol{X}_k^i = \boldsymbol{X}_k + \boldsymbol{\mathcal{N}}_k^i$
    if sequential selection and $f(\boldsymbol{X}_k^i) < f(\boldsymbol{X}_k)$ then
        $j \leftarrow 0$    start with a new sample in the next iteration
        break;
end while
return $\boldsymbol{X}_{k+1} = \mathrm{argmin}\{f(\boldsymbol{X}_k^1), \dots, f(\boldsymbol{X}_k^i)\}$
```

Fig. 1. Left: If for a unimodal function with convex sub-level sets, a sampled solution is better than its parent (dark arrow into shaded region of better objective function values), the mirrored one (gray) is always worse. **Right**: Pseudocode for one iteration step of mirrored sampling and sequential selection, returning the new parent $\boldsymbol{X}_{k+1}$. $\boldsymbol{\mathcal{N}}_{k+1}^0 = \boldsymbol{\mathcal{N}}_k^\lambda$ and before the first iteration, j is even. The pseudocode captures all combinations with/without mirrored sampling and/or sequential selection. The last line depicts comma-selection but can be replaced by plus selection.

Mirrored sampling uses a single random vector instantiation to create two offspring, one by adding and the other by subtracting the vector. In **Fig. 1**, the $(1, \lambda_{\mathrm{m}})$-ES is given, but mirrored sampling is entirely independent of the selection scheme.

We denote by $\boldsymbol{X}_k$ the parent at iteration k and consider the $(1 \overset{+}{,} \lambda_{\mathrm{m}})$-ES with even λ. In each iteration k, we sample $\lambda/2$ random vectors $(\boldsymbol{\mathcal{N}}_k^{2i-1})_{1 \leq i \leq \lambda/2}$. A given vector $\boldsymbol{\mathcal{N}}_k^{2i-1}$ is used for two offspring that equal $\boldsymbol{X}_k + \boldsymbol{\mathcal{N}}_k^{2i-1}$ and $\boldsymbol{X}_k - \boldsymbol{\mathcal{N}}_k^{2i-1}$. They are thus *mirrored* or *symmetric* with respect to the parent $\boldsymbol{X}_k$. For odd λ, every other iteration, the first offspring uses the mirrored last vector from the previous iteration, see j in **Fig. 1**. Consequently, in the (1+1_{m})-ES, a mirrored sample is used if and only if the iteration index is even. Note that in the $(1 \overset{+}{,} \lambda_{\mathrm{m}})$, two mirrored offspring are entirely dependent and, in a sense, complementary, similarly to antithetic variables for Monte-Carlo numerical integration [3, ref. [14]].

Mirrored sampling has also been used in an attempt to increase the robustness of Evolutionary Gradient Search (EGS) [1]. In contrast to its use here, its utility in EGS lies in the ability to compute a stochastic gradient approximation by means of finite differences that do not involve the (possibly noisy) fitness value of a single parental solution. With a large sample size, the use of mirrored samples also increases the rate of convergence of EGS on the sphere model.

Sequential selection. Evaluating a sampled solution and its mirrored counterpart can result in unnecessary function evaluations: on unimodal objective functions with convex sub-level sets, $\{x \mid f(x) \leq c\}$ for $c \in \mathbb{R}$, such as the sphere function, $f(x) = \|x\|^2$, the mirrored solution $\boldsymbol{X}_k - \boldsymbol{\mathcal{N}}$ must be worse than the parent $\boldsymbol{X}_k$, if $\boldsymbol{X}_k + \boldsymbol{\mathcal{N}}$ was better than $\boldsymbol{X}_k$, see **Fig. 1**. Sequential selection, originally introduced to save such unnecessary function evaluations, is however *independent of mirrored sampling*: in sequential selection, the offspring are evaluated one by one, compared to their parent, and the iteration is concluded immediately if one offspring is better than its parent. If the first $\lambda - 1$ offspring are worse than the parent, the original selection scheme is applied.

Sequential selection applied to $(1 + \lambda)$-selection coincides with (1+1)-selection: in both cases any offspring is accepted if and only if it is better than the parent[2]. The $(1, \lambda)$-ES with sequential selection is denoted as $(1, \lambda^{\mathrm{s}})$-ES and shown in **Fig. 1**. Note that an alternative view of the (1,λ^{s})-ES is as (1+1)-ES that periodically replaces the parent if no improvement is found after λ candidate samples.

Combining mirrored sampling and sequential selection. As the concepts of mirrored sampling and sequential selection are independent, they can be applied simultaneously. With plus selection we obtain the (1+$1_{\mathrm{m}}^{\mathrm{s}}$)-ES, independently of λ. Compared to the (1+1_{m})-ES, the (1+$1_{\mathrm{m}}^{\mathrm{s}}$)-ES does not use the mirrored vector after a success. With comma selection, the resulting algorithm is denoted by $(1, \lambda_{\mathrm{m}}^{\mathrm{s}})$-ES and shown in **Fig. 1**. In order to profit most profoundly from the interplay of mirrored sampling and sequential selection—namely from the increased likelihood that the mirrored solution is good, if the unmirrored solution was poor—we intertwine newly sampled solutions and their mirrored versions, i.e., we evaluate the offspring in the order $\boldsymbol{X}_k + \boldsymbol{\mathcal{N}}_k^1$, $\boldsymbol{X}_k - \boldsymbol{\mathcal{N}}_k^1$, $\boldsymbol{X}_k + \boldsymbol{\mathcal{N}}_k^3$, $\boldsymbol{X}_k - \boldsymbol{\mathcal{N}}_k^3$, ...

[2] However, the iteration counters differ and other parts of the algorithm might essentially depend on λ or the iteration counter.

3 Convergence Rates on the Sphere and Lower Bounds

In this section, we investigate theoretically the gain we can expect from mirrored samples and sequential selection on spherical functions. We are interested in convergence rates for isotropic $(1, \lambda)$-ESs with adaptive step-size where an offspring i at iteration k equals $\boldsymbol{X}_k + \sigma_k \boldsymbol{\mathcal{N}}^i$ with $\sigma_k > 0$ being the step-size. Here, $(\boldsymbol{\mathcal{N}}^i)_{1 \leq i \leq \lambda}$ will denote i.i.d. random vectors following a multivariate normal distribution with identity covariance matrix. Though (independently) sampled anew each iteration, we drop the dependency on k in the notation.

The dynamics and thus the convergence rate of a step-size adaptive ES obviously depends on the step-size rule. We will study here an (artificial) step-size setting that we call *scale-invariant step-size*, where σ_k is proportional to the distance to the optimum assumed w.l.o.g. in 0, that is $\sigma_k = \sigma \|\boldsymbol{X}_k\|$ for $\sigma > 0$. We will also explain how convergence rates with scale-invariant step-size on spherical functions relate to optimal bounds for convergence rates of general adaptive step-size ESs.

Preliminaries. The fastest convergence that can be achieved by step-size adaptive ESs is linear convergence, where the logarithm of the distance to the optimum decreases to $-\infty$ linearly like the number of function evaluations increases [3, ref. [13]]. An example of linear convergence is illustrated in **Fig. 2** for three different instances of the (1,2)- and (1,2$_\mathrm{m}$)-ESs. We now establish a formal definition of linear convergence taking into account that different numbers of evaluations are performed per iteration. Let T_k be the number of function evaluations performed until iteration k. Almost sure (a.s.) linear convergence takes place if there exists a constant $c \neq 0$, such that

$$\frac{1}{T_k} \ln \frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} \to c \ \ a.s.[3] \tag{1}$$

The convergence rate c is the slope of the curves in **Fig. 2**. The $(1 \overset{+}{,} \lambda)$- and $(1 \overset{+}{,} \lambda_\mathrm{m})$-ES perform λ evaluations per iteration and therefore $T_k = \lambda k$. In the sequel $\mathcal{M}$ denotes the set of functions $g : \mathbb{R} \mapsto \mathbb{R}$ that are strictly increasing.

How do we prove linear convergence for scale-invariant step-size? We explain now the main idea behind the proofs that we cannot present in detail due to space limitations but which can be found in [3]. Assume that the number of offspring per iteration is fixed to λ such that $T_k = \lambda k$. The first step of the proofs expresses the left-hand side (LHS) of (1) as a sum of k terms exploiting standard properties of the logarithm function:

$$\frac{1}{\lambda}\frac{1}{k} \ln \frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} = \frac{1}{\lambda}\frac{1}{k} \sum_{i=0}^{k-1} \ln \frac{\|\boldsymbol{X}_{i+1}\|}{\|\boldsymbol{X}_i\|} \quad . \tag{2}$$

We then exploit the isotropy of the sphere function, the isotropy of the multivariate normal distribution and the scale-invariant step-size rule to prove that all terms $\ln(\|\boldsymbol{X}_{i+1}\|/\|\boldsymbol{X}_i\|)$ are independent and identically distributed. A law of large numbers (LLN)[4] therefore implies that the right-hand side (RHS) of (2) converges when

[3] Literally, *convergence* of $\boldsymbol{X}_k$ takes place only if $c < 0$.

[4] This also requires verifying some technical conditions, such that the expectation and the variance of $\ln(\|\boldsymbol{X}_{i+1}\|/\|\boldsymbol{X}_i\|)$ are finite.

k goes to infinity to $E[\ln(\|\boldsymbol{X}_{i+1}\|/\|\boldsymbol{X}_i\|)]$ almost surely. For more details see [3, ref. [13]].

Convergence rate for the $(1, \lambda)$***-ES.*** Linear convergence for the $(1, \lambda)$-ES with scale-invariant step-size has been shown for instance in [4]. We restate the result while denoting the first coordinate of a vector $\boldsymbol{Z}$ by $[\boldsymbol{Z}]_1$.

Theorem 1. *For a* $(1, \lambda)$*-ES with scale-invariant step-size (*$\sigma_k = \sigma\|\boldsymbol{X}_k\| > 0$*) on the class of spherical functions* $g(\|\boldsymbol{x}\|)$*,* $g \in \mathcal{M}$*, linear convergence holds with*

$$\frac{1}{\lambda}\frac{1}{k}\ln\frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} \xrightarrow[k\to\infty]{} \frac{1}{2}\frac{1}{\lambda}E\left[\ln\left(1+\sigma\min_{1\le i\le\lambda}\left(2[\boldsymbol{\mathcal{N}}^i]_1+\sigma\|\boldsymbol{\mathcal{N}}^i\|^2\right)\right)\right]\ a.s., \tag{3}$$

where $(\boldsymbol{\mathcal{N}}^i)_{1\le i\le\lambda}$ *are* λ *independent random vectors.*

The proof follows the sketch presented above. Exploiting the isotropy of the sphere and the scale-invariant step-size rule, we find that the random variable $\|\boldsymbol{X}_{i+1}\|^2/\|\boldsymbol{X}_i\|^2$, for all i, is distributed as the random variable $Z_{(1,\lambda)} = 1 + \sigma\min_{1\le i\le\lambda}(2[\boldsymbol{\mathcal{N}}^i]_1 + \sigma\|\boldsymbol{\mathcal{N}}^i\|^2)$. Applying the LLN to (2), we prove the linear convergence with convergence rate $\frac{1}{2}\frac{1}{\lambda}E[\ln(Z_{(1,\lambda)})]$.

Convergence rate for the $(1, \lambda_{\mathrm{m}})$***-ES.*** In a similar manner we derive the linear convergence for the $(1, \lambda)$-ES with mirrored samples.

Theorem 2. *For a* $(1, \lambda_{\mathrm{m}})$*-ES with even* λ *and scale-invariant step-size (*$\sigma_k = \sigma\|\boldsymbol{X}_k\| > 0$*) on the class of spherical functions* $g(\|x\|)$*, for* $g \in \mathcal{M}$*, linear convergence holds and*

$$\frac{1}{\lambda}\frac{1}{k}\ln\frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} \xrightarrow[k\to\infty]{} \frac{1}{2}\frac{1}{\lambda}E\left[\ln\left(1+\sigma\min_{1\le i\le\lambda/2}\left(-2|[\boldsymbol{\mathcal{N}}^i]_1|+\sigma\|\boldsymbol{\mathcal{N}}^i\|^2\right)\right)\right]\ a.s. \tag{4}$$

where $(\boldsymbol{\mathcal{N}}^i)_{1\le i\le\lambda/2}$ *are* $\lambda/2$ *independent random vectors.*

The difference to the previous proof lies in the expression of the random variable $\|\boldsymbol{X}_{i+1}\|^2/\|\boldsymbol{X}_i\|^2$ equal to $Z_{(1,\lambda_{\mathrm{m}})} = 1 + \sigma\min_{1\le i\le\lambda/2}\left(-2|[\boldsymbol{\mathcal{N}}^i]_1| + \sigma\|\boldsymbol{\mathcal{N}}^i\|^2\right)$ in distribution.

Convergence rate for the $(1, 2^{\mathrm{s}})$***-ES.*** To tackle the convergence of algorithms with sequential selection, we need to handle the fact that T_k, the number of offspring evaluated until iteration k, is a random variable, because the number of offspring per iteration is itself not a constant but a random variable in this case. This difficulty can be solved for λ even as we illustrate for $\lambda = 2$.

Theorem 3. *For a* $(1, 2^{\mathrm{s}})$*-ES with scale-invariant step-size (*$\sigma_k = \sigma\|\boldsymbol{X}_k\| > 0$*) on the class of spherical functions* $g(\|\boldsymbol{x}\|)$*, for* $g \in \mathcal{M}$*, linear convergence holds and*

$$\frac{1}{T_k}\ln\frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} \xrightarrow[k\to\infty]{} \frac{1}{2}\frac{E\left[\ln\left(1+\sigma\left(Y_1 1_{\{Y_1<0\}}+\min(Y_1,Y_2)1_{\{Y_1\ge0\}}\right)\right)\right]}{2-p_s(\sigma)}\ a.s. \tag{5}$$

where T_k *is the random variable for the number of function evaluations until iteration* k*,* $Y_1 = 2[\boldsymbol{\mathcal{N}}^1]_1 + \sigma\|\boldsymbol{\mathcal{N}}^1\|^2$*,* $Y_2 = 2[\boldsymbol{\mathcal{N}}^2]_1 + \sigma\|\boldsymbol{\mathcal{N}}^2\|^2$ *with* $\boldsymbol{\mathcal{N}}^1$*,* $\boldsymbol{\mathcal{N}}^2$ *being two independent random vectors and* $p_s(\sigma) = \Pr(2[\boldsymbol{\mathcal{N}}^1]_1 + \sigma\|\boldsymbol{\mathcal{N}}^1\|^2 < 0)$ *corresponds to the probability that the first offspring is better than its parent.*

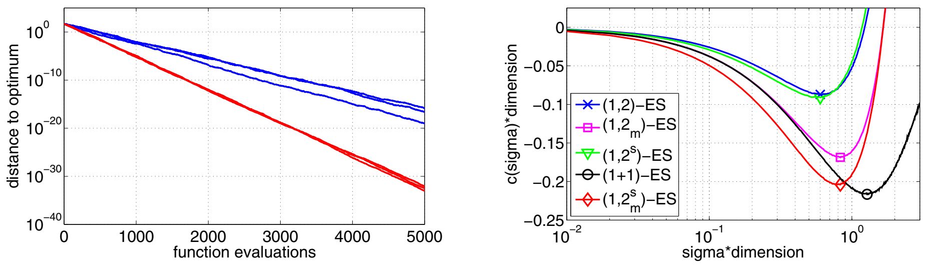


Fig. 2. Left: Evolution of distance to the optimum versus number of function evaluations for the (1,2)-ES (3 upper curves) and (1,2$_\mathrm{m}$)-ES (3 lower curves) with scale-invariant step-sizes ($d = 20$, $\sigma = 0.6/d$) on $f(\boldsymbol{x}) = \|\boldsymbol{x}\|^2$; **Right**: Convergence rate $c(\sigma)$ multiplied by the dimension d versus $\sigma \cdot d$ for different algorithms with scale-invariant step-size in dimension $d = 20$. The estimated best convergence rate for each algorithm is depicted by a marker

The first step of the proof expresses the LHS of (5) as $A_k = k/T_k$ times $B_k = \frac{1}{k}\ln(\|\boldsymbol{X}_k\|/\|\boldsymbol{X}_0\|)$. Then we handle both terms separately. For B_k, we proceed as before and obtain convergence towards $\frac{1}{2}E[\ln Z_{(1,2^\mathrm{s})}]$ with $Z_{(1,2^\mathrm{s})} = 1 + \sigma\left(Y_1 1_{\{Y_1<0\}} + \min(Y_1, Y_2)\ 1_{\{Y_1\geq 0\}}\right)$. For the term A_k, we denote by Λ_i the number of offspring evaluated at iteration i. Then, $T_k = \Lambda_1 + \ldots + \Lambda_k$ and $1/A_k = \frac{1}{k}\sum_{i=1}^k \Lambda_i$. Using the isotropy of the sphere function and the multivariate normal distribution and exploiting the scale-invariance of the step-size, we prove that Λ_i are identically distributed and independent. We can again apply the LLN and prove that $1/A_k$ converges almost surely to $E(\Lambda_1)$. Moreover, we prove that $E(\Lambda_1) = 2 - p_s(\sigma)$.

Convergence rate for the $(1, 2^\mathrm{s}_\mathrm{m})$***-ES.*** To establish the results for the (1,2)-ES with mirrored samples and sequential selection, we proceed exactly as in Theorem 3. Note that similar results can be derived for the (1,4)-ES with sequential selection [3].

Theorem 4. *For a $(1, 2^\mathrm{s}_\mathrm{m})$-ES with scale-invariant step-size ($\sigma_k = \sigma\|\boldsymbol{X}_k\| > 0$) on the sphere function $g(\|\boldsymbol{x}\|)$, for $g \in \mathcal{M}$, linear convergence holds and*

$$\frac{1}{T_k}\ln\frac{\|\boldsymbol{X}_k\|}{\|\boldsymbol{X}_0\|} \xrightarrow[k\to\infty]{} \frac{1}{2}\frac{1}{2-p_s(\sigma)} \times E\left[\ln\left(1 - 2\sigma|[\boldsymbol{\mathcal{N}}]_1| + \sigma^2\|\boldsymbol{\mathcal{N}}\|^2\right)\right]\ a.s. \quad (6)$$

where T_k is the random variable for the number of function evaluations until iteration k, $\boldsymbol{\mathcal{N}}$ is a random vector following a multivariate normal distribution, and $p_s(\sigma) = \Pr(2[\boldsymbol{\mathcal{N}}]_1 + \sigma\|\boldsymbol{\mathcal{N}}\|^2 < 0)$ is the probability that the first offspring is successful.

Link between convergence rates on the sphere and lower bounds for convergence. The convergence rates in (3), (4), (5) and (6) depend on σ. The RHS of **Fig. 2** illustrates the dependence on σ for $\lambda = 2$. For the $(1, \lambda)$- and the $(1, \lambda_\mathrm{m})$-ES, the minimal values in σ of the RHS of (3) and (4) correspond to the fastest convergence rate that can be achieved on any function with any step-size adaptation technique. The proof is similar to the one presented in [3, ref. [13]] for the (1+1)-ES. For the $(1, \lambda^\mathrm{s})$-ES and $(1, \lambda^\mathrm{s}_\mathrm{m})$-ES, our result might be less general, but the minimal values in σ of the RHS of (5) and

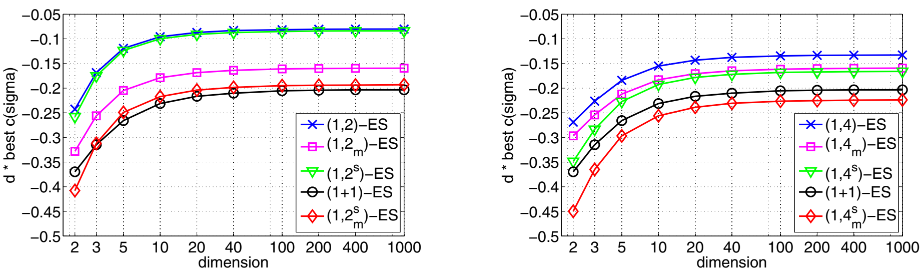


Fig. 3. Estimated optimal convergence rates on the sphere function for several algorithms with scale-invariant-constant step-size depending on the dimension d

(6) are at least the fastest convergence rates that can be achieved on spherical functions with any step-size adaptation technique. We refer to [3] for details of the proofs.

Numerical simulation of convergence rates. To evaluate the improvements that can be brought about by mirrored samples and sequential selection, we now compare the different convergence rates. However, those convergence rates are expressed only implicitly as the expectation of some random variables. We therefore simulate the convergence rate with a Monte-Carlo technique. For each convergence rate expression, we have simulated 10^6 times the random variables inside the expectation and averaged to obtain an estimate of the convergence rate for different σ. Here, σ has been chosen such that $0.01 \leq \sigma \cdot d \leq 3$ and with steps of 0.01 in $\sigma \cdot d$. The minimum of the measured convergence rates over $\sigma \cdot d$ is used as estimate of the *best* convergence rate for each algorithm and dimension—resulting in a slightly (systematically) smaller value than the true one, due to taking the minimal value from several random estimates. The right-hand plot of **Fig. 2** shows resulting convergence rate estimates versus σ in dimension 20. The step-sizes for the best measured convergence rates for the (1,2)-ESs are smaller than for the (1+1)-ES. The same is true for the (1,4)-ESs (not shown).

Fig. 3 presents the estimated best convergence rates for several algorithms for different dimensions. The strongest effect is observed from mirrored sampling in the (1,2)-ES. Only in dimension 2, the improvement is smaller than a factor of 1.5. Sequential selection alone offers little benefit for the (1,2)-ES, but the effect from mirrored sampling and sequential selection is clearly overadditive and the (1,2^s_m)-ES almost achieves the progress rate of the (1+1)-ES. In the (1,4)-ES, the impact of mirrored sampling or sequential selection is similar and less than a factor of 1.5. Their combined effect is close to additive and the (1,4^s_m)-ES becomes significantly faster than the (1+1)-ES.

4 Application to the CMA-ES Algorithm

We implemented *mirrored sampling* and *sequential selection* into the well-known *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES), where in addition to the step-size, the covariance matrix of the multivariate normal distribution is adapted [3, ref. [16,17,21,23]]. The additional implementational and numerical effort for the method is

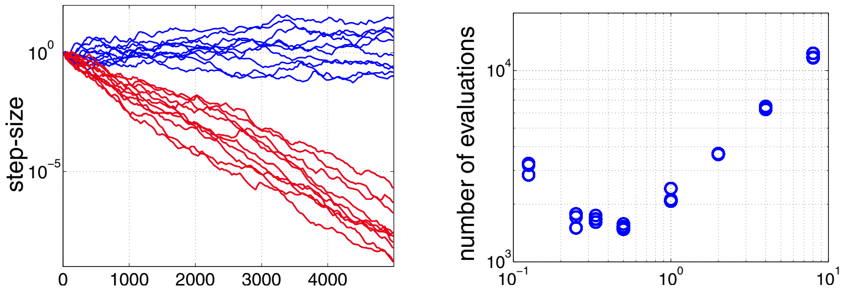


Fig. 4. Left: Step-size σ versus number of function evaluations of 20 runs on a purely random fitness function in dimension 10. The upper ten graphs show the $(5/5_{\mathrm{W}}, 10)$-CMA-ES revealing a random walk on $\log(\sigma)$. The lower ten graphs show the $(5/5_{\mathrm{W}}, 10_{\mathrm{m}})$-CMA-ES and reveal a strong bias of σ due to the recombination of mirrored vectors. **Right**: Number of function evaluations to reach function value 10^{-9} on the 20-D sphere function, versus multiplier of the default damping parameter d_σ for the $(1, 2^{\mathrm{s}}_{\mathrm{m}})$-CMA-ES starting from search point all-ones with $\sigma = 1$. Shown are three runs per d_σ-value. For smaller values of the multiplier the algorithm fails

negligible and even fewer random numbers need to be sampled with mirrored vectors. For parent number $\mu = 1$, the implementation is straightforward in both cases. Taking $\mu > 1$ with sequential selection, the decision for when to conclude the iteration is not entirely obvious and we stick to $\mu = 1$ for sequential selection.

Mirrored sampling with recombination. Taking $\mu > 1$ seems to have, a priori, no impact on the implementation of mirrored samples. Unfortunately, for $\mu > 1$, mirrored sampling introduces a strong bias on the step-size and the covariance matrix update in the $(\mu/\mu_{\mathrm{W}}, \lambda)$-CMA-ES under neutral selection (i.e., "pure random" selection). This effect is shown in **Fig. 4**, left. The bias is due to the recombination of mirrored offspring and systematically reduces the sampling variance. The bias can facilitate premature convergence for example in a noisy selection situation and is therefore considered as undesirable [6]. On the other hand, the bias can help to focus the convergence to a single optimum in a multi-modal or rugged search landscape. We have experimented with several ways to remove the bias, but leave the question of "which way is the best" open to future work. In the following also for mirrored sampling, only $\mu = 1$ is used.

Parameter setting. We modified the damping parameter for the **step-size** to $d_\sigma = 0.3 + 2\mu_{\mathrm{W}}/\lambda + c_\sigma$. Here, $1 \leq \mu_{\mathrm{W}} \leq \mu$ is the effective selection mass determined by the recombination weights and therefore $\mu_{\mathrm{W}} = \mu = 1$ in our case and usually $c_\sigma \ll 1$ [7]. For a given μ_{W}, the modification introduces a dependency of d_σ on λ. The setting was found by performing experiments on the sphere function, where the convergence rate is a unimodal function of d_σ. The default d_σ was chosen, such that in all cases (a) decreasing d_σ from the default value by a factor of two led to a better performance than increasing it by a factor of two, (b) decreasing d_σ by a factor of three never led to an observed failure (this is not always achieved for $\lambda = 2$ without mirroring), and (c) the performance with d_σ was at most two times slower than the optimal performance in the tuning graph. An example of a tuning graph for the $(1, 2^{\mathrm{s}}_{\mathrm{m}})$-CMA-ES is shown in **Fig. 4**, right. The graph meets the specifications (a)–(c), but ideally d_σ could have been

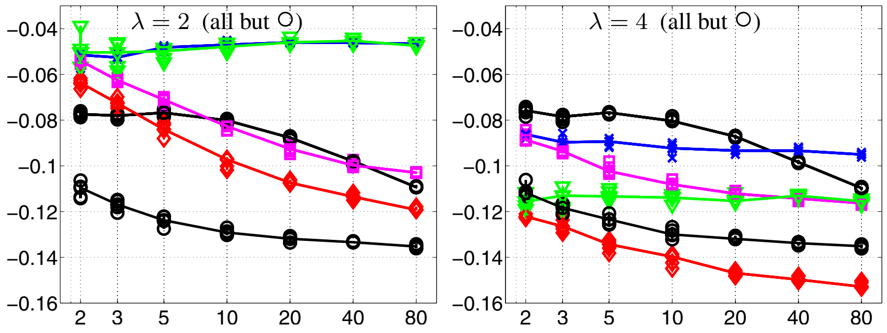


Fig. 5. Serial convergence rates $d \ln(\|\boldsymbol{X}_k\|/\|\boldsymbol{X}_0\|)/T_k$ versus dimension d of the CMA-ES on the sphere function with $\|\boldsymbol{X}_0\| = 1$, initial step-size $1/d$ and $\|\boldsymbol{X}_k\| \approx 10^{-150}$. ○: default (1+1)-CMA-ES (lower graph) and (6/6$_\mathrm{w}$,12)-CMA-ES; ×: (1,λ)-CMA-ES; ▽: sequential (1,λ^s)-CMA-ES; □: mirrored (1,λ_m)-CMA-ES; ◇: mirrored and sequential (1,$\lambda^\mathrm{s}_\mathrm{m}$)-CMA-ES. For each setting, five runs are shown and lines connect the median. Lower values are better

chosen almost two times smaller in this case. For λ as large as 1000 and dimension up to 5, even smaller values for d_σ are useful, but not exploited in the given default value.

For $\mu_\mathrm{w}/\lambda = 0.35$ and $\mu_\mathrm{w} \leq d + 2$, where d is the dimension, the former default setting of d_σ is recovered. For a smaller ratio of μ_w/λ or for $\mu_\mathrm{w} > d+2$, the new setting allows faster changes of σ and might be harmful in a noisy or too rugged landscape. In order to prevent a detrimental increment of the step-size for very large values of μ_w, the step-size multiplier is clamped from above at $\exp(1)$.

The learning rate for the **covariance matrix** in the CMA was originally designed for values of $\lambda \geq 5$. We rectified the learning rate of the rank-one update for small values of λ: the multiplier 2 is replaced by $\min(2, \lambda/3)$, resulting in $c_1 = \min(2, \lambda/3)/((d + 1.3)^2 + \mu_\mathrm{w})$. Similar as for the damping factor d_σ, the new value was guided by the specifications (a)–(c) from above when replacing d_σ with $1/c_1$ and optimizing the sphere function with a non-spherical initial covariance matrix and (d) the condition number of the final covariance matrix is smaller than ten. The learning rate for the rank-μ update of the covariance matrix is unchanged and zero for $\mu = 1$ [3, ref. [17,20]].

Convergence speed on the sphere. Similar to **Fig. 3**, we show in **Fig. 5** the convergence speed of various CMA-ES variants on the sphere function. We used `cmaes.m`, version `3.41.beta`, from `http://www.lri.fr/~hansen/cmaes_inmatlab.html` for implementing mirrored sampling and sequential selection. The resulting code is available at `http://coco.gforge.inria.fr/doku.php?id=bbob-2010-results`. In **Fig. 3**, the variance of the sample distribution was chosen optimal. In the CMA-ES, the covariance matrix is adapted and either *cumulative step-size adaptation* or the *1/5th success rule* is used for step-size control, in the non-elitist and the elitist variant respectively. While the overall convergence speed in moderate or large dimension is roughly two times slower than in **Fig. 3**, the ordering of the different variants essentially remains the same. The new sampling and selection schemes lead to a significant speedup. In low dimension, the convergence rate remains far from optimal, in accordance with observations in [2].

Experiments with BBOB-2010. The (1,2)- and the (1,4)-CMA-ES with mirrored sampling and/or sequential selection have been extensively empirically studied on 54 noisy [9] and noiseless [8] functions in the companion papers [3, ref. [3–10]]. Mirrored sampling improves the performance (number of function evaluations to reach a target function value) consistently on many functions by about a factor of two in the (1,2)-CMA-ES and by a much smaller but non-negligible factor in the (1,4)-CMA-ES. The larger factor for $\lambda = 2$ mainly reflects the comparatively poor performance of the baseline (1,2)-selection. On the attractive sector function f_6, the performance gain is more than a factor of three even for the (1,4)-CMA-ES in dimension 20. Additional sequential selection improves the performance again on many functions, typically by 10–30% for both values of λ. Even for the (1,4)-ES, the effect of mirrored sampling is still slightly more pronounced than that of sequential selection. Overall, the $(1, 4_{\mathrm{m}}^{\mathrm{s}})$-CMA-ES is consistently faster than the (1,$2_{\mathrm{m}}^{\mathrm{s}}$)-CMA-ES. On the noisy functions, the picture is qualitatively the same. Surprisingly, the differences are not less pronounced. Even sequential selection never impairs the performance significantly. In conclusion from this rather huge benchmarking exercise, the $(1, 4_{\mathrm{m}}^{\mathrm{s}})$-CMA-ES becomes the candidate of choice to replace the (1+1)-CMA-ES as *the* fast and robust local search ES.

Acknowledgments. This work receives support by the French national research agency (ANR) within the SYSCOMM project ANR-08-SYSC-017 and within the COSINUS project ANR-08-COSI-007-12.

References

1. Arnold, D.V., Salomon, R.: Evolutionary gradient search revisited. IEEE Transactions on Evolutionary Computation 11(4), 480–495 (2007)
2. Arnold, D.V., Van Wart, D.C.S.: Cumulative step length adaptation for evolution strategies using negative recombination weights. 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. 545–554. Springer, Heidelberg (2008)
3. Auger, A., Brockhoff, D., Hansen, N.: Mirrored sampling and sequential selection for evolution strategies. Research Report RR-7249, INRIA Saclay—Île-de-France (June 2010)
4. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: Keijzer, et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 445–452. ACM Press, New York (2006)
5. Auger, A., Hansen, N.: Benchmarking the (1+1)-CMA-ES on the BBOB-2009 noisy testbed. In: Rothlauf, F., et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009), Companion Material, pp. 2467–2472. ACM Press, New York (2009)
6. Hansen, N.: An analysis of mutative σ-self-adaptation on linear fitness functions. Evolutionary Computation 14(3), 255–275 (2006)
7. Hansen, N.: The CMA evolution strategy: a comparing review. In: Lozano, J., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms, pp. 75–102. Springer, Heidelberg (2006)
8. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA (2009), http://coco.gforge.inria.fr/bbob2010-downloads (updated February 2010)

9. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noisy functions definitions. Tech. Rep. RR-6869, INRIA (2009), http://coco.gforge.inria.fr/bbob2010-downloads (updated February 2010)
10. Igel, C., Suttorp, T., Hansen, N.: A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: Keijzer, et al. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 453–460. ACM Press, New York (2006)
11. Teytaud, O., Gelly, S., Mary, J.: On the ultimate convergence rates for isotropic algorithms and the best choices among various forms of isotropy. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 32–41. Springer, Heidelberg (2006)

Optimisation and Generalisation: Footprints in Instance Space

David W. Corne and Alan P. Reynolds

School of MACS, Heriot-Watt University, Edinburgh, UK
d.w.corne@hw.ac.uk, a.reynolds@hw.ac.uk

Abstract. The chief purpose of research in optimisation is to understand how to design (or choose) the most suitable algorithm for a given distribution of problem instances. Ideally, when an algorithm is developed for specific problems, the boundaries of its performance should be clear, and we expect estimates of reasonably good performance within and (at least modestly) outside its 'seen' instance distribution. However, we show that these ideals are highly over-optimistic, and suggest that standard algorithm-choice scenarios will rarely lead to the best algorithm for individual instances in the space of interest. We do this by examining algorithm 'footprints', indicating how performance generalises in instance space. We find much evidence that typical ways of choosing the 'best' algorithm, via tests over a distribution of instances, are seriously flawed. Also, understanding how footprints in instance spaces vary between algorithms and across instance space dimensions, may lead to a future platform for wiser algorithm-choice decisions.

1 Introduction

The chief purpose of research in optimisation is to understand how to find the most suitable algorithm for a given space of problem instances. When researchers concentrate on algorithm design without reference to specific problems, this view implies that the outcomes of such research should provide salient results on the applicability of the designed algorithms in terms of distributions of instances in one or more problem domains. Similarly, when research is clearly tied to a problem domain, the ideal outcomes would include a characterisation of the space of instances in that domain for which the studied algorithms are favoured.

We note that all of these desired outcomes are rare. However, such outcomes are extraordinarily important for two reasons: (a) it is easy to imagine practitioners choosing an algorithm for their problem on the basis of a research paper, and then applying it to real problems from a quite different instance distribution to that examined in the paper (indeed, often the paper's distribution is a small set of specific instances); (b) despite c. 50 years of research in optimisation, there is little re-usable knowledge that aids *a priori* choice/configuration of algorithms based on knowledge of the target distribution of instances.

Here we contribute to and explore the issues in (a) and (b) by examining algorithm 'footprints'. Such a 'footprint' indicates how an algorithm's comparative performance generalises in instance space (with reference to a collection of algorithms under consideration). In particular, we find much evidence to support the claim that the typical ways of choosing the 'best' algorithm, via tests over a distribution of instances, are seriously flawed: this may rarely yield an algorithm that is best on many well-defined subsets within that instance space, and similarly outside that instance space. Also, algorithms have footprints in instance space that define the regions of prowess of certain configurations. Understanding these footprints, how they vary between algorithms and across instance space dimensions, may lead to a future platform for wiser algorithm-choice decisions.

The remainder is set out as follows. After discussing related work and associated issues in Section 2, Section 3 simulates the task of choosing an algorithm for a given space of instances, for each of two problem domains. After establishing the 'best' algorithm, by sampling from an instance space, we then partition instance space into subspaces and do further experiments on samples from these subspaces, in turn summarising the results by showing that individual algorithms have performance 'footprints' over the instance subspaces. This reveals, for example, the extent to which the presumed 'best' algorithm is not best for most of the instance subspaces. We discuss the implications of this in discussions within section 3, and conclude in section 4.

2 Related Work and Associated Issues

There is a rich academic literature concerned with understanding how to discover the ideal algorithm for a given problem. This broad topic encompasses many inter-related questions. High-level such questions include how we are to define both 'best algorithm' and 'given problem'. In different circumstances, the best algorithm may be defined in terms of mean quality of solution, speed of convergence, or some function of these and/or other factors. The given problem may be a suite of test problems from one or more domains, or a stochastically defined space of instances.

Given some appropriate context that disambiguates these high level questions, most research in algorithm design and configuration focuses on algorithms for configuration search. In such work, it is common to use a 'training set' of instances, and treat algorithm configuration as an optimisation problem in which we seek to optimise a measure of performance over this set. In some cases, the optimally configured algorithm's performance is validated on a test set. However the latter is quite rare in many factions of the literature on optimisation algorithms, in which the typical approach is to test algorithms on a suite of instances commonly used in the literature, with little or no consideration given to estimating performance on different instances.

Recently prominent in algorithm configuration research are Hoos and co-workers [1,2,3], who have designed stochastic search algorithms such as paramILS [4], which is very successful at finding ideal configurations for a given algorithm and set of problem instances. ParamILS is one of several approaches that have been independently investigated for this task (e.g. [5,6,7,8]), and is distinguished from others by its attention to the problems of 'over-confidence' (performance on the training instances becoming an over-optimistic estimate of performance on unseen instances) and 'over-tuning' (the familiar machine learning scenario, in which prolonged training will lead to worse performance on the test set).

Essentially, such work (and similar, e.g. [9,10,11]) finds an algorithm that works best (given some definition) on a given instance distribution. Often this is a point distribution - a specific set of instances that may or may not be divided into training and test sets. It is worth noting that, though [4] deals with issues of over-confidence and over-tuning, such work rarely considers generalisation outside or within the instances considered. That is, the optimised configuration is delivered with (in [4], for example) some justified confidence that it is close to ideal for the training instances. However, it remains open how we might estimate its performance on alternative instances, or indeed on strict subsets of the training instance space.

To gain intuition for the significance of strict subsets of the instance space, consider configuring an algorithm for a job shop based on a given distribution of instances. By either using these instances directly, or constructing a generator based on them (e.g. with a distribution of processing times inferred from the instances), tuning will typically be biased towards some specific algorithm B, with an ideal mean performance over this instance space. However, there is an arbitrarily large number of coherent subsets of this space (defined by, for example, a given mean task length, or a given value for the range of processing times, and so forth) on which B's performance may well by trumped by an alternative configuration. The extent to which B may under-perform on new instances could be dramatic. When generalising outside the instance distribution, standard lessons from machine learning lead us to expect that, the better the performance of B on the test set, the worse its performance may be on unseen out-of-distribution instances. Although Hutter et al [4] avoid over-fitting in appropriate versions of ParamILS, this applies to the given instance distribution, and provides no certificates about performance outside this distribution.

Another area of related work is 'per-instance tuning', in which models are developed which enable an algorithm to be tuned based on features of the given instance [12–16]. In such work, typically preliminary experiments are done to capture the performance of several configurations of an algorithm on several instances. A model (e.g. perhaps a neural network) is then inferred, which predicts performance characteristics from combined instance features and algorithm parameters. Given a new instance, this model can then be used to predict the performance of any given configuration, and indeed a simple search can be wrapped around this, yielding predictions of ideal parameters for that instance. Smith-Miles et al [17] explore a variant of such ideas, concerned with algorithm choice based on instance features. In [17], 75,000 instances of a single-machine earliness/tardiness problem are generated, and on each instance they compare the earliest-due-date heuristic (EDD) and the shortest-processing time heuristic (SPT). A selection of learning methods are then applied, to infer models that relate instance features (e.g. mean due date, range of processing times, etc.) to the choice of either EDD or SPT. Results were very promising; some rules inferred from a training set of instances could predict with c. 97% accuracy the best choice between these two heuristics on a test set. In turn, this gives promises for better-informed choice of algorithm than may be obtained, say, by resorting to the single algorithm that was best over a wide distribution of instances that included the instance in question.

Our arguments and contributions are complementary to such findings from the work so far on the per-instance tuning approach. The latter studies typically reveal that small differences between instances correspond to differences in the ideal algorithm for those instances. By arguing (and confirming in test domains) that an algorithm tuned over a space of instances may rarely be the ideal algorithm for subsets of that space (let alone outside that space), we deliver similar findings from an alternative perspective. However our emphasis is different, focusing on the particular subset of instances that is best addressed by a given algorithm. We argue that the nature of this 'footprint' is particularly interesting to study, and understanding how the footprints of individual algorithms vary (e.g. simulated annealing versus population-based search) may lead to new ways to choose the right algorithm for a given problem domain, especially where the model development costs of accurate per-instance tuning (for example) are infeasible due to problem size and associated concerns regarding computational expense.

3 Empirical Examples

3.1 Overview of Experiments

In the sequel, empirical footprints in instance spaces are produced for a selection of algorithms and two domains. The domains are single-machine job shop (SMT; optimising tardiness) and vehicle-routing (VRP; optimising a weighted combination of distance, tardiness and overtime). These were chosen partly for convenience (the SMT is simply implemented and evaluation of solutions is computationally light) and partly for relevance: both are very common in industry, and are such that individual sources of instances (e.g. factories, warehouses, distributors) can be expected to draw their real-world instances from individually characteristic instance spaces. Further domains (e.g. the quadratic assignement problem) are under study, but space restraints here limit us to two.

Our experiments had the following overall shape. A set A of algorithm variants is defined, and a distribution $p(I)$ of instances is defined over a space of instances I. Simulating a simple algorithm-choice scenario, each algorithm in A is applied (with n independent trials) to a sample from $p(I)$, and we record the algorithm A_{best} that achieves the best overall performance. To measure performance we consider both the fitness values reached after the maximum allowed time (a given number of fitness evaluation), and the time (evaluation number) in each trial when the best fitness of that trial was found. The details of defining 'best' are as follows: given the $|A| \times n$ results for each sample instance, we consider all pairs of results from distinct algorithms in distinct trials on the same instance; in each such pair, the algorithm with the better fitness gains a point, with ties broken by speed of finding the solution. These points are accumulated over several instances. The algorithm with most points over a set of instances is considered best for that set, and the significance of its peak position (assessed by randomisation testing [18]) is considered in context. We found this more reliably inforative than simply resorting to mean fitness values per algorithm (which were not sfficiently distinguishing of algorithms on several of the smaller SMT instances).

We presume that A_{best} would be the algorithm delivered to the 'client', to be used to solve future real-world instances. Each of these phases of experimentation (to find A_{best} on a master instance space) typically involved from 4 to 400 billion evaluations: applying c. 200 algorithms to c. 1000 instances, for (usually) 20 trials of between 1,000 and 100,000 solution evaluations each.

Following this, we define several instance subspaces by partitioning $p(I)$ (details later), and we find the A^i_{best} for each subspace i; we then visualise the footprints of chosen algorithms - in particular we view the footprint of A_{best}, typically observing that it covers few of the instance subspaces. Further, when we explore subspaces, we include several outside the master space; this allows us to observe how A_{best} generalises beyond its deemed area of prowess.

The footprint graphics are obtained as follows. Each instance subspace is represented by a specific square. A given algorithm variant is identified by circles of a given shade. E.g. A_{best} (on the master space in context) is always black. If a square contains a shaded circle, then the corresponding algorithm was best in the corresponding subspace. The size of the circle is a measure of signifiance. Following the 'instance subspace' experiments, 1000 randomisation tests are done, in each of which the (best-fitness, evaluation number) pairs for each trial are randomly permuted within the results for the same instance. The points-assignment system (given above) is repeated each time, and this enables us to find (i) the signifiance of any given algorithm with at least the same number of points as the 'best' on that subspace, and (ii) the significance of the difference in performance between the best and second-best algorithms. The first signifiance value was always 1 (100%) in all of our experiments, confirming that the 'all algorithms were equivalent' hypothesis is easily rejected. The second signifiance value varied widely, indicating a 'distance' in performance between the best and second-best algorithms. The size of the circle relates to this second signifiance measure. Broadly speaking, large indicates that the algorithm was much better than the second-best, and that the difference is likely to be significant. A small circle means there was not a very distinct difference between the best and second best. Our code (for all experiments, for the randomisations, as well as producing the footprint graphics) can be obtained from http://macs.hw.ac.uk/~dwcorne/pubs.htm

3.2 Footprints in SMT Instance Space

First, experiments were performed to explore the performance of a collection of algorithm parameterisations on a distribution of instances of the single-machine tardiness (SMT) problem. In the SMT variant used here, T tasks, $t_1, ..., t_T$ need to be scheduled on a single machine, each task t_i having a processing time p_i and a due date d_i. A candidate solution is a permutation of the tasks, defining an order of processing on the machine. Given that order of processing (and with setup times assumed to be zero) each task has a finish time, and an individual task's tardiness is L_i, which is the maximum of $\{0, T_i - d_i\}$. Fitness (to be minimised) is simply the sum of the L_i.

The first 'master' instance distribution was defined by setting the number of tasks to be fixed at 20, and setting uniform ranges of processing times and due dates from the (integer) intervals [10, 40] and [10, 550] respectively. In the second phase, 40 separate subsets of instance distributions were used. Each subset was defined by a (processing time, due date) pair (p, d), and instances were produced by generating 50% of the tasks with processing times uniformly generated from a sub-range of the processing time distribution - the interval $[p, p+10]$ - and with due dates similarly generated from the interval $[d, d+60]$. Further details are in the caption of Figure 1. In this way, each instance generated within an instance subset is 'included' in the master instance space, with a nontrivial chance of being generated in the master instance distribution, however each such subspace is also clearly a well defined subset. Moreover, such a clustered nature in the due date and processing time distributions is consistent with what we may expect for sets of orders that many machine shops need to handle. E.g. clients tend to prefer deliveries at certain times, or certain products or processes may be particularly popular in certain periods.

The set of algorithms tested comprised 204 variants of hillclimbing (HC) and an evolutionary algorithm (EA). The 204 variants emerged from the rates of four distinct operators: adjacent-swap mutation, any-swap mutation, shift-2 mutation, and shift-3 mutation. The first two need no explanation; shift-n mutation means choosing a gene position i uniformly at random, and moving it to $i + n$ (treating the permutation as circular), while decrementing the positions of genes previously in positions $i + 1, ..., i + n$. An algorithm configuration was a vector of rates for the first three operators (the probability of the fourth being 1 minus their sum), ranging through all subsets of $\{0.1, 0.2, ..., 0.7\}^3$ whose components summed to 0.9 or below. The second set of algorithms were simple steady state EAs using binary tournament selection, with population sizes in {10, 20, 30}, and operators defined and allowed to vary precisely as described for HC.

Two SMT master instance spaces are considered here, corresponding to 20-task and 30-task problems. For each instance space, we consider footprints at different evaluation limits (1,000, 10,000 and 100,000). The resulting footprints are in Figure 1. The six footprint graphics each show footprints for a selection of five algorithm variants (each with its own shade of grey). In every case, 'black' denotes the algorithm that earned A_{best} on the master instance space. The four other algorithms are those deemed best most frequently over the subspaces. Where no circle appears, this means that the best algorithm for that subspace was best in few (if any) more subspaces It turns out that HC variants were always better than EA variants in the SMT cases studied here; the best algorithm in each subspace was an HC variant.

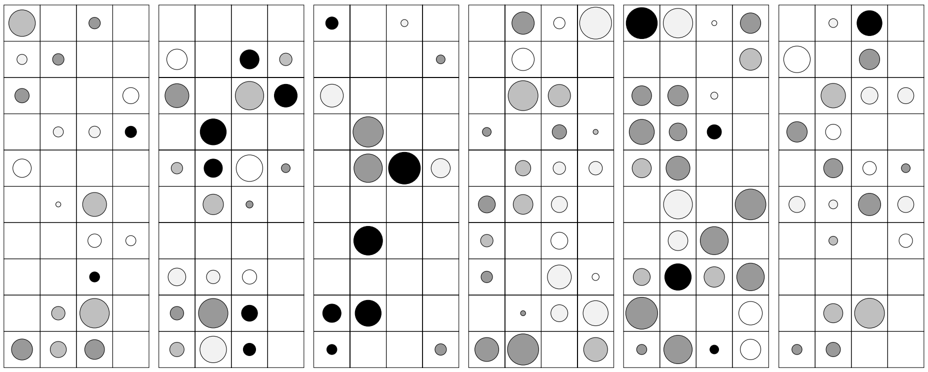


Fig. 1. Six groups of footprints for selected algorithms in SMT instance space. The leftmost (rightmost) three relate to 20 (30)-task problems, and from left to right in each group of three, the footprints relate to max evaluation limits of 1,000, 10,000, and 100,000. Each square indicates an instance subspace, in which the foci for task processing times are, from left to right, [10, 20], [20, 30], [30, 40] and [40, 50], and the due date focus for 20-task problems varies from top to bottom from [10, 70], [70, 130], ..., [550, 610]. For 30-task problems the due dates start at [10,100] and end with [820-910]. The rightmost columns and lowest rows always represent subspaces outwith the master instance space. Black represents A_{best}. Four other footprints are shown in each of the six groups, for the four algorithms that (neglecting the 'black' one) 'won' most frequently among the 40 instance subsets.

Figure 1 reveals a number of interesting points. The 'black' algorithm (always A_{best} for the master space in context) is rarely the best on the subspaces. In practice this suggests that, when an algorithm is tuned on the basis of a master instance space (or perhaps a set of specific instances), the delivered algorithm may rarely be fit for purpose. Footprints are generally patchy, however there are hints that algorithms have 'regions' of prowess. Considering the best in each subspace (including the infrequently winning algorithms in empty cells - whose display in further alternative shades of grey would confuse the visualisation), we explored the level of regionalisation by calculating the degree to which neighbouring (Von Neumann neighbourhood) subspaces shared the same best algorithm. In all cases in Figure 1 this was significant, as evaluated by randomisation tests that permuted 'bests' randomly within the subspaces. Confidence levels were (respectively for the six cases in Figure 1 in left to right order), 98.9%, 90.8%, 99.5%, 98.8%, 99.6 99.6%. The hints of regionalisation in the figures remain clear in the footprints not displayed - although such algorithms tended to win infrequently, their regions of high performance tended to be in the same neighbourhood. Results for subspaces in the rightmost columns and lowest rows speak to generalisation of the 'black' algorithm slightly beyond master instance space. Clearly, the performance of A_{best} beyond the master space is no better than its surprisingly poor showing in subspaces of the master space.

3.3 Footprints in VRP Instance Space

We considered a space of vehicle routing problems (VRP) in which up to three vehicle handle a total of 30 orders, each with a given 60 unit time window. We invented 50 customer locations (and one depot location) uniformly at random from $\{0, 1, 2, ..., 100\}^2$, and a random selection of 10 customers were stamped as 'regulars'. Distances were asymmetric (reflecting some amount of one-way systems and related factors), and generated by adding a uniform perturbation from $[-p, p]$, where p was 20% of the Manhattan distance.

An instance comprised 30 orders, each defined by a customer and a time window. Orders were generated first with a 50% chance of drawing the customer uniformly from regulars, otherwise drawn uniformly from all customers. Then, one of five possible time windows (the first five indicated in the caption of Figure 2) was chosen uniformly at random. A candidate solution was a permutation of 32 numbers, alleles 0–29 indexing the orders, while 30 and 31 were 'dividers', separating vehicles. Reading left to right, from one to three (depending on the divider positions) vehicle routes are calculated in the natural way. Fitness combined distance, lateness and idle time. Full and clear details may be found in the available code.

Instance subspaces were defined by a pair of time windows in which orders would be focussed (e.g. reflecting plausible business practice for many clients). Each subspace corresponded to a pair of the 7 windows listed in Figure 2's caption, as follows: when a customer was (not) regular, there was a 40% chance of the time window being the first (second) of the pair; otherwise time windows were drawn uniformly from all windows.

Figure 2 shows footprints for the cases of maximum evaluations of 10,000 (left) and 20,000 (right). We note first that, in contrast to the SMT cases, the winning algorithms in all subspaces for the VRP were EA variants. Otherwise, much the same can be said as was said for the SMT, although the 'black' algorithm is more prominent in this case. However one clear difference is that fewer distinct algorithms are represented as winners of the subspaces - the mean number of subspaces per winning algorithm was generally c. 3 in the SMT footprints, but c. 4 for the VRP. Also, the appearance of these footprints suggests regionalisation, but in fact the level of neighbour similarity is insignificant in both cases (again deterined by randomisation tests). I.e. there is no more regionalisation than would be expected for a random permutation of the same distribution of winning algorithms within the subspaces. Another difference is that the circles are generally larger. This suggests that choosing an algorithm based on subspace performance in this case may generally be a confident choice, but one which may often generalise poorly, given the general lack of regional similarity.

The nature of the footprints we have observed is clearly a function of the chosen algorithm collection, the problem domains, the instance spaces, and the way we have partitioned instances into subspaces. Further research is needed to understand these dependencies. If, for example there tend to be high levels of regionality for certain algorithms in certain domains, this would allow some confidence for generalisation of performance outside the master space; in other cases, it may be that ideal algorithm choice is highly sensitive to the co-ordinates

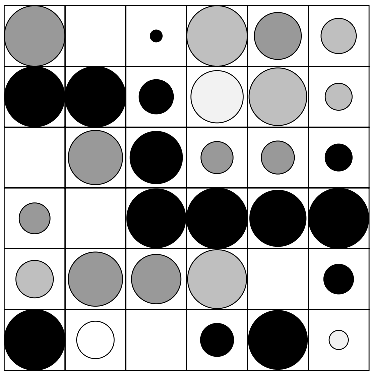
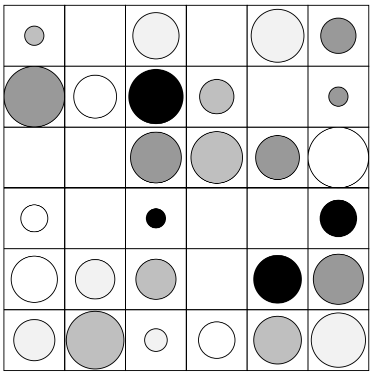



Fig. 2. Two groups of footprints for selected algorithms in a vehicle-routing instance space. Each group relates to a given max number of evaluations: 10,000 (left), 20,000 (right). Instance subspaces are defined by foci on regular and non-regular customer time windows (see text), from left to right (and top to bottom), [60, 120], [120, 180], [180, 240], [300, 360] and [360, 420],[420, 480]. The rightmost columns and lowest rows are outside the 'master' space. Black again represents A_{best} on the 'master' space. Four other footprints are shown, for the (other) algorithms that 'won' most frequently among the 36 subspaces.

of instance subspace - such a situation demands extra care in delivering the right algorithm, perhaps indicating a per-instance tuning approach. In particular, it could be that distinct tyes of algorithm have distinct footprint characteristics that tend to appear across domains. Especially when prior tuning approaches are costly, such characteristics may be used to inform algorithm choice.

Meanwhlle, the footprints we have examined here clearly challenge certain prior expectations of the robustness of a standard algorithm tuning approach. Our experiments arise from essentially arbitrary but representative and relevant choices of algorithms, domains and instance distributions, and we see similar findings in other domains (work in progress), so we do not expect these to be pathologic cases. Finally, however, we do not claim that the footprints captured herein have clear statistical signifiance in context, but we appeal to their being indicative of the general nature of algorithm footprints in optimisation.

4 Summary and Conclusion

We examined algorithm 'footprints' in the domains SMT and VRP. The footprint of an algorithm indicates how its performance generalises across different dimensions of instance space. In particular, we have found much evidence in support of the claim that the typical way of choosing the 'best' algorithm, via tests over a distribution of instances, is seriously flawed. The algorithm best overall on a broadly defined collection of instances may rarely be best on many well-defined subsets within that instance space, and similarly outside that instance space. The results also hint at potentially systematic differences between footprints, depending on algorithm family and domain. Such differences, given further understanding in future, may be usefully informative as regards algorithm-choice decisions in many scenarios. In particular, if the natures of footprints tend to generalise well across domains, footprint-oriented algorithm choice may be informative without the need for time-consuming development work that may otherwise be needed.

Acknowledgments. We are grateful for insightful anonymous reviews.

References

1. Hoos, H.H., Stutzle, T.: Stochastic Local Search Foundations & Applications. Morgan Kaufmann, San Francisco (2005)
2. Hutter, F., Hamadi, Y., Leyton-Brown, K., Hoos, H.H.: Performance prediction and automated tuning [...]. In: CP 2006, pp. 213–228 (2006)
3. Hutter, F., Hoos, H.H., Stutzle, T.: Automatic algorithm configuration based on local search. In: Proc. National Conf. on AI, vol. 2, pp. 1152–1157. MIT Press, Cambridge (2007)
4. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stutzle, T.: ParamILS: An Automatic Algorithm Configuration Framework. JAIR 36, 267–306 (2009)
5. Gratch, J., Chien, S.A.: Adaptive problem-solving for large-scale scheduling problems: A case study. JAIR 4, 365–396 (1996)
6. Minton, S.: Automatically configuring constraint satisfaction programs: A case study. Constraints 1(1), 1–40 (1996)
7. Johnson, D.S.: A theoretician guide to the experimental analysis of algorithms. In: Data Structures, Near Neighbor Searches and Methodology: Fifth and Sixth DIMACS Implementation Challenges, pp. 215–250. AMS, Providence (2002)
8. Birattari, M.: The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. PhD thesis, ULB, Belgium (2004)
9. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: IEEE CEC 2005, pp. 773–780 (2005)
10. Nannen, V., Eiben, A.E.: A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms. In: GECCO, pp. 183–190. ACM, New York (2006)
11. Nannen, V., Eiben, A.E.: Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In: IJCAI 1997, pp. 975–980 (1997)
12. Horvitz, E., Ruan, Y., Gomes, C.P., Kautz, H., Selman, B., Chickering, D.M.: A Bayesian approach [...]. In: UAI 2001, pp. 235–244. Morgan Kaufmann, San Francisco (2001)
13. Patterson, D.J., Kautz, H.: Auto-WalkSAT: a self-tuning implementation of WalkSAT. In: Electronic Notes in Discrete Mathematics (ENDM), vol. 9 (2001)
14. Carchrae, T., Beck, J.C.: Applying machine learning to low-knowledge control of optimization algorithms. Computational Intelligence 21(4), 372–387 (2005)
15. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for SAT. JAIR 32, 565–606 (2008)
16. Preuss, M.: Adaptability of ALgorithms for Real-Valued Parameter Optimization. In: Evoworkshops 2009. LNCS, pp. 665–674. Springer, Heidelberg (2009)
17. Smith-Miles, K., James, R.J., Giffin, J., Tu, Y.: Understanding the Relationship between (Structure and Performance). In: Proc. LION (2009)
18. Edgington, E.S.: Randomization Tests. Marcel Dekker AG, USA, 147 p. (1995)

Adaptive Drift Analysis*

Benjamin Doerr[1] and Leslie Ann Goldberg[2]

[1] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
[2] Department of Computer Science, University of Liverpool, Ashton Bldg, Liverpool L69 3BX, UK

Abstract. We show that the (1+1) evolutionary algorithm using an arbitrary mutation rate $p = c/n$, c a constant, finds the optimum of any n-bit pseudo-Boolean linear function f in expected time $\Theta(n \log n)$.

Since previous work shows that universal drift functions cannot exist for c larger than a certain constant, we define drift functions depending on p and f. This seems to be the first time in the theory of evolutionary algorithms that drift functions are used that take into account the particular problem instance.

1 Introduction

The introduction of drift analysis to the theory of evolutionary algorithms was a major breakthrough in this field [12,14]. Its most visible achievement is a simple and clean analysis of how the classical (1+1) evolutionary algorithm ((1+1) EA) optimizes linear (pseudo-Boolean) functions. This made obsolete the highly technical proof of Droste, Jansen and Wegener [6]. Other fundamental applications of drift analysis in the theory of evolutionary computation include [8,9,11,17,15].

Recent work by Johannsen, Winzen and the first author [5,4] shows that the drift analysis, as currently employed, strongly relies on the mutation probability being relatively small. The drift (=potential) function used in the first paper [12], as observed already in [13], is only admissible if the mutation probability p is less than $1/n$, where n is the length of the bit-strings of the search space.

This was fixed in [14], where a drift function was presented that works for the most common value of $p = 1/n$. As observed in [5], however, it ceases to work for p greater than or equal to $4/n$. This cannot be overcome by choosing a different drift function. As shown in [5], if $p > 4/n$, then for any drift function (from the class of log-of-linear functions) there is a linear objective function and a search point such that the drift from this point is negative. This problem does not change substantially if we use the averaging approach of Jägersküpper—that approach fails for $p \geq 7/n$ [4].

The results of [5,4] suggest that drift analysis, while successful for many particular problems, is still not that well-understood in the context of evolutionary

* This work was begun while both authors were visiting the "Centre de Recerca Matemática de Catalunya". It profited greatly from this ideal environment for collaboration.

algorithms and remains a topic asking for further research. For the particular test problem of analyzing linear functions, the results show that, unless completely different methods are invented, we cannot use a drift function that is defined independently of the objective function.

This seems to be a bad news, since typically it is quite tricky to come up with a feasible drift function. Currently, there seems to be little general advice of how to design such drift functions.

Nevertheless, we solve this problems in this paper by defining drift functions individually for each objective function and each mutation probability $p = c/n$. In consequence, we prove that the (1+1) EA with any such mutation probability optimizes any linear function in time $O(n \log n)$. A corresponding lower bound follows easily from standard arguments, see Section 6.

While we feel that the major contribution of this work is providing a way to overcome previous limitations of drift analysis, we should point out that also the result on the optimization time for linear functions a priori was neither known nor expected. Changing the mutation probability can make quite a difference. E.g., [3] showed that there are strictly monotone functions having an exponential optimization time for mutations probabilities $p > 33/n$, whereas all monotone functions are optimized in polynomial time for $p \leq 1/n$.

2 Analyzing the (1+1) EA via Drift Analysis

In this section, we introduce the (1+1) EA with general mutation probability p, describe drift analysis to the extent needed to analyze the optimization behavior of the (1+1) EA on linear functions, and discuss known and our result for linear functions. For other uses of drift analysis and the general background, we refer to the papers cited above.

2.1 The (1+1) Evolutionary Algorithm for Optimizing Functions Defined over Bitstrings

Let $f : \{0,1\}^n \to \mathbb{R}$ be a linear objective function. Without loss of generality, we may assume (and shall assume in this paper) that f is to be minimized, that $\mathbf{0} := (0, \ldots, 0)$ is the unique minimum, and that $f(\mathbf{0}) = 0$.

The randomized search heuristic that we study is the well-known *(1+1) evolutionary algorithm*. It starts with an initial solution, x, chosen uniformly at random from the *search space* $\{0,1\}^n$. In each iteration, from its existing solution x, it generates a new solution x' by flipping each bit of x with probability p independently. In other words, for each $i \in \{1, \ldots, n\}$ independently, we have $\Pr(x'_i = 1 - x_i) = p$ and $\Pr(x'_i = x_i) = 1 - p$. This step is called *mutation*. In the subsequent *selection* step, if $f(x') \leq f(x)$, the EA *accepts* x' as a solution, meaning that the next iteration starts with $x_{\text{new}} := x'$. Otherwise, the next iteration starts with $x_{\text{new}} := x$ unchanged. Since we are interested in analyzing how many iterations are necessary until an optimal solution is found, we do not specify a termination criterion here.

We should stress that the (1+1) EA typically is not used to actually solve difficult optimization problems. For these, one would instead choose more complex search heuristics. However, understanding the optimization behavior of the (1+1) EA often helps in predicting the behavior of other, complicated EAs, which mostly are too complex to admit a rigorous theoretical analysis.

2.2 Drift Analysis

The idea of drift analysis is to measure the progress of the optimization, not necessarily with respect to the fitness function, but with respect to a suitably chosen potential function (or artificial fitness function), which we shall call *drift function.* The definition below is adapted to our aim of analyzing linear objective functions.

Definition 1. *We call $\Phi : \{0,1\}^n \to \mathbb{R}$ a feasible drift function for f and the (1+1) EA with mutation probability p, if the following conditions are satisfied.*

1. $\Phi(\mathbf{0}) = 0$;
2. $\Phi(x) \geq 1$ *for all* $x \in \{0,1\}^n \setminus \{\mathbf{0}\}$;
3. *there is a constant* $\delta > 0$ *(independent of n) such that for all* $x \in \{0,1\}^n \setminus \{\mathbf{0}\}$,

$$E(\Phi(x_{\text{new}})) \leq (1 - \delta/n)\,\Phi(x),$$

where, as above, we denote by x_{new} the solution resulting from executing a single iteration (consisting of mutation and selection) with initial solution x.

As a semi-trivial example, note that if f is a linear function with coefficients at least one, then f itself is a feasible drift function for f and all mutation probabilities $p = c/n$. However, this often is not a very useful drift function.

When feasible drift functions exist, they allow an elegant analysis yielding upper bounds for the expected optimization time of EAs. The *optimization time* of a randomized search heuristic is usually defined to be the number of evaluations of the objective function f performed until the optimum is found. In case of the (1+1) EA, this is (apart from an additive deviation of one) equal to the number of mutation-selection iterations.

The following theorem, taken from [5], shows how the expected optimization time can be bounded using a drift function. Similar arguments appear in other contexts, see e.g. [10], [12] or, in the context of coupling proofs, [7, Section 5].

Theorem 1. *Let $\Phi : \{0,1\}^n \to \mathbb{R}$. Denote by $\Phi_{\max}$ the maximum value* $\max\{\Phi(x) \mid x \in \{0,1\}^n\}$ *of Φ. If Φ is a feasible drift function for f and the (1+1) EA with mutation probability p, then the expected optimization time of f is $O(n \log(\Phi_{\max} + 1))$.*

Theorem 1 indicates that a drift function is better if its maximum value, $\Phi_{\max}$, is small. For example, taking $\Phi = f$ only yields an expected optimization time of $O(n \log f_{\max})$. If f is a function like the binary value function BINVAL defined by $\text{BINVAL}(x) = \sum_{i=1}^{n} 2^{i-1} x_i$, we obtain a bound of only $O(n^2)$ for the expected optimization time.

2.3 Drift Analysis for Linear Functions

Finding feasible drift functions is typically quite tricky. If the mutation probability p is $1/n$, then He and Yao [14] showed that the drift function $\tilde{\Phi} : \{0,1\}^n \to \mathbb{R}$ defined by $\tilde{\Phi}(x) = \ln(1 + \sum_{i=1}^{\lfloor n/2 \rfloor} x_i + 2\sum_{i=\lfloor n/2 \rfloor+1}^{n} x_i)$ is a feasible drift function for *all* linear objective functions (in the additive setting they regard, that is, they have $E(\tilde{\Phi}(x_{\text{new}})) \leq \tilde{\Phi}(x) - \delta$ for all non-optimal x). With similar arguments, one can show that $\Phi(x) = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i + (5/4)\sum_{i=\lfloor n/2 \rfloor+1}^{n} x_i$ is a feasible drift function [5] as defined above, again valid for all linear functions f. Since $\Phi_{\max} = \Theta(n)$, we obtain an expected optimization time of $O(n \log n)$, which is asymptotically optimal [6]. Discovering such "universal" drift functions (applicable for *all* linear functions) is a beautiful and elegant way of solving the problem.[1]

Unfortunately, such universal drift functions only exist for small mutation probabilities p (cf. the introduction of this paper). In consequence, for larger values of p, the drift function has to depend on the particular objective function.

Non-trivial drift functions of this type were not known previously. Consequently, it was an open problem, prior to this work, to determine whether the $O(n \log n)$ time bound extends to mutation probabilities greater than $p = 4/n$. (We show that it does.) In this paper, we assume that the mutation probability is $p = c/n$, for an arbitrary constant c. This is appropriate because existing results of Droste, Jansen and Wegener [6, Theorems 13 and 14] show that this is the optimal order of magnitude for mutation probabilities.

In this paper, we solve the above-mentioned problems. For each p and f, we explicitly construct a feasible drift function Φ that is piecewise polynomially bounded. See Section 3.1 for a definition of this notion and the extension of Theorem 1 to such drift functions.

This result is interesting for two reasons. On the methodological side, it greatly enlarges our understanding of how good drift functions have to be chosen. This might help analyzing problems like the minimum spanning tree problem [16] or the single-criterion formulation of the single-source shortest path problem [1]. For both problems, the expected optimization time contains a $\log(f_{\max})$-factor stemming from the fact that, at least implicitly, drift analysis with the trivial drift function f is conducted.

On the other hand, naturally, our construction of drift functions proves that linear functions are optimized by the (1+1) EA in time $O(n \log n)$, regardless of what mutation probability $p = c/n$ is used.

Theorem 2. *Let c be an arbitrary constant. Then the (1+1) EA with mutation probability $p = c/n$ finds the optimum of any linear function $f : \{0,1\}^n \to \mathbb{R}$ in an expected number of $O(n \log n)$ iterations.*[2]

[1] We should add that a very similar function Φ was already used in the proof of [6]. However, without the use of drift analysis, the proof became highly technical.

[2] Without proof, let us remark that this bound also holds with high probability $1-n^{-d}$, where d is an arbitrary constant. This follows from [2].

3 Preliminaries and Notation

3.1 Piecewise Polynomially Bounded Drift and Fitness Based Partitions

Let f be an objective function to be minimized via the (1+1) EA as introduced in Subsection 2.1. We start with an elementary observation about the drift function Φ, namely that we do not really need $\Phi_{\max}$ to be polynomially bounded to obtain an $O(n \log n)$ bound on the expected optimization time. We can afford a constant number of 'huge jumps'. Suppose that $\mathcal{M} = M_0, \ldots, M_k$ is a partition of the search space $\{0,1\}^n$. Similar as in [18], we say that $\mathcal{M}$ is a *fitness-based partition* if $M_0 = \{\mathbf{0}\}$ and, for all $i < j$, $x \in M_i$ and $y \in M_j$, we have $f(x) < f(y)$. We use the notation $\min f(M_j)$ to denote $\min\{f(x) \mid x \in M_j\}$ and the notation $\max f(M_j)$ to denote $\max\{f(x) \mid x \in M_j\}$.

Lemma 1. *Let $\mathcal{M} = M_0, \ldots, M_k$ be a* fitness based partition *of the search space $\{0,1\}^n$, where k is a constant independent of n. Suppose that Φ is a feasible drift function for f. Then the expected optimization time of f is*

$$O\Bigg(n \sum_{j=1}^{k} \Big(\log(\max \Phi(M_j)) - \log(\min \Phi(M_j))\Big)\Bigg).$$

We say that Φ is *piecewise polynomially bounded* (with respect to f and the (1+1) EA), if there is a constant k (independent of n) and a fitness based partition $\mathcal{M} = M_0, \ldots, M_k$ of the search space such that for every $j \in \{1, \ldots, k\}$, $\log(\max \Phi(M_j)) - \log(\min \Phi(M_j)) = O(\log n)$.

3.2 Pseudo-boolean Linear Functions, Notation

We call a function $f : \{0,1\}^n \to \mathbb{R}$ a (pseudo-Boolean) *linear function*, if there are real numbers $a_0, a_1, \ldots, a_n$ such that $f(x) = a_0 + \sum_{i=1}^{n} a_i x_i$. It is relatively easy to see that, when discussing how such a function is optimized by the (1+1) EA, we may always assume that $a_0 = 0$ and that $a_{i+1} \geq a_i > 0$ for all $i \in \{1, \ldots, n-1\}$.

In the following, we shall denote an $x \in \{0,1\}^n$ by a binary string *written from right to left*, that is, $x = x_n \ldots x_1$ where x_n is the most-significant bit and x_1 is the least-significant bit. The reason for indexing the bits in this way is that, for the prominent example of the linear function $\textsc{binval}(x) = \sum_{i=1}^{n} 2^{i-1} x_i$, the binary string $x_n \ldots x_1$, interpreted in the usual way, is the value of the function. Note that it is important to remember this notation in the following, as we shall often use the word "left" to refer to the most-significant bit (with the largest index) and "right" to refer to the least-significant bit (with the smallest index).

4 Construction of the Drift Function

In this section, we show how to construct a feasible piecewise polynomially bounded drift function Φ. As discussed above, a feasible drift function Φ cannot be defined universally for all pseudo-Boolean linear functions f and all mutation probabilities p. Therefore, we have to define the drift function taking into account both f and $p = c/n$. Recall that we assume c to be constant.

Let ε be an arbitrary small constant, needed to precisely formulate the intermediate results. To define the drift function Φ, we will use a sufficiently large constant K (depending on c and ε) and a sufficiently small constant γ (depending on c, ε and K).

4.1 Splitting into Blocks

The difficulty in designing a feasible drift function lies in the fact that the optimization of f via the EA depends heavily on the coefficients a_i. If these coefficients are steeply increasing, then the optimization behavior resembles the behavior of the optimization function BINVAL defined by $\text{BINVAL}(x) = \sum_{i=1}^{n} 2^{i-1} x_i$. For this function, the left-most bit that is flipped decides whether the new solution is accepted. On the other hand, if the coefficients are of comparable sizes, then we see an optimization behavior resembling the behavior of the optimization function ONEMAX defined by $\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i$. For this function, it is instead the number of "good" bit-flips, relative to the number of "bad" flips that decides whether the new solution is accepted. Of course, what is "steeply increasing" and what constitutes "comparable sizes" depends on the mutation probability. Also, f can be of a mixed type, having regions with steeply increasing coefficients and regions with coefficients of comparable sizes.

To analyze such objective functions, we split the bit positions $\{1, \ldots, n\}$ into blocks. The idea is that, within such a block, one of the two behaviors is dominant. The definition of blocks, naturally, has to be such that the interaction between different block can be analyzed appropriately.

We first split the bit positions $\{1, \ldots, n\}$ into *mini-blocks* in the following manner. Start with $j = 1$. A mini-block starting at bit position j is constructed as follows. If $a_n/a_j < n^2$ then $\{j, \ldots, n\}$ is a single mini-block. Otherwise, let i be the minimum value in $\{j+1, \ldots, n\}$ such that $a_i/a_j \geq n^2$. The set $\{j, \ldots, i\}$ is a mini-block. If $i = n$, we are finished. Otherwise, set $j = i$ and repeat to form the next mini-block, starting at bit position j. Note that, in general, consecutive mini-blocks overlap by one bit position.

The next thing that we do is merge consecutive pairs of mini-blocks into *blocks.* To start out with, we just go through the mini-blocks from right to left, making a block out of each pair of mini-blocks. Note that this is (intentionally) different from just defining blocks analogous to mini-blocks with the n^2 replaced by n^4.

A block is said to be *long* if it contains at least γn bit positions and *short* otherwise. Recall from the beginning of this section that γ is a small constant depending on c, ε and K. It helps our analysis if any pair of long blocks has at least three short blocks in between. So if two long blocks are separated by at most two short blocks, then we call the union of these two long blocks and the at most two short blocks in between a long block. We will use ℓ_B to denote the leftmost bit position in block B and r_B to denote the rightmost bit position in block B. As long as B is not the left-most block, we have $a_{\ell_B}/a_{r_B} \geq n^4$.

4.2 Definition of Φ

We will define weights $w_1, \dots, w_n \in \mathbb{R}$ such that $\Phi(x) = \sum_{i=1}^{n} w_i x_i$. We call the w_i *weights* to distinguish them from the *coefficients* $a_1, \dots, a_n$ of f. We define the weights $w_1, \dots, w_n$ as follows, starting with $w_1 = 1$. Suppose that bit-position i is in block B (and that $i \neq r_B$). If block B is a long block, or is immediately to the left of a long block, then we define w_i by $w_i = w_{r_B} a_i / a_{r_B}$. We call this the "copy regime" since $w_i / w_{r_B} = a_i / a_{r_B}$. Otherwise, we are in the "damped regime" and we define w_i by $w_i = w_{r_B} \min(K^{(i-r_B)c/n}, a_i/a_{r_B})$. The reason that we define the weights in this way will be apparent in Section 5.1. The main constraints are that blocks had to be defined so that the fitness function increases sufficiently from block to block. The same has to be true of the weights, however, it has to remain piecewise linear. Also, the details of the damping ensure the necessary geometric decay of the weights that allow the analysis to go through. The main technical work in the analysis is showing that Φ is a feasible drift function. By contrast, the proof of the following lemma is straightforward.

Lemma 2. *Φ is piece-wise polynomial.*

5 Feasible Drift

The main technical work of the paper is to show that Φ, as just defined, is a feasible drift function for f. To do so, we use the following notation. The state after t steps is a binary string $x[t] = x_n[t] \dots x_1[t]$. Recall from Section 3.2 that we write bit-strings as $x_n, \dots, x_1$ where x_n is the leftmost bit. In the $(t+1)$'st step of the algorithm, the bits of a binary string $y[t+1] = y_n[t+1] \dots y_1[t+1]$ are chosen independently. The probability that $y_i[t+1] = 1$ is $p = c/n$ by definition of the (1+1) EA. Then $x'[t+1]$ is formed from $x[t]$ by flipping the bits that are 1 in string $y[t+1]$. That is, $x'_n[t+1] \dots x'_1[t+1] = (x_n[t] \oplus y_n[t+1]) \dots (x_1[t] \oplus y_1[t+1])$. Let A_{t+1} be the event that $\sum_i a_i x'_i[t+1] \leq \sum_i a_i x_i[t]$. We say that the move in step $t+1$ is "accepted" in this case. If A_{t+1} occurs then $x[t+1] = x'[t+1]$. Otherwise, $x[t+1] = x[t]$. To show that Φ, defined by $\Phi(x) = \sum_i w_i x_i$ for all $x \in \{0,1\}^n$, is a feasible drift function, we show the following.

Lemma 3. *For all $x \in \{0,1\}^n \setminus \{\mathbf{0}\}$,*

$$E[\Phi(x[t+1]) \mid x[t] = x] \leq \left(1 - \tfrac{1}{n} c e^{-3c}(1-\varepsilon)\right) \Phi(x).$$

Proof (sketch). Suppose that $x[t]$ is not the all-zero string. For a bit position i with $x_i[t] = 1$, let $I_i[t+1]$ be the event

$$y_i[t+1] = 1 \wedge \forall j \in \{i+1, \dots, n\} : y_j[t+1] = 0 \vee x_j[t] = 0.$$

$I_i[t+1]$ is the event that i is the left-most '1' to be considered for a flip in step $t+1$. Note that, for any fixed $x[t]$,

$$E[\Phi(x[t]) - \Phi(x[t+1])] = \sum_{i : x_i[t]=1} \Pr(I_i[t+1]) E[\Phi(x[t]) - \Phi(x[t+1]) \mid I_i[t+1]]$$

since the events $I_i[t+1]$ for $1 \leq i \leq n$ are disjoint and $\Phi(x[t]) = \Phi(x[t+1])$ unless one of them occurs.

We will show that $E[\Phi(x[t]) - \Phi(x[t+1]) \mid I_i[t+1]] \geq 0$ for all i with $x_i[t] = 1$. so we will be able to use the lower bound $\Pr(I_i[t+1]) \geq p(1-p)^n$ to get

$$E[\Phi(x[t]) - \Phi(x[t+1])] \geq p(1-p)^n \sum_{i:x_i[t]=1} E[\Phi(x[t]) - \Phi(x[t+1]) \mid I_i[t+1]]. \quad (1)$$

Let $I'_\ell[t+1]$ be the event

$$\forall j \in \{\ell+1, \dots, n\} : y_j[t+1] = 0 \lor x_j[t] = 1.$$

$I'_\ell[t+1]$ is the event that no '0' to the left of bit ℓ is considered for a flip in step $t+1$. Note that $\Pr(I'_\ell[t+1]) \geq (1-p)^n$ and that this event is independent of $I_i[t+1]$ for any i.

Fix any $\varepsilon > 0$. We shall split the analysis into several cases in Section 5.1 and show that, in each case,

$$E[\Phi(x[t]) - \Phi(x[t+1]) \mid I_i[t+1]] \geq (1-p)^{2n} w_i (1-\varepsilon).$$

Thus, by (1),

$$E[\Phi(x[t]) - \Phi(x[t+1])] \geq p(1-p)^n (1-p)^{2n} (1-\varepsilon) \Phi(x[t]),$$

so $E[\Phi(x[t+1])] \leq (1 - p(1-p)^{3n}(1-\varepsilon))\Phi(x[t])$.

Noting that $(1-p)^{3n} \geq e^{-3c}(1-\varepsilon)$ for n sufficiently large, this shows Lemma 3. Note that since ε can be chosen arbitrarily small, a factor of $(1-\varepsilon)^2 \leq (1-2\varepsilon)$ is as good as one of $(1-\varepsilon)$. □

5.1 The Case Analysis

Suppose that bit position i is contained in a block B, that block L is to the left of B and that block R is to the right of block B. We study the expected drift conditioned on $I_i[t+1]$, which is the event that bit-position i is the left-most '1' to be considered for a flip in step $t+1$. If the mutation is accepted, then the flip of bit position i turns a '1' into a '0', making the drift function go down. What we have to prove is that the drift function is not likely to go up much, due to other '0' bits turning into '1'. It is easy to see that the '0's to the left of block L are irrelevant—the construction of the blocks ensures that the fitness function increases sufficiently from block-to-block, so a mutation will not be accepted if one of these bits is flipped. Similar, the '0's to the right of block R are irrelevant—the construction of the blocks and the definition of the weights ensures that the weights in the drift function increase sufficiently from block-to-block, so the contribution of these bits is small relative to the contribution of bit i. What remains, then, is to show that the contribution to the drift from '0's in blocks L, B and R is negligible compared to the contribution from bit i

flipping from ‘1’ to ‘0’. This analysis is done in separate cases sketched below. The details are omitted for reasons of space.

Case 1: None of blocks L, B and R are long. In this case, blocks L and B are in the damped regime, and this ensures the sum of the weights in block L and to its right are not too large, so the contribution of bit i is sufficient to compensate for them.

Case 2: Block L is long, but bit position i is in the rightmost miniblock of block B. This is similar to Case 1.

Case 3: Block R is long (so blocks L and B are not). In this case, blocks B and R are in the copy regime. Using this fact, we can carefully analyze the changes to the drift function due to bits in blocks B and R that would arise as a result of an accepting move. The copy regime allows us to translate knowledge about the fact that a move is accepted — which tells us about the relative fitness-function coefficients in these two blocks — to facts about the relative weights in these two blocks.

Case 4: Block L is long, and bit position i is in the leftmost miniblock of block B. This is the most difficult part of the analysis, and requires combining the copy-regime analysis in block L with the damped-regime analysis in blocks B and R. The construction of the blocks, and the details of the damping are tuned precisely to make it work.

Case 5: Block B is long. Once again, we can combine the copy-regime analysis and the damped-regime analysis to derive the necessary inequalities on the weights in blocks S, L and B that are added and subtracted to the drift function due to an accepting move.

6 A Simple Lower Bound

The previous sections showed that the (1+1) EA finds the optimum of any linear function in expected time $O(n \log n)$, regardless of the mutation probability $p = c/n$, as long as c is a constant. The simple argument that each bit that is not ‘1’ in the initial solution has to be part of at least one mutation, shows that, with high probability, $O(n \log n)$ iterations are needed.

Theorem 3. *Let c be any constant. Let $\tilde{c} = \max\{1, c\}$. Let f be any linear function. The probability that the (1+1) EA with mutation probability $p = c/n$ finds the optimum of f within $(1/4\tilde{c})(n-1)\ln(n)$ steps, is at most $\exp(-n^{\Omega(1)})$.*

7 Conclusion

We analyzed the optimization time of the (1+1) EA for optimizing a linear function using an arbitrary mutation probability $p = c/n$, c constant. By defining a potential function depending on the linear objective function and the mutation probability p, we show that the expected optimization time is $O(n \log n)$, which is optimal.

References

1. Baswana, S., Biswas, S., Doerr, B., Friedrich, T., Kurur, P.P., Neumann, F.: Computing single source shortest paths using single-objective fitness. In: Proceedings of FOGA 2009, pp. 59–66. ACM, New York (2009)
2. Doerr, B., Goldberg, L.A.: Drift analysis with tail bounds. In: Parallel Problem Solving from Nature–PPSN XI. LNCS. Springer, Heidelberg (to appear 2010)
3. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Optimizing monotone functions can be difficult. In: Parallel Problem Solving from Nature–PPSN XI. LNCS. Springer, Heidelberg (to appear 2010)
4. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: Proceedings of CEC 2010. IEEE, Los Alamitos (to appear 2010)
5. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. In: Proceedings of GECCO 2010. ACM, New York (to appear 2010)
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
7. Dyer, M., Greenhill, C.: Random walks on combinatorial objects. In: Surveys in Combinatorics 1999, pp. 101–136. University Press (1999)
8. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
9. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimization. In: Proceedings of GECCO 2006, pp. 651–658. ACM, New York (2006)
10. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability 13, 502–525 (1982)
11. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Proceedings of GECCO 2008, pp. 953–960. ACM, New York (2008)
12. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127, 57–85 (2001)
13. He, J., Yao, X.: Erratum to: Drift analysis and average time complexity of evolutionary algorithms (artificial intelligence 127, 57-85 (2001)). Artificial Intelligence 140, 245–248 (2002)
14. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3, 21–35 (2004)
15. Neumann, F., Oliveto, P.S., Witt, C.: Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In: Proceedings of GECCO 2009, pp. 835–842. ACM, New York (2009)
16. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms and the minimum spanning tree problem. Theoretical Compututer Science 378, 32–40 (2007)
17. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 82–91. Springer, Heidelberg (2008)
18. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In: Sarker, R., Yao, X., Mohammadian, M. (eds.) Evolutionary Optimization, pp. 349–369. Kluwer, Dordrecht (2002)

Optimizing Monotone Functions Can Be Difficult

Benjamin Doerr[1], Thomas Jansen[2], Dirk Sudholt[3], Carola Winzen[1], and Christine Zarges[4]

[1] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
[2] University College Cork, Cork, Ireland
[3] International Computer Science Institute, Berkeley, CA 94704, USA
[4] Technische Universität Dortmund, 44221 Dortmund, Germany

Abstract. Extending previous analyses on function classes like linear functions, we analyze how the simple (1+1) evolutionary algorithm optimizes pseudo-Boolean functions that are strictly monotone. Contrary to what one would expect, not all of these functions are easy to optimize. The choice of the constant c in the mutation probability $p(n) = c/n$ can make a decisive difference.

We show that if $c < 1$, then the (1+1) EA finds the optimum of every such function in $\Theta(n \log n)$ iterations. For $c = 1$, we can still prove an upper bound of $O(n^{3/2})$. However, for $c > 33$, we present a strictly monotone function such that the (1+1) EA with overwhelming probability does not find the optimum within $2^{\Omega(n)}$ iterations. This is the first time that we observe that a constant factor change of the mutation probability changes the run-time by more than constant factors.

1 Introduction

Rigorously understanding how randomized search heuristics solve optimization problems and proving guarantees for their performance remains a challenging task. The current state of the art is that we can analyze some heuristics for simple problems. Nevertheless, this gave new insight, helped to get rid of wrong beliefs, and turned correct beliefs into proven facts.

For example, it was long believed that a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is easy to optimize if it is *unimodal*, that is, if each $x \in \{0,1\}^n$ that is not optimal has a Hamming neighbor y with $f(y) > f(x)$ [1]. Recall that y is called a Hamming neighbor of x if x and y differ in exactly one bit.

This belief was debunked in [2]. There the unimodal long k-path function [3] was considered and it was proven that the simple (1+1) evolutionary algorithm ((1+1) EA) with high probability does not find the optimum within $2^{\sqrt{n}}$ iterations. This classical episode shows how important it is to support an intuitive understanding of evolutionary algorithms with rigorous proofs.

It also shows that it is very difficult to identify problem classes that are easy for a particular randomized search heuristic. This, however, is needed for a successful application of such methods, because the no free lunch theorems [4] tell us, in simple words, that no randomized search heuristic can be superior to another if we do not restrict the problem class we are interested in.

1.1 Previous Work

In the following, we restrict ourselves to classes of simple pseudo-Boolean functions. We stress that the last ten years also produced a number of results on combinatorial problems [5]. At the same time research on classical test functions and function classes continued, spurred by the many still open problems.

We also restrict ourselves to one of the most simple randomized search heuristics, the (1+1) EA. The first rigorous results on this heuristic were given by Mühlenbein [1], who determined how long it takes to find the optimum of simple test functions like $\text{OneMax}(x) := \sum_{i=1}^{n} x_i$, counting the number of 1-bits. Quite some time later, and with much more technical effort necessary, Droste, Jansen and Wegener [6] extended the $O(n \log n)$ bound to all linear functions $f(x) := \sum_{i=1}^{n} a_i x_i$. Since it was hard to believe that such a simple result should have such a complicated proof, this work initiated a sequence of follow-up results in particular introducing drift analysis to the community [7,8] or refining it for our purposes [9,10,11]. However, not all promising looking function classes are easy to optimize. As laid out in the first paragraphs of this paper, already unimodal functions are difficult.

Almost all results described above were proven for the standard mutation probability $1/n$. It is easy to see from their proofs (or, in the case of linear functions, cf. [12]), that all results remain true for $p(n) = c/n$, where c can be an arbitrary constant.

We should add that the question how to determine the right mutation probability is also far from being settled. Most theory results for simplicity take the value $p(n) = 1/n$, but it is known that this is not always optimal [13]. In practical applications, similarly $1/n$ is the most recommended static choice for the mutation probability [14,15] in spite of known limitations of this choice [16].

1.2 Our Work

In this work, we regard the class of strictly monotone functions. A pseudo-Boolean function is called *strictly monotone* (or simply *monotone* in the following) if any mutation flipping at least one 0-bit into a 1-bit and no 1-bit into a 0-bit strictly increases the function value. Hence much stronger than for unimodal functions, we not only require that each non-optimal x has a Hamming neighbor with better f-value, but we even ask that this holds for all Hamming neighbors that have an additional 1-bit.

Obviously, the class of monotone functions includes linear functions with all bit weights positive. On the other hand, each monotone function is unimodal. Contrary to the long k-path function there is always a short path of at most n search points with increasing f-value connecting a search point to the optimum.

It is easy to see that monotone functions are just the ones where a simple coupon collector argument shows that random local search finds the optimum in time $O(n \log n)$. Surprisingly, we find that monotone functions are not easy to optimize for the (1+1) EA in general. Secondly, our results show that for this class of functions, the mutation probability $p(n) = c/n$, c a constant, can make a crucial difference.

More precisely, we show that for $c < 1$ the (1+1) EA with mutation probability c/n finds the optimum of any monotone function in time $\Theta(n \log n)$, which is best possible given previous results on linear functions. For $c = 1$, the drift argument breaks down and we have resort to an upper bound of $O(n^{3/2})$ based on a related model by Jansen [17]. We currently do not know what is the full truth. As lower bound, we only have the general lower bound $\Omega(n \log n)$ for all mutation-based evolutionary algorithms.

If c is sufficiently large, an unexpected change of regime happens. For $c > 33$, we show that there are monotone functions such that with overwhelming probability, the (1+1) EA does not find the optimum in exponential time. The construction of such functions heavily uses probabilistic methods. To the best of our knowledge, this is the first time that problem instances are constructed this way in the theory of evolutionary computation.

2 Preliminaries

We consider the maximization of a pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ by means of a simple evolutionary algorithm, the (1+1) EA. The results can easily be adapted for minimization. In this work, n always denotes the number of bits in the representation.

The (1+1) EA (described below) maintains a population of size 1. In each generation it creates a single offspring by independently flipping each bit in the current search point with a fixed mutation probability $p(n)$. The new search point replaces the old one in case its f-value is not worse.

(1+1) EA with mutation probability $p(n)$

```
1: Initializiation: Choose $x \in \{0,1\}^n$ uniformly at random.
2: repeat forever
3:     Create $y \in \{0,1\}^n$ by copying $x$.
4:     Mutation: Flip each bit in $y$ independently with probability $p(n)$.
5:     Selection: if $f(y) \geq f(x)$ then $x := y$.
```

In our analyses we denote by $\mathrm{mut}(x)$ the bit string that results from a mutation of x. We denote as x^+ the search point that results from a mutation of x and a subsequent selection. Formally, $y = \mathrm{mut}(x)$ and $x^+ = y$ if $f(y) \geq f(x)$ and $x^+ = x$ otherwise.

For $x = x_1 \cdots x_n$ let $Z(x)$ describe the positions of all 0-bits in x, $Z(x) := \{1 \leq i \leq n \mid x_i = 0\}$. By $|x|_0 = |Z(x)|$ we denote the number of 0-bits in x and by $|x|_1 = n - |x|_0$ we denote the number of 1-bits. For $k \in \mathbb{N}$ let $[k] := \{1, 2, \ldots, k\}$. For a set $I = \{i_1, i_2, \ldots, i_\ell\} \subseteq [n]$ we write $x_{|I} = x_{i_1} x_{i_2} \cdots x_{i_\ell}$ for the sub-string of x with the bits selected by I. To simplify notation we assume that any time we consider some $r \in \mathbb{R}_0^+$ but in fact need some $r' \in \mathbb{N}_0$ we assume that r is silently replaced by $\lfloor r \rfloor$ or $\lceil r \rceil$ as appropriate.

We are interested in the optimization time, defined as the number of mutations until a global optimum is found. For the (1+1) EA this is an accurate measure of the actual run time. For bounds on the optimization time we use common asymptotic notation. A run time bound is called *exponential* if it is $2^{\Omega(n)}$. We also say that an event A occurs *with overwhelming probability (w. o. p.)* if $1-\Pr(A) = 2^{-\Omega(n)}$.

A function f is called *linear* if it can be written as $f(x) := \sum_{i=1}^{n} a_i x_i$ for weights $a_1, \dots, a_n \in \mathbb{R}$. The most simple linear function is the function $\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i = |x|_1$. Another intensively studied linear function is $\textsc{BinVal}(x) := \sum_{i=1}^{n} 2^{n-i} x_i$. As $2^{n-i} > \sum_{j=i+1}^{n} 2^{n-j}$, the bit value of some bit i dominates the effect of all bits $i+1, \dots, n$ on the function value. Both will later be needed in our construction.

For two search points $x, y \in \{0,1\}^n$, we write $x \leq y$ if $x_i \leq y_i$ holds for all $1 \leq i \leq n$. We write $x < y$ if $x \leq y$ and $x \neq y$ hold. We call f a *strictly monotone function* (usually called simply *monotone* in the following) if for all $x, y \in \{0,1\}^n$ with $x < y$ it holds that $f(x) < f(y)$. Observe that the above condition is equivalent to $f(x) < f(y)$ for all x and y such that x and y only differ in exactly one bit and this bit has value 1 in y. In other words, every mutation that only flips bits with value 0 strictly increases the function value. Clearly, the all-ones bit string 1^n is the unique global optimum for a monotone function.

For the (1+1) EA, the difficulty of monotone functions strongly depends on the mutation probability $p(n)$. We are interested in mutation probabilities $p(n) = c/n$ for some constant $c \in \mathbb{R}^+$. For constants $c < 1$ on average in a single mutation less than one bit flips. If this is a 1-bit we have $f(x) > f(\mathrm{mut}(x))$ and $x = x^+$ holds. Otherwise, $f(x^+) > f(x)$ holds and we accept this move. This way the number of 0-bits is quickly reduced to 0 and the unique global optimum is found. Using drift analysis this reasoning can easily be made precise. We state the result here and omit the proof due to space restrictions.

Theorem 1. *The expected optimization time of the (1+1) EA with mutation probability $p(n) = c/n$, $0 < c < 1$ constant, is $\Theta(n \log n)$ for every monotone function.*

The proof of Theorem 1 breaks down for $c = 1$. In this case the drift in the number of 1-bits can be bounded pessimistically by a model due to Jansen [17] where we consider a random process that mutates x to y with mutation probability $p(n) = 1/n$ and replaces x by y if either $x \leq y$ holds or we have neither $x \leq y$ nor $y \leq x$ but $|y|_1 < |x|_1$ holds. This worst case model yields an upper bound of $O(n^{3/2})$ for the expected optimization time of the (1+1) EA with mutation probability $p(n) = 1/n$ on monotone functions.

Theorem 2. *The expected optimization time of the (1+1) EA with mutation probability $p(n) = 1/n$ is $O(n^{3/2})$ for every monotone function.*

Our main result is that using mutation probability $p(n) = c/n$ where c is a sufficiently large constant, optimization of monotone functions can become very difficult for the (1+1) EA. This is the first result where increasing the mutation probability by a constant factor increases the optimization time from polynomial to exponential with overwhelming probability.

Theorem 3. *There exists a monotone function $f\colon \{0,1\}^n \to \mathbb{N}$ such that the (1+1) EA with mutation probability $p(n) = c/n$, $c \geq 33$ constant, does not optimize f within $2^{\Omega(n)}$ mutations with overwhelming probability.*

The formal proof of this result is somewhat technical and lengthy. Therefore, in this extended abstract, we present how to construct such a monotone function f in the following section. In Section 4, we describe why this function is difficult to optimize. Complete proofs are available from the authors by request.

3 A Difficult to Optimize Monotone Function

In this section, we describe a monotone function that is difficult to optimize via a (1+1) EA with mutation probability $p(n) = c/n$, if $c > 33$ is constant.

The main idea is the construction of a kind of long path function (compare the work by Horn, Goldberg, and Deb [3]). We also have an exponentially long path such that shortcuts can only be taken if a large number of bits flip simultaneously, a very unlikely event. The construction is complicated by the fact that the function needs to be monotone. Hence we cannot forbid leaving the path by giving the boundary of the path an unfavorable fitness. We solve this problem, roughly speaking, by implementing the path on a level of bit strings having similar numbers of 1-bits. Monotonicity simply forbids leaving the level to strings having fewer 1-bits. The crucial part of our construction is setting up the function in such a way that, in spite of monotonicity, not too many 1-bits are collected.

For $x \in \{0,1\}^n$ let $B \subseteq [n]$ be a subset of all indices $[n]$. The bits x_i with $i \in B$ are referred to as window. The bits x_i with $i \notin B$ are outside the window. Inside the window the function value is given by BinVal. The weights for BinVal are ordered differently for each window in order to avoid correlation between windows. The window is placed such that there is only a small number of 0-bits outside the window. Reducing the number of 0-bits outside causes the window to be moved. This is a likely event that happens frequently. However, we manage to construct an exponentially long sequence of windows with the additional property that in order to come from one window to one with large distance in this sequence a large number of bits needs to be flipped simultaneously. Since this is highly unlikely, it is very likely that the sequence of windows is followed. This takes an exponential number of steps with overwhelming probability. Droste, Jansen, and Wegener [2] embed the long path into a unimodal function in a way that the (1+1) EA reaches the beginning of the path with probability close to 1. We adopt this technique and extend it to our monotone function.

The following Lemma 1 defines the sequence of windows of our function by defining the index sets B_i. The property that windows with large distance have large Hamming distance is formally stated as $|i - j| \geq \ell \Rightarrow |B_i \cap B_j| \leq \gamma\ell$ for $\ell = \Theta(n)$ and some constant $\gamma > 0$.

Lemma 1. *Let $\beta, \gamma \in \mathbb{R}$ be constants with $0 < \beta$ and $\rho := \beta/(1 - 2\beta) < \gamma < 1$. Let $n \in \mathbb{N}$ and $L := \lfloor \exp((\gamma - \rho)^2(1 - 2\beta)n/6) \rfloor$. Let $\ell := \beta n$ and*

$L' := L - \ell + 1$. Then there are $b_1, b_2, \ldots, b_L \in [n]$ such that the following holds. Let $B_i := \{b_i, b_{i+1}, \ldots, b_{i+\ell-1}\}$ for all $i \in [L']$. Then

(i) $|B_i| = \ell$ for all $i \in [L']$,
(ii) $|B_i \cap B_j| \leq \gamma\ell$ for all $i, j \in [L']$ such that $|i - j| \geq \ell$.

Proof. The proof invokes the probabilistic method [18], that is, we describe a way to randomly choose the b_i that ensures that properties (i) and (ii) hold with positive probability. This necessarily implies the existence of such a sequence.

Let the $b_1, b_2, \ldots, b_L$ be chosen uniformly at random subject to condition (i). More precisely, let $b_1 \in [n]$ be chosen uniformly at random. If $b_1, \ldots, b_{i-1}$ are already chosen, then choose b_i from $[n] \setminus \{b_{\max\{1,i-\ell\}}, \ldots, b_{i-1}\}$ uniformly at random.

Let $i, j \in [L']$ with $i < j$ and $|i - j| \geq \ell$. By definition, the sets B_i and B_j do not share an index. Fix any outcome of B_i. For all $k \in \{0, \ldots, \ell-1\}$ we have that, conditional on any outcomes of all other bs, the probability that $b_{j+k} \in B_i$ is at most $|B_i|/(n-2\ell) = \beta/(1-2\beta)$. Consequently, the random variable $C = |B_i \cap B_j|$ is dominated by a random variable X that is the sum of ℓ independent indicator random variables that are one with probability $\rho = \beta/(1-2\beta)$. Hence a simple Chernoff bound (cf. e.g. [19]) yields

$$\Pr(C > \gamma\ell) \leq \Pr(X > \gamma\ell) = \Pr(X > (1 + \tfrac{\gamma-\rho}{\rho}) \cdot \rho\ell) \leq \exp(-(\tfrac{\gamma-\rho}{\rho})^2 \rho\ell/3).$$

Since there are less than L^2 choices of $i, j \in [L']$, a simple union bound yields

$$\Pr(\exists i, j \in [L']\colon (|i - j| \geq \ell) \wedge (|B_i \cap B_j| > \gamma\ell)) < L^2 \exp(-(\tfrac{\gamma-\rho}{\rho})^2 \rho\ell/3) \leq 1. \quad \square$$

The following definition introduces the difficult monotone function we consider. Note that it assumes the sequence of windows B_i to be given. For $x \in \{0,1\}^n$ some $i \in [L']$ is a potential position in the sequence of windows if the number of 0-bits outside the window B_i is limited by αn, $\alpha > 0$ some constant. We select the largest potential position i as actual position and have the function value for x depend mostly on this position. If no potential position i exists, we have not yet found the path of windows and lead the (1+1) EA towards it. If $i = L'$, i.e., the end of the path is reached, the (1+1) EA is lead towards the unique global optimum via OneMax.

Definition 1. *Let β, γ, ℓ, L, L', the b_i and B_i be as in Lemma 1. Let $\alpha \in \mathbb{R}$ with $0 < \alpha < \beta$. For $x \in \{0,1\}^n$ let $\mathcal{B}_x := \{i \in [L'] \mid |\{j \in [n] \mid x_j = 0\} \setminus B_i| \leq \alpha n\}$. Let $i^*_x := \max \mathcal{B}_x$, if $\mathcal{B}_x$ is non-empty. For $i \in [L']$ let $\pi^{(i)}$ be a permutation of B_i. Denote by $\Pi = (\pi^{(1)}, \ldots, \pi^{(L')})$ the sequence of these permutations. We use the short-hand $\pi^{(i)}(x)$ to denote the vector obtained from permuting the components of $(x_{b_i}, \ldots, x_{b_{i+\ell-1}})$ according to $\pi^{(i)}$. Consequently, $\pi^{(i)}(x) = (x_{\pi^{(i)}(b_i)}, \ldots, x_{\pi^{(i)}(b_{i+\ell-1})})$.*

We define $f_\Pi : \{0,1\}^n \to \mathbb{N}_0$ via

$$f_\Pi(x) := \begin{cases} \left|x_{|[n]\setminus B_1}\right|_1 \cdot 2^n + \textsc{BinVal}(\pi^{(|x_{|[n]\setminus B_1}|_1)}(x)), & \text{if } \mathcal{B}_x = \emptyset, \\ i^*_x \cdot 2^{2n} + \textsc{BinVal}(\pi^{(n+i^*_x)}(x)), & \text{if } \mathcal{B}_x \neq \emptyset \text{ and } n + i^*_x < L', \\ L \cdot 2^{3n} + |x|_1, & \text{otherwise.} \end{cases}$$

We state one observation concerning the function f_Π that is important in the following. It states that as long as the end of the path of windows is not found the number of 0-bits outside is not only bounded by αn but equals αn. This property will be used later on to show that the window is moved frequently.

Lemma 2. *Let $f_\Pi \colon \{0,1\}^n \to \mathbb{N}_0$ be as in Definition 1. Let $x \in \{0,1\}^n$ with $\mathcal{B}_x \neq \emptyset$ and $i_x^* = \max \mathcal{B}_x$. If $n + i_x^* < L'$, then $\left|Z(x) \setminus B_{i_x^*}\right| = \alpha n$.*

Proof. By assumption we have $n + i_x^* < L'$. We consider $B_{n+i_x^*+1}$ and see that the set coincides with $B_{n+i_x^*}$ in all but two elements: we have $B_{n+i_x^*} \setminus B_{n+i_x^*+1} = \{b_{n+i_x^*}\}$ and $B_{i_x^*+1} \setminus B_{i_x^*} = \{b_{n+i_x^*+\ell}\}$. Consequently, $|Z(x) \setminus B_{n+i_x^*}|$ and $|Z(x) \setminus B_{n+i_x^*+1}|$ differ by at most one. Thus, $|Z(x) \setminus B_{n+i_x^*}| < \alpha n$ implies $|Z(x) \setminus B_{n+i_x^*+1}| \leq \alpha n$ and we can replace i_x^* by $i_x^* + 1$. This contradicts $i_x^* = \max \mathcal{B}_x$. We have $|Z(x) \setminus B_{n+i_x^*}| \leq \alpha n$ by definition and thus $|Z(x) \setminus B_{n+i_x^*}| = \alpha n$ follows. □

Our first main claim is that f_Π is in fact monotone. This is not difficult, but might, due to the complicated definition of f_Π, not be obvious.

Lemma 3. *For all Π as above, f_Π is monotone.*

Proof. Let $f := f_\Pi$. Let $x \in \{0,1\}^n$ and $j \in [n]$ such that $x_j = 0$. Let $y \in \{0,1\}^n$ be such that $y_k = x_k$ for all $k \in [n] \setminus \{j\}$ and $y_j = 1 - x_j$. That is, y is obtained from x by flipping the jth bit (which is zero in x) to one. To prove the lemma, it suffices to show $f(x) < f(y)$.

Let first $\mathcal{B}_x = \emptyset$. If $\mathcal{B}_y \neq \emptyset$ we have $f(x) < n \cdot 2^n + 2^n$ and $f(y) \geq 2^{2n}$ so $f(x) < f(y)$ follows. If $\mathcal{B}_y = \emptyset$ we have either $\left|x_{|[n]\setminus B_1}\right|_1 < \left|y_{|[n]\setminus B_1}\right|_1$ (in case $j \notin B_1$) or $\textsc{BinVal}(\pi^{(i)}(x)) < \textsc{BinVal}(\pi^{(i)}(y))$ (in case $j \in B_1$). In both cases, $f(x) < f(y)$ holds.

Now assume $\mathcal{B}_x \neq \emptyset$ and $n + i_x^* < L'$. By definition $\mathcal{B}_x \subseteq \mathcal{B}_y$, hence $i_y^* \geq i_x^*$. If $i_y^* = i_x^*$, then $j \in B_{i_x^*}$ and $f(y) > f(x)$ follows from $\textsc{BinVal}(\pi^{(n+i_y^*)}(y)) = \textsc{BinVal}(\pi^{(n+i_x^*)}(y)) > \textsc{BinVal}(\pi^{(n+i_x^*)}(x))$. If $i_y^* > i_x^*$, then $f(y) > f(x)$. In all other cases, $f(x) = L2^{3n} + |x|_1$ and $f(y) = L2^{3n} + |y|_1$, hence $f(y) > f(x)$. □

4 Proof of the Lower Bound

Theorem 4. *Consider the (1+1) EA with mutation probability c/n for $c \geq 33$ on the function $f := f_\Pi$ from Definition 1 where Π is chosen uniformly at random and the parameters are chosen according to $\beta := 10/131$, $\gamma := 10/111$, and $\alpha := 1/(1000c)$. There is a constant $\kappa > 0$ such that with probability $1 - 2^{-\Omega(n)}$ the (1+1) EA needs at least $2^{\kappa n}$ generations to optimize f.*

This result above shows that if f is chosen randomly (according to the construction described), then the (1+1) EA w. o. p. needs an exponential time to find the optimum. Clearly, this implies that there exists a particular function f, that is, a choice of Π, such that the EA faces these difficulties. This is Theorem 3.

The proof for Theorem 4 is long and technical. Therefore, we only discuss the main proof ideas here. A complete proof is available by request.

Both after a typical initialization, when $\mathcal{B}_x = \emptyset$, and afterwards, when $\mathcal{B}_x \neq \emptyset$ and $n + i_x^* < L'$, we have the following situation. There is a window of bits ($B_{i_x^*}$ if i_x^* is defined and B_1 otherwise) such that the fitness increases with BinVal as a function on the bits inside the window. Moreover, the fitness is always increased in case the mutation decreases the number of 0-bits outside the window. If $\mathcal{B}_x = \emptyset$ this is due to the term $\left|x_{|[n]\setminus B_1}\right|_1 \cdot 2^n$ in the fitness function and otherwise it is because the current i_x^*-value has increased. The gain in fitness is so large that it dominates any change of the bits in the window.

We claim that with this construction it is very likely that the current window always contains at least $\beta n/11$ 0-bits. This is proven by showing that in case the number of 0-bits in the window is in the interval $[\beta n/11, \beta n/10]$ then there is a tendency ("drift") to increase the number of 0-bits again. Applying a drift theorem by Oliveto and Witt [20] yields that even in an exponential number of generations the probability that the number of 0-bits in the window decreases below $\beta n/11$ is exponentially small. We first elaborate on why this drift holds and then explain how the lower bound of $\beta n/11$ 0-bits implies the claim.

If a mutation decreases the number of 0-bits outside the window, the bits inside the window are subject to random, unbiased mutations. Hence, if the number of 0-bits is at most $\beta n/10$ the expected number of bits flipping from 1 to 0 is larger than the expected number of bits flipping from 0 to 1. If the mutation probability is large enough, this makes up for the 0-bits lost outside the window and it leads to a net gain in 0-bits in expectation, with regard to the whole bit string. Note that the window is moved during such a mutation. As by Lemma 2 the number of 0-bits outside the window is fixed to αn, we have a net gain in 0-bits for the window, regardless of its new position.

In case the number of 0-bits outside the window remains put, acceptance depends on a BinVal instance on the bits inside the window. For BinVal accepting the result of a mutation is completely determined by the flipping bit with the largest weight. In an accepted step, this bit must have flipped from 0 to 1. All bits with smaller weights have no impact on acceptance and therefore are subject to random, unbiased mutations. If, among all bits with smaller weights, there is a sufficiently small rate of 0-bits, more bits will flip from 1 to 0 than from 0 to 1. In this case, we again obtain a net increase in the number of 0-bits in the window, in expectation. Here we also require a large mutation probability since every increase of BinVal implies that one 0-bit has been lost and a surplus of flipping 1-bits has to make up for this loss. This holds in particular since the window only contains βn bits and the surplus' absolute value must still be large.

For a fixed BinVal instance the bits tend to develop correlations between bit values and weights over time; bits with large weights are more likely to become 1 than bits with small weights. This development is disadvantageous since the above argument relies on many 1-bits with small weights. In order to break up these correlations we use random instances of BinVal wherever possible. These random instances change quickly. If $\mathcal{B}_x = \emptyset$ and, by Lemma 2, also if $n+i_x^* < L'$ we have at least αn 0-bits outside the current window and every mutation that flips exactly one of these bits leads to a new BinVal instance. Since this happens with probability $\Omega(1)$, this frequently prevents the algorithm from gathering 1-bits at bits with large weights. Pessimistically dealing with bits that have been touched by mutation while optimizing the same BinVal instance, a positive expected increase in the number of 0-bits can be shown.

How does the lower bound of $\beta n/11$ 0-bits inside the window imply Theorem 4? With overwhelming probability we start with $\mathcal{B}_x = \emptyset$ and at least $\beta n/10$ 0-bits in the window B_1. We maintain at least $\beta n/11$ 0-bits in B_1, while the algorithm is encouraged to turn the 0-bits outside of B_1 to 1 quickly. Once the number of 0-bits outside of B_1 has decreased to or below αn, the path has been reached. The 0-bits in B_1 thereby ensure that the initial i_x^*-value is at most βn. The reason is that every two sets B_i, B_j with $|i-j| \geq \ell$ only intersect in at most $\gamma\beta n$ bits, so $\beta n/11$ 0-bits in B_i imply at least $\beta n/11 - \gamma\beta n$ 0-bits outside of B_j. For j to become the new window, however, at most αn 0-bits outside of B_j are allowed. By choice of α, β, and γ, moving from B_1 to B_j requires a linear number of 0-bits in B_1 to flip to 1 if $j > \beta n$. The described mutation has probability $n^{-\Omega(n)}$. The last argument also shows that the probability of increasing i_x^* by more than βn in one generation is $n^{-\Omega(n)}$. Hence, with overwhelming probability in each generation the (1+1) EA only makes progress at most βn on the path. As the path has exponential length, the claimed lower bound follows.

5 Conclusions

In this work, we analyzed how the (1+1) EA optimizes monotone functions. We showed that the optimum of any monotone function is found efficiently if the mutation probability is at most $1/n$. Surprisingly, once the mutation probability exceeds $33/n$, the situation drastically changes. In this case, there are monotone functions that with high probability are not optimized within exponential time.

This results indicates that, to a greater extent than expected, care has to be taken when choosing the mutation probability, even if restricting oneself to mutation probabilities c^*/n with a constant c^*. Contrary to previous observations, e. g., for linear functions, it may well happen that constant factor changes in the mutation probability lead to more than constant factor changes in the efficiency.

Besides generally suggesting more research on the right mutation probability, this work leaves two particular problems open. (i) For the mutation probability $1/n$, give a sharp upper bound for the optimization time of monotone functions (this order of magnitude is between $\Omega(n \log n)$ and $O(n^{3/2})$). (ii) Determine the largest constant c^* such that the expected optimization time of the (1+1) EA with mutation probability $p(n) = c^*/n$ is $n^{O(1)}$ on every monotone function. Currently, we only know that $1 < c^* < 33$ holds.

Acknowledgments. The third author was supported by a postdoctoral fellowship from the German Academic Exchange Service. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 07/SK/I1205.

References

1. Mühlenbein, H.: How genetic algorithms really work. Mutation and hillclimbing. In: Proceedings of PPSN II, pp. 15–25. North-Holland, Amsterdam (1992)
2. Droste, S., Jansen, T., Wegener, I.: On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 13–22. Springer, Heidelberg (1998)
3. Horn, J., Goldberg, D., Deb, K.: Long path problems. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 149–158. Springer, Heidelberg (1994)
4. Igel, C., Toussaint, M.: A no-free-lunch theorem for non-uniform distributions of target functions. J. of Mathematical Modelling and Algorithms 3, 313–322 (2004)
5. Oliveto, P., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4, 281–293
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
7. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127, 57–85 (2001)
8. He, J., Yao, X.: Erratum to [7]. Artificial Intelligence 140, 245–248 (2002)
9. Jägersküpper, J.: A blend of Markov-chain and drift analysis. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 41–51. Springer, Heidelberg (2008)
10. Doerr, B., Fouz, M., Witt, C.: Quasirandom evolutionary algorithms. In: Proceedings of GECCO 2010, pp. 1457–1464. ACM, New York (2010)
11. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: Proceedings of CEC 2010, pp. 1967–1974. IEEE, Los Alamitos (2010)
12. Doerr, B., Goldberg, L.: Adaptive drift analysis. In: Schaefer, R., et al. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 32–41. Springer, Heidelberg (2010)
13. Jansen, T., Wegener, I.: On the choice of the mutation probability for the (1+1) EA. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 89–98. Springer, Heidelberg (2000)
14. Bäck, T., Fogel, D., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Oxford University Press, Oxford (1997)
15. Ochoa, G.: Setting the mutation rate: Scope and limitations of the 1/L heuristic. In: Proceedings of GECCO 2002, pp. 495–502. Morgan Kaufmann, San Francisco (2002)
16. Cervantes, J., Stephens, C.R.: Limitations of existing mutation rate heuristics and how a rank GA overcomes them. IEEE Trans. on Evol. Comp. 13, 369–397 (2009)
17. Jansen, T.: On the brittleness of evolutionary algorithms. In: Stephens, C.R., Toussaint, M., Whitley, L.D., Stadler, P.F. (eds.) FOGA 2007. LNCS, vol. 4436, pp. 54–69. Springer, Heidelberg (2007)
18. Alon, N., Spencer, J.H.: The Probabilistic Method, 2nd edn. Wiley, Chichester (2000)
19. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
20. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. Algorithmica (2010), http://dx.doi.org/10.1007/s00453-010-9387-z

Log-Linear Convergence of the Scale-Invariant $(\mu/\mu_w, \lambda)$-ES and Optimal μ for Intermediate Recombination for Large Population Sizes

Mohamed Jebalia and Anne Auger

TAO Team - INRIA Saclay-Ile-de-France, LRI, Paris-Sud University,
91405 Orsay Cedex, France
firstname.lastname@inria.fr

Abstract. Evolution Strategies (ESs) are population-based methods well suited for parallelization. In this paper, we study the convergence of the $(\mu/\mu_w, \lambda)$-ES, an ES with weighted recombination, and derive its optimal convergence rate and optimal μ especially for large population sizes. First, we theoretically prove the log-linear convergence of the algorithm using a scale-invariant adaptation rule for the step-size and minimizing spherical objective functions and identify its convergence rate as the expectation of an underlying random variable. Then, using Monte-Carlo computations of the convergence rate in the case of equal weights, we derive optimal values for μ that we compare with previously proposed rules. Our numerical computations show also a dependency of the optimal convergence rate in $\ln(\lambda)$ in agreement with previous theoretical results.

1 Introduction

Evolution Strategies (ESs) are robust stochastic search methods [2,3] for solving continuous optimization problems where the goal is to minimize[1] a real valued objective function f defined on an open subset of $\mathbb{R}^d$. At each iteration of an ES, new solutions are in general generated by adding Gaussian perturbations (mutations) to some (optionally recombined) current ones. These Gaussian mutations are parameterized by the step-size giving the general scale of the search, and the covariance matrix giving the principal directions of the Gaussian distribution. In state-of-the art ESs, these parameters are adapted at each iteration [1,2,3,4]. We focus on isotropic ESs where the step-size is adapted and the covariance matrix is kept equal to the identity matrix I_d and therefore the search distribution is spherical. Adaptation in ESs allows them to have a log-linear behavior (convergence or divergence) when minimizing spherical objective functions [10,13,5,7]. Log-linear convergence (resp. divergence) means that there exists a constant value $c < 0$ called convergence rate (resp. $c > 0$) such that the distance to the optimum, d_n, at an iteration n satisfies $\lim_n \frac{1}{n}\ln(d_n) = c$. Spherical objective functions are defined as

$$f(x) = g(\|x\|), \tag{1}$$

[1] Without loss of generality, the minimization of a real value function f is equivalent to the maximization of $-f$.

where $g : [0, \infty[\mapsto \mathbb{R}$ is a strictly increasing function, $x \in \mathbb{R}^d$ and $\|.\|$ denotes the Euclidean norm on $\mathbb{R}^d$. Log-linear behavior holds also when minimizng spherical functions perturbed by noise [11].

In this paper, we investigate ESs with weighted recombination, denoted $(\mu/\mu_w, \lambda)$-ES, and used in the state-of-the-art ES, the Covariance Matrix Adaptation-ES (CMA-ES) [4]. The $(\mu/\mu_w, \lambda)$-ES is an ES which evolves a single solution. Let $\mathbf{X}_n$ be the solution (the parent) at iteration n, λ new solutions $\mathbf{Y}_n^i$ (offspring) are then generated using independent Gaussian samplings of mean $\mathbf{X}_n$. Then, the offspring are evaluated, the μ best offspring $(\mathbf{Y}_n^{i:\lambda})_{1 \le i \le \mu}$ are selected and the new solution $\mathbf{X}_{n+1}$ is obtained by recombining these selected offspring using recombination weights denoted $\left(w^i\right)_{1 \le i \le \mu}$, i.e., $\mathbf{X}_{n+1} = \sum_{i=1}^{\mu} w^i \mathbf{Y}_n^{i:\lambda}$[2]. We will specifically study the $(\mu/\mu_w, \lambda)$-ES with large (offspring) population size λ compared to the search space dimension d, i.e., $\lambda \gg d$. This is motivated by the increasing possibilities of parallelization with the raise of the number of parallel machines, supercomputers and grids. ESs are population-based methods and then are well suited for parallelization which consists in distributing the number of evaluations λ on the processes available. The performance of the $(\mu/\mu_w, \lambda)$-ES as a function of λ has been theoretically investigated [14,16]. Under the approximation $d \to +\infty$, the study in [14] investigated the $(\mu/\mu_w, \lambda)$-ES minimizing any spherical function and using an artificial step-size adaptation rule termed scale-invariant which sets the step-size at each iteration proportionally to the distance of the current solution to the optimum. The progress rate φ which measures the one-step expected progress to the optimum verifies $\varphi = O\left(\mu \ln\left(\frac{\lambda}{\mu}\right)\right)$ [14]. This suggests that, if μ is chosen proportional to λ, the progress rate of the $(\mu/\mu_w, \lambda)$-ES can be linear in μ and in λ. The study in [16] is based on a theoretical computations of lower bounds for the convergence ratio which measures the convergence rate in probability of wide classes of ESs. It shows that the convergence ratio of the $(\mu/\mu_w, \lambda)$-ES varies at best linearly with $\ln(\lambda)$ for sufficiently large λ when minimizing any spherical function [16]. This suggests that the bound found in [14] is not tight for finite dimensions.

A natural question arising when using recombination is how to choose the number of offspring μ to be recombined. Studies based on computations of the progress rate when the search space dimension goes to infinity suggest to use $\mu = \lfloor \frac{\lambda}{4} \rfloor$ [14] or $\mu = \lfloor \frac{\lambda}{2} \rfloor$ [6][3] for two different choices of the (positive) weights $\left(w^i\right)_{1 \le i \le \mu}$. CMA-ES which has been designed to work well on small population sizes uses $\mu = \lfloor \frac{\lambda}{2} \rfloor$ as a default parameter. However, when using a large population size λ, the convergence rate of some real-world algorithms tested in [15,8] using the rules $\mu = \lfloor \frac{\lambda}{4} \rfloor$ or $\mu = \lfloor \frac{\lambda}{2} \rfloor$ as recommended in [14,6] is worse than the theoretical prediction of [16]. This is due to the fact that the rules used in these tests for choosing μ, are recommended by the studies performed under the approximation $(d \to +\infty)$ [14,6] and thus under the assumption $\lambda \ll d$. For some values of λ and d such that $\lambda \gg d$, Beyer [17] computed, using some approximations permitted by the assumption $(d \to +\infty)$, optimal choices for μ when minimizing spherical functions. However, no explicit rule for the choice of μ has been

[2] If $\mu = 1$, only the best offspring is taken and then the $(\mu/\mu_w, \lambda)$-ES is simply the $(1, \lambda)$-ES.

[3] The rule proposed in [6] where negative weights are allowed is rather $\mu = \lambda$, but the study implies that if the weights can be only positive the rule becomes $\mu = \lfloor \frac{\lambda}{2} \rfloor$.

proposed when $\lambda \gg d$. Performing experiments with $\lambda \gg d$ on a $(\mu/\mu_w, \lambda)$-ES using equal weights, the so-called self-adaptation rule for the step-size and two variants for the covariance matrix adaptation, Teytaud [9] proposed to choose μ equal to $\min\{d, \lfloor \frac{\lambda}{4} \rfloor\}$.

Since it is in general difficult to appraise whether the effect observed when changing the setting of one parameter on a real algorithm is coming from the fact that the setting of an other parameter may subsequently becomes sub-optimal, we want here to identify independently of any real step-size or covariance matrix update rule the optimal setting for μ especially for lagre λ. This optimal setting can be used to identify a rule for choosing best optimal values μ in real-world algorithms like CMA-ES. We want also to verify whether an optimal choice for μ allows to have a dependency of the convergence rate in $\ln(\lambda)$ and thus reach the lower bounds predicted by [16]. In order to do so, we perform in this paper a theoretical and numerical investigation of the convergence and the optimal choice for μ relative to the isotropic $(\mu/\mu_w, \lambda)$-ES. We focus on large population sizes. The objective functions investigated are the spherical functions allowing ESs which do not use recombination to reach optimal convergence rates [5,7]. In Section 2, we present the mathematical formulation of the algorithm. In Section 3, we identify the optimal step-size adaptation rule of the algorithm when minimizing spherical functions. In Section 4, we theoretically prove the log-linear convergence of the algorithm using the scale-invariant adaptation rule and identify its convergence rate. In Section 5, using Monte-Carlo computations of the convergence rate, optimal μ values and optimal convergence rates are derived for some dimensions and in the specific case of equal weights $(w^i)_{1\leq i\leq\mu}$. A new rule for choosing μ is proposed based on our results. Throughout the paper, we explain only the basic ideas of the proofs because of space limitation. For complete proofs, we refer to [12].

2 Mathematical Formulation of the Isotropic $(\mu/\mu_w, \lambda)$ Evolution Strategy Minimizing Spherical Functions

Throughout the remainder of this paper, we suppose that μ and λ are two positive integers such that $1 \leq \mu \leq \lambda$, and that the recombination weights $(w^i)_{1\leq i\leq\mu}$ are positive constants summing to one, i.e., $\sum_{i=1}^{\mu} w^i = 1$. In this section we will introduce the mathematical formulation of the isotropic $(\mu/\mu_w, \lambda)$-ES for minimizing a spherical function (1). Let $\mathbf{X}_0 \in \mathbb{R}^d$ be the first solution randomly chosen using a law absolutely continuous with respect to the Lebesgue measure. Let σ_0 be a strictly positive variable (possibly) randomly chosen. Let $(\mathbf{N}_n^i)_{i\in[1,\lambda],n\in\mathbb{Z}^+}$, be a sequence of random vectors defined on a probability space $(\Omega, \mathcal{A}, P)$, independent and identically distributed (i.i.d.) with common law the isotropic multivariate normal distribution on $\mathbb{R}^d$ with mean $(0,\ldots,0) \in \mathbb{R}^d$ and covariance matrix identity I_d, which we will denote $\mathcal{N}(0, I_d)$. We assume that the sequence $(\mathbf{N}_n^i)_{i\in[1,\lambda],n\in\mathbb{Z}^+}$ is independent of $\mathbf{X}_0$. Let σ_n be the step-size mutation at iteration n such that for all $(i,n) \in [1,\lambda] \times \mathbb{Z}^+$, σ_n and $\mathbf{N}_n^i$ are independent. An offspring $\mathbf{Y}_n^i$ where $i = 1,\ldots,\lambda$ writes as $\mathbf{Y}_n^i := \mathbf{X}_n + \sigma_n \mathbf{N}_n^i$, and its objective function value is $g(\|\mathbf{Y}_n^i\|)$ in our case of minimization of spherical functions. Let $\mathbf{N}_n^{i:\lambda}(\mathbf{X}_n, \sigma_n)$ $(1 \leq i \leq \mu)$ denotes the mutation vector relative to the i^{th} best offspring according to its fitness value. As the function g is increasing, the vectors $\mathbf{N}_n^{i:\lambda}(\mathbf{X}_n, \sigma_n)$ (where, for all i in $\{1,\ldots,\mu\}$, the indices $i\!:\!\lambda$ are in $\{1,\ldots,\lambda\}$) verify:

$$\begin{aligned}\left\|\mathbf{X}_n+\sigma_n\mathbf{N}_n^{1:\lambda}(\mathbf{X}_n,\sigma_n)\right\| &\leq \ldots \leq \left\|\mathbf{X}_n+\sigma_n\mathbf{N}_n^{\mu:\lambda}(\mathbf{X}_n,\sigma_n)\right\| \text{ and}\\ \left\|\mathbf{X}_n+\sigma_n\mathbf{N}_n^{\mu:\lambda}(\mathbf{X}_n,\sigma_n)\right\| &\leq \left\|\mathbf{X}_n+\sigma_n\mathcal{N}_n^{j}\right\| \forall j\in\{1,\ldots,\lambda\}\backslash\{1:\lambda,\ldots,\mu:\lambda\}\,.\end{aligned} \tag{2}$$

Using the fact that $\sum_{i=1}^{\mu} w^i=1$, the new parent $\mathbf{X}_{n+1}=\sum_{i=1}^{\mu} w^i\mathbf{Y}_n^{i:\lambda}$ can be rewritten as:

$$\mathbf{X}_{n+1}=\mathbf{X}_n+\sigma_n\sum_{i=1}^{\mu} w^i\mathbf{N}_n^{i:\lambda}(\mathbf{X}_n,\sigma_n)\,. \tag{3}$$

In the specific case where the scale-invariant rule is used for the adaptation of $(\sigma_n)_{n\in\mathbb{Z}^+}$, i.e., $\sigma_n=\sigma\|\mathbf{X}_n\|$ (with $\sigma>0$), the previous equation becomes:

$$\mathbf{X}_{n+1}=\mathbf{X}_n+\sigma\|\mathbf{X}_n\|\sum_{i=1}^{\mu} w^i\mathbf{N}_n^{i:\lambda}(\mathbf{X}_n,\sigma\|\mathbf{X}_n\|)\,. \tag{4}$$

Finally, σ_n is updated, i.e., σ_{n+1} is computed independently of $\mathbf{N}_{n+1}^i$ for all $i\in[1,\lambda]$. Throughout the remainder of this paper, we will denote in a general context where $u\in\mathbb{R}^d$, $s\in\mathbb{R}$ and $(\mathbf{N}_n^i)_{i\in[1,\lambda],n\in\mathbb{Z}^+}$ is a sequence of random vectors (i.i.d.) with common law $\mathcal{N}(0,I_d)$ and such that for all $(i,n)\in[1,\lambda]\times\mathbb{Z}^+$, $\mathbf{N}_n^i$ is independent of u and s, $\mathbf{N}_n^{i:\lambda}(u,s)$ the random vector which verifies (2) where $\mathbf{X}_n$ and σ_n are respectively replaced by u and s. For $n=0$ and $i\in\{1,\ldots,\mu\}$, the notation $\mathbf{N}_0^{i:\lambda}(u,s)$ will be replaced by the notation $\mathbf{N}^{i:\lambda}(u,s)$.

3 Optimal Step-Size Adaptation Rule When Minimizing Spherical Functions

The (log-linear) convergence rate of the isotropic scale-invariant $(\mu/\mu_w,\lambda)$-ES minimizing any spherical function and satisfying the recurrence relation (4) is, as will be shown in Section 4, the function V that we will introduce in the following definition.

Definition 1. *Let* e_1 *denotes the unit vector* $(1,0,\ldots,0)\in\mathbb{R}^d$*. For* $\sigma\geq 0$*, let* $\mathrm{Z}(\sigma)$ *be the random variable defined as* $\mathrm{Z}(\sigma):=\left\|\mathrm{e}_1+\sigma\sum_{i=1}^{\mu} w^i\mathbf{N}^{i:\lambda}(\mathrm{e}_1,\sigma)\right\|$ *where the random variables* $\mathbf{N}^{i:\lambda}(\mathrm{e}_1,\sigma)$ *are obtained similarly to (2) but with* $n=0$ *and* $(\mathbf{X}_n,\sigma_n)$ *replaced by* (e_1,σ)*. We introduce the function* V *as the function mapping* $[0,+\infty[$ *into* $\mathbb{R}$ *as follows:*

$$\mathrm{V}(\sigma):=E\left[\ln\mathrm{Z}(\sigma)\right]=E\left[\ln\left\|\mathrm{e}_1+\sigma\sum_{i=1}^{\mu} w^i\mathbf{N}^{i:\lambda}(\mathrm{e}_1,\sigma)\right\|\right]\,. \tag{5}$$

Fig. 1 (left) represents numerical computations of the function V in some specific settings. In the following proposition, we show that V is well defined and we study its properties. Note that in the following, the notation V will be sometimes replaced by V_μ when we need to stress the dependence of V on μ.

Proposition 1. *The function* V *introduced in* (5) *has the following properties:*

(i) V *is well defined for* $d \geq 1$, *and continuous for* $d \geq 2$, *on* $[0, +\infty[$.
(ii) *For* $d \geq 2$, $\lim_{\sigma \to +\infty} \mathrm{V}(\sigma) = +\infty$.
(iii) *If* $\mu \leq \frac{\lambda}{2}$, *for* $d \geq 2$, $\exists\ \bar{\sigma} > 0$ *such that* $\mathrm{V}(\bar{\sigma}) < 0$.
(iv) *If* $\mu \leq \frac{\lambda}{2}$, *for* $d \geq 2$, $\exists\ \sigma_{opt} > 0$ *such that* $\inf_{\{\sigma \geq 0\}} \mathrm{V}(\sigma) = \mathrm{V}(\sigma_{opt}) < 0$.
(v) *For* $d \geq 2$ *and* $\lambda \geq 2$, *if* $\mu \leq \lambda/2$, $\exists\ (\sigma_{opt}, \mu_{opt})$ *such that* $\mathrm{V}_{\mu_{opt}}(\sigma_{opt}) = \inf_{\{\sigma \geq 0, \mu \leq \lambda/2\}} \mathrm{V}_{\mu}(\sigma) < 0$.

Summary of the proof A basic step in the proof of (i) and (ii) is to write V as the sum of $\mathrm{V}^{+}(\sigma) := E\left[\ln^{+} \mathrm{Z}(\sigma)\right]$ and $\mathrm{V}^{-}(\sigma) := E\left[\ln^{-} \mathrm{Z}(\sigma)\right]$. Then, for (i), integrands in these quantities are upper bounded by quantities which do not depend on σ and the result follows by the Lebesgue dominated convergence theorem for continuity. For (ii), we show that $\mathrm{V}(\sigma)$ is lower bounded by an expectation of a given random variable which depends on σ. We show using the Monotone convergence theorem that this lower bound converges to infinity when σ goes to infinity and then the result follows. For proving (iii), we prove before, using the concept of uniform integrability of a family of random variables that $d\,\mathrm{V}\left(\frac{\sigma^*}{d}\right)$ ($\sigma^* > 0$ fixed) converges to a certain limit depending on σ^* when d goes to $+\infty$. Using the fact that this limit can be negative for a given σ^* we prove our claim. (iv) is proven using (i), (ii) and (iii) and the intermediate value theorem. (v) follows easily from (iv).

An important point that we can see from this proposition is that, given $\lambda \geq 2$ and $d \geq 2$, and under the condition $\mu \leq \lambda/2$, μ and σ can be chosen such that the relative convergence rate V is optimal (v). We conducted numerical computations of V in the case where $d = 10$, $\lambda = 10$ and equal weights $(w^i)_{1 \leq i \leq \mu}$. The cases with $\mu = 1, 2$ and 5 are represented in Fig. 1 (left). It can be seen that the curves are in conformity with (i), (iii), (iv) and (v) of Proposition 1. In particular, for each μ, there exists a σ_{opt} realizing the minimum of V and we can see that the optimal μ (among the represented μ values 1,2 and 5) is 2. In the following theorem, we will see that the optimal value of V is also the optimal convergence rate in expectation that can be reached by the $(\mu/\mu_w, \lambda)$-ES minimizing a spherical function and using any step-size adaptation rule $(\sigma_n)_{n \geq 0}$, or more precisely, the smallest value of $\frac{1}{n} E\left[\ln \frac{\|\mathbf{X}_n\|}{\|\mathbf{X}_0\|}\right]$ that can be reached by the sequence $(\mathbf{X}_n)_{n \geq 0}$ satisfying the recurrence relation (3). This optimal value corresponds also to the smallest value of $\frac{1}{n} E\left[\ln \frac{\|\mathbf{X}_n\|}{\|\mathbf{X}_0\|}\right]$ that can be reached by the isotropic scale-invariant $(\mu/\mu_w, \lambda)$-ES minimizing a spherical function, i.e., where $(\mathbf{X}_n)_{n \geq 0}$ satisfies the recurrence relation (4) with $\sigma = \sigma_{opt}$.

Theorem 1. *Let* $(\mathbf{X}_n)_{n \geq 0}$ *be the sequence of random vectors satisfying the recurrence relation* (3) *and relative to the* $(\mu/\mu_w, \lambda)$*-ES minimizing any spherical function* (1)*. Then, for* $\lambda \geq 2$ *and* $d \geq 2$*, we have*

$$\frac{1}{n} E\left[\ln \frac{\|\mathbf{X}_n\|}{\|\mathbf{X}_0\|}\right] \geq \mathrm{V}(\sigma_{opt}) \,, \tag{6}$$

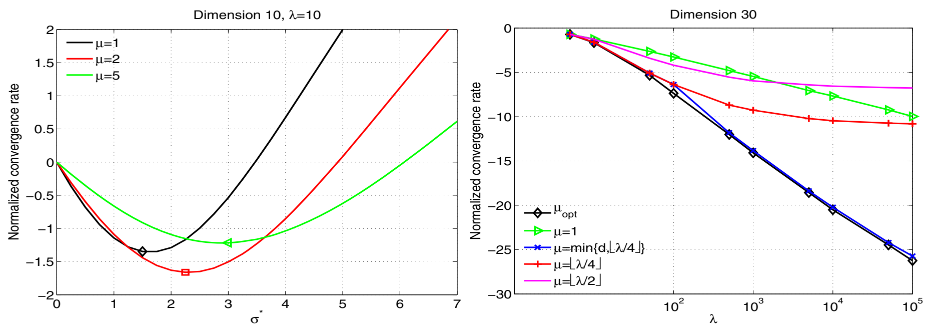


Fig. 1. Left: Plots of the normalized convergence rate $d \times \mathrm{V}_\mu(\frac{\sigma^*}{d})$ where V_μ $(= \mathrm{V})$ is defined in (5) as a function of $\sigma^* > 0$ with $d = 10$, $\lambda = 10$, $w^i = \frac{1}{\mu}$, $\forall\, i = 1, \ldots, \mu$ and $\mu \in \{1, 2, 5\}$. The plots were obtained doing Monte-Carlo estimations of V using 10^6 samples. **Right:** Optimal convergence rate $(d \times \mathrm{V}_\mu(\frac{\sigma^*_{\mathrm{opt}}}{d}))$ associated to different choices of μ as a function of λ for dimension 30 and μ_{opt} realizing the minimum of $(\sigma^*, \mu) \mapsto \mathrm{V}_\mu(\frac{\sigma^*}{d})$.

where σ_{opt} is given in Proposition 1 as $\sigma_{opt} = argmin_{\{\sigma>0\}}\mathrm{V}(\sigma)$ and $\mathrm{V}(\sigma_{opt})$ corresponds to $\frac{1}{n}E\left[\ln\left(\frac{\|\mathbf{X}_n\|}{\|\mathbf{X}_0\|}\right)\right]$ for a $(\mu/\mu_w, \lambda)$-ES using the specific scale-invariant adaptation rule with $\sigma_n = \sigma_{opt}\|\mathbf{X}_n\|$ and minimizing any spherical function (1).

Summary of the proof. The first step for proving the theorem is to remark that:

$$E\left[\ln\frac{\|\mathbf{X}_{k+1}\|}{\|\mathbf{X}_k\|}\right]$$
$$= E\left[E\left[\ln\left\|\frac{\mathbf{X}_k}{\|\mathbf{X}_k\|} + \frac{\sigma_k}{\|\mathbf{X}_k\|}\sum_{i=1}^{\mu} w^i \mathbf{N}_k^{i:\lambda}\left(\frac{\mathbf{X}_k}{\|\mathbf{X}_k\|}, \frac{\sigma_k}{\|\mathbf{X}_k\|}\right)\right\| \mid (\mathbf{X}_k, \sigma_k)\right]\right].$$

By the isotropy of the norm function and of the multivariate normal distribution, the term $\frac{\mathbf{X}_k}{\|\mathbf{X}_k\|}$ in the previous equation can be replaced by e_1. Then $E\left[\ln\frac{\|\mathbf{X}_{k+1}\|}{\|\mathbf{X}_k\|}\right] = E\left[\mathrm{V}\left(\frac{\sigma_k}{\|\mathbf{X}_k\|}\right)\right]$ where $E\left[\mathrm{V}\left(\frac{\sigma_k}{\|\mathbf{X}_k\|}\right)\right]$ is, by Proposition 1, lower bounded by $\mathrm{V}(\sigma_{opt})$. The result follows from summing such inequalities from $k = 0$ to $k = n - 1$.

This theorem states that the artificial scale-invariant adaptation rule with the specific setting $\sigma_n = \sigma_{opt}\|\mathbf{X}_n\|$ is the rule which allows to obtain the best convergence rate of the $(\mu/\mu_w, \lambda)$-ES when minimizing spherical functions. The relative convergence rate is then a tight lower bound that can be reached in this context. Then, for our study on minimization of spherical functions, we will use the $(\mu/\mu_w, \lambda)$-ES with the artificial scale-invariant adaptation rule, i.e., with $\sigma_n = \sigma\|\mathbf{X}_n\|$ where σ is a strictly positive constant. In the specific case where σ equals σ_{opt}, the convergence rate is optimal.

4 Log-Linear Behavior of the Scale-Invariant $(\mu/\mu_w, \lambda)$-ES Minimizing Spherical Functions

Log-linear convergence of ESs can be in general shown using the application of different Law of Large Numbers (LLN) such as LLN for independent or orthogonal random variables or LLN for Markov chains. Log-linear behavior has been shown for ESs which do not use recombination [10,5,13,7]. The key idea of the proof is stated in the following proposition.

Proposition 2. *Let $\sigma \geq 0$ and let $(\mathbf{X}_n)_n$ be the sequence of random vectors satisfying the recurrence relation (4). We introduce the sequence of random variables $(\mathrm{Z}_n)_{n\in\mathbb{Z}^+}$ by $\mathrm{Z}_n := \left\| \frac{\mathbf{X}_n}{\|\mathbf{X}_n\|} + \sigma \sum_{i=1}^{\mu} w^i \mathbf{N}_n^{i:\lambda}\left(\frac{\mathbf{X}_n}{\|\mathbf{X}_n\|}, \sigma\right) \right\|$ where $\mathbf{N}_n^{i:\lambda}\left(\frac{\mathbf{X}_n}{\|\mathbf{X}_n\|}, \sigma\right)$ are obtained similarly to (2) but with replacing $(\mathbf{X}_n, \sigma_n)$ by $\left(\frac{\mathbf{X}_n}{\|\mathbf{X}_n\|}, \sigma\right)$. Then for $n \geq 0$, we have*

$$\frac{1}{n} \ln \frac{\|\mathbf{X}_n\|}{\|\mathbf{X}_0\|} = \frac{1}{n} \sum_{k=0}^{n-1} \ln \mathrm{Z}_k \ a.s. \tag{7}$$

Using the isotropy of the norm function and of the multivariate normal distribution, the terms $\ln \mathrm{Z}_k$ appearing in the right hand side of the previous equation are independent identically distributed with a common expectation $\mathrm{V}(\sigma)$ which we have proved to be finite in Proposition 1. The following theorem is then obtained by the application of the LLN for independent identically distributed random variables with a finite expectation to the right hand side of the previous equation.

Theorem 2 (Log-linear Behavior of the Scale-invariant $(\mu/\mu_w, \lambda)$-ES). *The scale-invariant $(\mu/\mu_w, \lambda)$-ES defined in (4) and minimizing any spherical function (1) converges (or diverges) log-linearly in the sense that for $\sigma > 0$ the sequence $(\mathbf{X}_n)_n$ of random vectors given by the recurrence relation (4) verifies*

$$\lim_{n\to+\infty} \frac{1}{n} \ln \|\mathbf{X}_n\| = \mathrm{V}(\sigma) \tag{8}$$

almost surely, where V refers to the quantity defined in (5).

Theorem 2 establishes that, provided that V is non zero, the convergence of the scale-invariant $(\mu/\mu, \lambda)$-ES minimizing any spherical objective function given in (1) is log-linear. This theorem also provides the convergence (or divergence) rate $\mathrm{V}(\sigma)$ of the sequence $(\ln(\|\mathbf{X}_n\|))_n$: If $\mathrm{V}(\sigma) < 0$, the distance to the optimum, $(\|\mathbf{X}_n\|)_{n\geq 0}$, converges log-linearly to zero and if $\mathrm{V}(\sigma) > 0$, the algorithm diverges log-linearly. From Proposition 1, we know that, for all $d \geq 2$, for all $\lambda \geq 2$ and all $\mu \geq 1$ with the condition $\mu \leq \lambda/2$, there exists $\sigma > 0$ such that $\mathrm{V}(\sigma) < 0$ and therefore the algorithm converges. Moreover, by the same proposition, we know that for any $d, \lambda \geq 2$ there is an optimal choice of (σ, μ) such that the optimal convergence rate is reached.

A practical interest of this result is that if someone chooses the optimal value of μ and is able to tune the adaptation rule of his algorithm such that the quantity $\frac{\sigma_n}{\|\mathbf{X}_n\|}$ is (after an adaptation time) stable around the optimal value for σ, a convergence rate close to the optimal convergence rate can be obtained at least for spherical functions. This can be useful especially for choosing μ when the population size λ is large.

The goal is then to compute those optimal values (i.e., μ_{opt} and σ_{opt}) depending on λ and d. Fortunately, another important point of Theorem 2 is that the convergence rate is expressed in terms of the expectation of a given random variable (see Definition 1). Therefore, the convergence rate V can be numerically computed using Monte-Carlo simulations. Numerical computations allowing to derive optimal convergence rate values and relative optimal values of μ will be investigated in the following section.

5 Numerical Experiments

In this section, we numerically compute, for a fixed dimension and λ, values of μ leading to optimal convergence rates. We compare the convergence rate associated to those optimal μ with the ones obtained with previous choices of μ (proportional to $\lfloor \lambda/2 \rfloor, \ldots$). We also investigate how the optimal convergence rate depends on the population size λ in particular for $\lambda \gg d$. The context of our numerical study is the specific $(\mu/\mu_w, \lambda)$-ES with intermediate recombination, i.e., with equal weights $w^i = \frac{1}{\mu}$, $(i = 1, \ldots, \mu)$ which is simply denoted $(\mu/\mu, \lambda)$-ES.

Since V is expressed in terms of expectation of a random variable, we can perform a Monte-Carlo simulation of the normalized convergence rate $d \times \mathrm{V}_\mu \left(\frac{\sigma^*}{d}\right)$ where $\sigma^* > 0$ is called normalized step-size. The values computed are then relative to the scale-invariant $(\mu/\mu, \lambda)$-ES with $\sigma_n = \frac{\sigma^*}{d} \|\mathbf{X}_n\|$ and minimizing a spherical function. Our experimental procedure relies on finding the minimal value of $(\sigma^*, \mu) \mapsto d \times \mathrm{V}_\mu \left(\frac{\sigma^*}{d}\right)$ for μ in a range $\mu_{\mathbf{range}}$ and for values of σ^* taken in a range $\sigma_{\mathbf{range}}$. The minimal value, denoted $d \times \mathrm{V}_{\mu_{opt}} \left(\frac{\sigma^*_{opt}}{d}\right)$, is the normalized optimal convergence rate. However, we will also call 'normalized optimal convergence rate' the minimal value of $\sigma^* \mapsto d \times \mathrm{V}_\mu \left(\frac{\sigma^*}{d}\right)$ for μ fixed which we denote $d \times \mathrm{V}_\mu \left(\frac{\sigma^*_{opt}}{d}\right)$. The difference should be clear within the context.

As a first experiment, we took $\mu_{\mathbf{range}} = \{2^k; k \in \mathbb{Z}^+ \text{and } 2^k \leq \frac{\lambda}{2}\}$ and $\sigma_{\mathbf{range}} = \ln(\mu + 1) * \ln(\lambda) * [0 : 0.1 : 3]$. We experimented discrete values of λ from $\lambda = 5$ to $\lambda = 10^5$ with a number of Monte-Carlo samplings decreasing as a function of λ from 10^4 to 500. These first computations show that for the values of λ and d tested, the approximation

$$\min_{\{\sigma^* \in \sigma_{\mathbf{range}}\}} d \times \mathrm{V}_\mu \left(\frac{\sigma^*}{d}\right) \simeq a(\lambda, d) \ln^2(\mu) + b(\lambda, d) \ln(\mu) + c(\lambda, d) \tag{9}$$

is reliable (for $\mu > 1$) and we determined numerically the coefficients $a(\lambda, d), b(\lambda, d)$ and $c(\lambda, d)$. Using these quadratic approximations, we performed a second serie of tests where the values of μ were taken around the optimal value of the polynomial approximation, $\sigma_{\mathbf{range}} = m * \ln(\mu + 1) * \ln(\lambda) * [0 : 0.1 : 3]$ (with $m \leq \frac{2}{3}$) and using more Monte-Carlo samplings.

In Fig. 2 (left), we plotted the normalized optimal convergence rate values and the normalized optimal convergence rates relative to the rule $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$ from [9] as

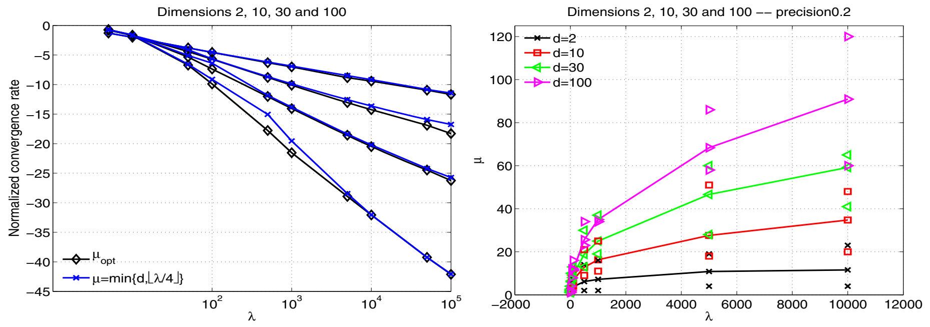


Fig. 2. Left: Plots of the normalized optimal convergence rate $d \times \mathrm{V}_{\mu_{opt}}\left(\frac{\sigma^*_{opt}}{d}\right)$ where $V_\mu (= \mathrm{V})$ is defined in (5) and normalized optimal convergence rate ($d \times \mathrm{V}_\mu(\frac{\sigma^*_{\mathrm{opt}}}{d})$) relative to the rule $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$, as a function of λ (log-scale for λ) for dimensions 2, 10, 30 and 100 (from top to bottom). **Right:** Plots of the values μ_{th} (solid lines with markers) giving the optimal μ relative to the quadratic approximation (9) together with extremity of range of μ values (shown with markers) giving convergence rates up to a precision of 0.2 from the optimal numerical value. The dimensions represented are 2, 10, 30 and 100 (from bottom to top).

a function of λ and for different dimensions. It can be seen that the optimal convergence rate is, for λ sufficiently large, linear as a function of $\ln(\lambda)$. This result is in agreement with the results in [16]. This figure shows also that the rule $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$ provides convergence rates very close to optimal ones. The curves in Fig. 2 (left) are smooth. However, to obtain the exact optimal values of μ (denoted μ_{opt}), we would need a very large number of Monte-Carlo samplings and (in parallel) a very small discretisation in σ^* that is not affordable. Therefore, we plotted in Fig. 2 (right), the ranges of μ values giving the optimal convergence rate up to a precision of 0.2, as a function of λ and for dimensions $d = 2, 10, 30$ and 100. Those ranges are called 0.2-confidence intervals in μ in the sequel. In the same graph, we plotted values of μ computed as the argmin of the polynomial approximation (9) that we denote μ_{th}. It can be seen that μ_{th} values are in the 0.2-confidence interval in μ. However, the values $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$ for $\lambda = 10^4$ and $d \in \{10, 30, 100\}$, are not in the 0.2-confidence interval in μ. In Figure 1, we compare, for $d = 30$, optimal convergence rates for different choices of μ, namely $\mu = 1$, $\lfloor \frac{\lambda}{4} \rfloor$ ([14]), $\lfloor \frac{\lambda}{2} \rfloor$ ([6]), $\min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$ ([9]) and the optimal rule (i.e., μ_{opt} values). We observe that for μ equal $\lfloor \frac{\lambda}{4} \rfloor$ and $\lfloor \frac{\lambda}{2} \rfloor$, the convergence rate does not scale linearly in $\ln(\lambda)$ and is thus sub-optimal. For $\mu = 1$ and $\min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$, the scaling is linear in $\ln(\lambda)$ and close to the optimal convergence rate for $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$.

Fig. 2 (right) suggests also that the values of μ_{th} vary as a function of $\ln(\lambda)$. Further investigations show that for λ large $\ln(\mu_{th}) = \alpha(d) \ln^2(\ln(\lambda)) + \beta(d)$ where $\alpha(d), \beta(d) > 0$ are some constants that have to be tuned for each dimension (see [12]).

6 Conclusion

In this paper, we have developed a complementary theoretical/numerical approach in order to investigate the isotropic $(\mu/\mu_w, \lambda)$-ES minimizing spherical functions. First, we have shown the log-linear convergence of this algorithm (provided good choice of parameters) with a scale-invariant adaptation rule for the step-size and we have expressed the convergence rate as the expectation of a given random variable. Second, thanks to the expression of the convergence rate, we have numerically computed, using Monte-Carlo simulations, optimal values for the choice of μ and $\frac{\sigma_n}{d_n}$ and their relative optimal convergence rates. We have investigated in particular large values of λ. Our results suggest that the optimal μ is monotonously increasing in λ as opposed to the rule $\mu = \min\{\lfloor \frac{\lambda}{4} \rfloor, d\}$ proposed in [9] but that however this latter rule gives a convergence rate close to the optimal one. We have confirmed as well that for the rules $\mu = \lfloor \frac{\lambda}{4} \rfloor$ and $\lfloor \frac{\lambda}{2} \rfloor$, the convergence rate does not scale linearly in $\ln(\lambda)$ and is thus sub-optimal.

Acknowledgments. The authors would like to thank Nikolaus Hansen for his advises on how to approach the problem tackled in the paper. This work received support by the French national research agency (ANR) within the COSINUS project ANR-08-COSI-007-12.

References

1. Schumer, M., Steiglitz, K.: Adaptive step size random search. IEEE Transactions on Automatic Control 13, 270–276 (1968)
2. Rechenberg, I.: Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution. Fromman-Hozlboog Verlag, Stuttgart (1973)
3. Schwefel, H.-P.: Collective phenomena in evolutionary systems. In: Checkland, P., Kiss, I. (eds.) Problems of Constancy and Change-The Complementarity of Systems Approaches to Complexity, Proc. of 31st Annual Meeting Int'l Soc. for General System Research, Budapest, vol. 2, pp. 1025–1033 (1987)
4. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
5. Auger, A., Hansen, N.: Reconsidering the progress rate theory for evolution strategies in finite dimensions. In: ACM Press (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 445–452 (2006)
6. Arnold, D.V.: Optimal weighted recombination. In: Foundations of Genetic Algorithms, vol. 8, pp. 215–237. Springer, Heidelberg (2005)
7. Jebalia, M., Auger, A., Liardet, P.: Log-linear convergence and optimal bounds for the (1+1)-ES. In: Monmarché, N., Talbi, E.-G., Collet, P., Schoenauer, M., Lutton, E. (eds.) EA 2007. LNCS, vol. 4926, pp. 207–218. Springer, Heidelberg (2008)
8. Teytaud, F., Teytaud, O.: On the parallel speed-up of Estimation of Multivariate Normal Algorithm and Evolution Strategies. In: Proceedings of EvoStar 2009, pp. 655–664 (2009)
9. Teytaud, F.: A new selection ratio for large population sizes. In: Proceedings of EvoStar 2010 (2010)
10. Bienvenüe, A., François, O.: Global convergence for evolution strategies in spherical problems: some simple proofs and difficulties. Th. Comp. Sc. 306(1-3), 269–289 (2003)
11. Jebalia, M., Auger, A., Hansen, N.: Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. Algorithmica (to appear 2010)
12. Jebalia, M., Auger, A.: Log-linear Convergence of the Scale-invariant $(\mu/\mu_w, \lambda)$-ES and Optimal μ for Intermediate Recombination for Large Population Sizes. Research Report n=°7275, INRIA (2010)

13. Auger, A.: Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible markov chains. Theoretical Computer Science 334(1-3), 35–69 (2005)
14. Beyer, H.-G.: The Theory of Evolution Strategies. Nat. Comp. Series. Springer, Heidelberg (2001)
15. 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)
16. Teytaud, O., Fournier, H.: Lower Bounds for Evolution Strategies Using VC-dimension. In: Rudolph, G., et al. (eds.) Proceedings of PPSN X, pp. 102–111. Springer, Heidelberg (2008)
17. Beyer, H.-G.: Toward a Theory of Evolution Strategies: On the Benefits of Sex - the (μ/μ,λ) Theory. Evolutionary Computation 3(1), 81–111 (1995)

Exploiting Overlap When Searching for Robust Optima

Johannes Kruisselbrink, Michael Emmerich, André Deutz, and Thomas Bäck

LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden
{jkruisse,emmerich,deutz,baeck}@liacs.nl
http://natcomp.liacs.nl

Abstract. This paper introduces a new view on fitness evaluation when searching for robust optima. It proposes to compare solutions in (successive) populations with respect to how they rank in robustness instead of aiming for accurate robustness estimates. This can be done by focusing on the non-overlapping parts of the regions of uncertainty of each pair of candidate solutions. An initial step toward a scheme implementing this view is made with the analysis and experiments on a simple scenario comparing two solutions on uniform input noise.

1 Introduction

Robust design optimization has received increasing attention in the evolutionary computation community ([7,4]) and one major issue is to search for solutions to optimization problems $f(\mathbf{x}) \rightarrow \min$, $\mathbf{x} \in \mathbb{R}^N$ which are robust to disturbances in the input variables. A common approach is to restate such problems as optimization of the expected objective function value given the distribution of the perturbations: $f_{\mathit{eff}} = \int_{-\infty}^{\infty} f(\mathbf{x}+\boldsymbol{\delta})\text{pdf}(\boldsymbol{\delta})d\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ denotes the perturbation of the input variables and $\text{pdf}(\boldsymbol{\delta})$ denotes its probability density function. The resulting function computes the so-called *effective fitness* and can be approximated by $\hat{f}_{\mathit{eff}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}+\boldsymbol{\delta}_i)$. However, this introduces noise due to approximation errors that eventually harms the precision of (evolutionary) optimization algorithms [2]. Moreover, increasing m only reduces the noise, but it does not eliminate it.

Previous work regarding the aim to find robust optima focused on finding the appropriate sampling schemes ([11,10,12,6]), reducing the number of function evaluations by means of smart sampling schemes ([5,6]), and using surrogate models to partially replace expensive function evaluations by cheaper ones (e.g. [8]). Also, correcting for noise and automatically increasing the population size (which has implicit resampling effects) has been considered [3].

Two interesting results were discussed in [6] and [8]. In [6] it was proposed to use the same perturbations for the evaluation of all solutions in a population. This leads to a reduction of errors in the comparisons, which is beneficial for the final quality of solutions. Another approach to reduce errors in comparisons was proposed in [8]: For each candidate solution a metamodel is trained and, for a

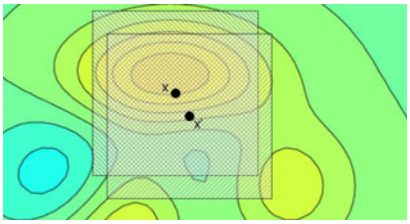


Fig. 1. Overlap sketch

given solution the ensemble of all metamodels is used in order to approximate its effective fitness. In both studies, the schemes in which all population points are considered for the evaluation of each candidate solution outperform the schemes that consider independent robustness approximations.

An important observation is that sampling regions of different candidate solutions in (successive) populations are often overlapping, particularly in later stages of the optimization where the population focuses on a single region of the search space. Hence, the evaluations made in overlapping regions can be used at the same time for evaluating the robustness of different candidate solutions. Moreover, in some cases, it is even possible to discard a large part of the sampling space when comparing solutions. For example, in Figure 1, where solutions $\mathbf{x}$ and $\mathbf{x}'$ are compared given a uniform distribution of the input noise, it suffices to sample the non-intersecting regions, because the contribution of the intersecting region to the effective fitness is the same for both solutions.

Instead of trying to acquire a robustness estimate for each candidate solution separately, this paper proposes to compare solutions in (successive) populations with respect to how they relate in robustness. The overlap of the regions of uncertainty will play a key role in this. This paper considers the simple case of Figure 1 with the intent to investigate if, and how, overlap can be exploited when searching for robust solutions.

This paper is structured as follows: Section 2 discusses the effect of overlap on the comparison of two solutions. Section 3 relates the distance between two solutions to the overlap area. Section 4 proposes two sampling schemes that avoid the overlap region. Section 5 presents a symmetry property that can be exploited. Section 6 presents a demonstration of how to adopt the ideas presented in this paper. Section 7 closes with conclusions and an outlook.

2 Comparing Two Solutions with Uniform Noise

This paper will focus on the special case of comparing two candidate solutions $\mathbf{x}$ and $\mathbf{x}'$ on their effective fitness based on a uniform perturbation $\boldsymbol{\delta} \sim \mathbf{U}(-\mathbf{l}, \mathbf{l})$. By normalizing the search space we can transform the sampling intervals of the independent input variables such that they have the same width l.

Consider the scenario depicted in Figure 2. Two solutions (the two black points) around which square regions are drawn indicating the regions of

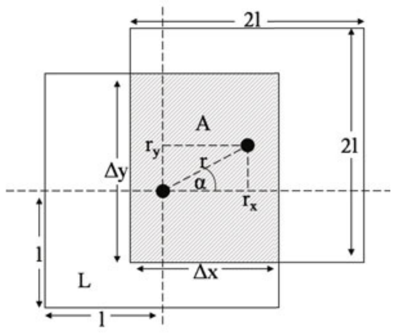


Fig. 2. A 2D sketch of the overlap of the regions $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$

uncertainty $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$. In the case of uniform noise, the intersection region A of $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$ has the same contribution to the effective fitness for both solutions. I.e.,

$$f_{eff_{\mathbf{x}}} = \int_{\tilde{\mathbf{x}} \in L_{\mathbf{x}} \setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} + \int_{\tilde{\mathbf{x}} \in A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{1}$$

$$f_{eff_{\mathbf{x}'}} = \int_{\tilde{\mathbf{x}} \in L_{\mathbf{x}'} \setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} + \int_{\tilde{\mathbf{x}} \in A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{2}$$

$$f_{eff_{\mathbf{x}}} - f_{eff_{\mathbf{x}'}} = \int_{\tilde{\mathbf{x}} \in L_{\mathbf{x}} \setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}} - \int_{\tilde{\mathbf{x}} \in L_{\mathbf{x}'} \setminus A} f(\tilde{\mathbf{x}})\ \mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}})\ d\tilde{\mathbf{x}}, \tag{3}$$

where $\mathrm{p}_{\mathbf{x}}(\tilde{\mathbf{x}}) \sim \mathrm{pdf}(\mathbf{x} + \boldsymbol{\delta})$ and $\mathrm{p}_{\mathbf{x}'}(\tilde{\mathbf{x}}) \sim \mathrm{pdf}(\mathbf{x}' + \boldsymbol{\delta})$. Hence, for absolute comparisons, the region A offers no relevant information. Moreover, when $\mathbf{x}$ and $\mathbf{x}'$ are located close to each other, A will be large compared to $L_{\mathbf{x}} \setminus A$ and $L_{\mathbf{x}'} \setminus A$ and for an evaluation method based on sampling in $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$, the probability of sampling within $L_{\mathbf{x}} \setminus A$ and $L_{\mathbf{x}'} \setminus A$ will be small. Given the surface area (or volume in higher dimensions) of A, the probability that one uniformly drawn random sample in $L_{\mathbf{x}}$ hits $L_{\mathbf{x}} \setminus A$ is

$$P(\text{sample not in } A \mid \text{sample in } L_{\mathbf{x}}) = 1 - \frac{A}{L_{\mathbf{x}}} = 1 - \frac{A}{2^N l^N}. \tag{4}$$

For a pure Monte-Carlo sampling scheme, this leads to the following expected number of samples needed to obtain one sample in $L_{\mathbf{x}}$ that is not in A:

$$E(X) = \sum_{n=1}^{\infty} P(X = n)n = \frac{1}{1 - \frac{A}{L_{\mathbf{x}}}} = \frac{2^N l^N}{2^N l^N - A}. \tag{5}$$

Here X is the discrete random variable for the number of samples until the first sample in $L_{\mathbf{x}} \setminus A$ is obtained. Obviously, the same holds for $L_{\mathbf{x}'}$, and, provided that there is an overlap in the regions of uncertainty, A can be computed as:

$$A = \prod_{i=1}^{N} 2l - |x_i - x_i'|. \tag{6}$$

3 Relating Distance to Overlap

A crucial question when designing adaptive optimization algorithms is how the distance between two points is related to the size of the intersection region. In 2D, the following expression can be derived for A, given that $\mathbf{x}$ and $\mathbf{x}'$ lie at a distance r from each other, making an angle α w.r.t. the x-axis:

$$A = \begin{cases} (2l - r_x) \cdot (2l - r_y) & \text{, if } r_x \leq 2l \wedge r_y \leq 2l \\ 0 & \text{, otherwise} \end{cases} . \tag{7}$$

Here, $r_x = |r \cos \alpha|$ and $r_y = |r \sin \alpha|$. From this, and assuming a periodicity of π for α (i.e., $\alpha \in [0, \pi]$), an expression can be derived for the expected number of samples in $L_{\mathbf{x}}$, m, needed in order to hit the area $L_{\mathbf{x}} \backslash A$ at least c times:

$$m = \begin{cases} 4cl^2 / \left(2lr\left(\cos\alpha + \sin\alpha\right) - r^2 \cos\alpha \sin\alpha\right) & \text{, if } r_x \leq 2l \wedge r_y \leq 2l \\ c & \text{, otherwise} \end{cases} . \tag{8}$$

By expressing r in terms of l, substituting $r = kl$, this can be simplified to:

$$m = \begin{cases} 4c / \left(2k\left(\cos\alpha + \sin\alpha\right) - k^2 \cos\alpha \sin\alpha\right) & \text{, } k \cos\alpha \leq 2 \wedge k \sin\alpha \leq 2 \\ c & \text{, otherwise} \end{cases} . \tag{9}$$

The derivation above is still dependent on the angle α between $\mathbf{x}$ and $\mathbf{x}'$. It would be desirable to have an approximation independent of α. This yields a general approximation for the required number of samples m needed for two individuals at a distance r (still using $r = lk$, i.e., $k = \frac{r}{l}$) to have at least c samples in the regions $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$ respectively. For this, we look at the upper bound of A w.r.t. α and note that for A to be maximized, $\alpha = 0$ or $\alpha = \frac{\pi}{4}$:

$$\begin{aligned} A &= \begin{cases} \max\{4l^2 - 2kl^2, 4l^2 - 2\sqrt{2}kl^2 + \frac{1}{2}k^2l^2\} & \text{, if } k \leq 2\sqrt{2} \\ 0 & \text{, otherwise} \end{cases} \\ &= \begin{cases} 4l^2 - 2\sqrt{2}kl^2 + \frac{1}{2}k^2l^2 & \text{, if } k \leq 4\left(\sqrt{2} - 1\right) \\ 4l^2 - 2kl^2 & \text{, if } 4\left(\sqrt{2} - 1\right) < k \leq 2\sqrt{2} \\ 0 & \text{, otherwise} \end{cases} . \end{aligned} \tag{10}$$

This upper bound of A can be used to approximate (the upper bound of) m:

$$m = \begin{cases} 8c / \left(4\sqrt{2}k - k^2\right) & \text{, if } k \leq 4\left(\sqrt{2} - 1\right) \\ 2c/k & \text{, if } 4\left(\sqrt{2} - 1\right) < k \leq 2\sqrt{2} \\ 0 & \text{, otherwise} \end{cases} . \tag{11}$$

For general ND cases we assume that, similar to the 2D case, A is maximized in the cases equivalent to the 2D cases of $\alpha = 0$ and $\alpha = \frac{\pi}{4}$. Following this, and again using the substitution $r = kl$, we obtain:

$$A = \begin{cases} 2^N l^N \max\left\{\left(1 - \frac{1}{2}k\right), \left(1 - \frac{1}{2\sqrt{N}}k\right)^N\right\} & \text{, if } k \leq 2\sqrt{N} \\ 0 & \text{, otherwise} \end{cases} , \tag{12}$$

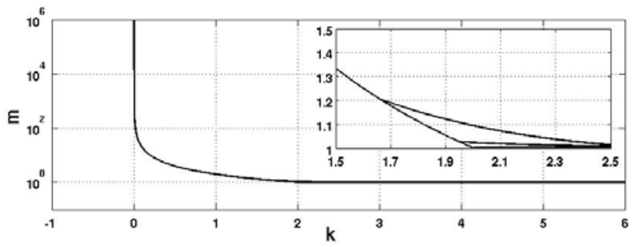


Fig. 3. m versus k for $N = 10^0, 10^1, \ldots, 10^6$ with a zoom on the interval $[1\frac{1}{2}, 2\frac{1}{2}]$

which we can use to obtain an expression for the expected number of samples in $L_{\mathbf{x}}$ needed to hit $L_{\mathbf{x}} \backslash A$ at least c times:

$$m = \begin{cases} c \max\left\{\frac{2}{k}, 1/\left(1 - \left(1 - \frac{1}{2\sqrt{N}}k\right)^N\right)\right\} & \text{, if } k \leq 2\sqrt{N} \\ c & \text{, otherwise} \end{cases} . \tag{13}$$

It is clear that when using normal sampling approaches, many samples are practically wasted when the distance between two solutions becomes small. Figure 3 shows the number of required samples m needed to hit $L_{\mathbf{x}} \backslash A$ at least once versus k for $N = 10^0, 10^1, \ldots, 10^6$. Interestingly, the plots are not much different for all values of N. For $k \lesssim 1.7$ the term $\frac{2}{k}$ becomes the determining factor and the other term leads to $m \approx 1$, even for $N = 10^6$. Hence, the following rule-of-thumb can be used to indicate the growth of m relative to k:

$$m = \begin{cases} \frac{2c}{k} & \text{, if } k \leq 1.7 \\ c & \text{, otherwise} \end{cases} . \tag{14}$$

4 Two Sampling Schemes

A straightforward rejection-based sampling procedure that uniformly samples in $L_{\mathbf{x}} \backslash A$ is described by the algorithm of Figure 4. It maintains a set of samples $\mathbf{X}$ for the design point $\mathbf{x}$, given $L_{\mathbf{x}}$ and $L_{\mathbf{x}'}$ (i.e., assuming the existence of the other point $\mathbf{x}'$), and continuously takes samples in the region $L_{\mathbf{x}}$ until it has found m samples that are not in $L_{\mathbf{x}'}$.

We can use (13) to obtain an estimate for the expected number of iterations of the algorithm of Figure 4 relative to the distance r between the points $\mathbf{x}$ and $\mathbf{x}'$. For optimization approaches such as the (1 + 1)-ES, this running time is directly coupled to the stepsize σ, because $E[r] = \sqrt{N}\sigma$. Hence, using such a sampling scheme becomes infeasible once the stepsize becomes very small.

Also a different sampling scheme can be considered, shown in Figure 5, that targets the region $L_{\mathbf{x}} \backslash A$ directly. For this, the region $L_{\mathbf{x}}$ is partitioned into 2^N regions as depicted in the right picture of Figure 5. Each region $A_{\mathbf{a}}$ is identified by means of a vector $\mathbf{a} \in \{0, 1\}$, where $a_i = 0$ identifies the non-overlapping interval

```
input: $L_{\mathbf{x}} = [\mathbf{x} - \mathbf{l}, \mathbf{x} + \mathbf{l}]$, $L_{\mathbf{x}'} = [\mathbf{x}' - \mathbf{l}, \mathbf{x}' + \mathbf{l}]$
1: $\mathbf{X} \leftarrow \emptyset$
2: while $|\mathbf{X}| < m$ do
3:     $\mathbf{x}_s \sim U(L_{\mathbf{x}})$
4:     if $\mathbf{x} \notin L_{\mathbf{x}'}$ then
5:         $\mathbf{X} \leftarrow \mathbf{X} \bigcup \{\mathbf{x}_s\}$
6:     end if
7: end while
8: return X
```

Fig. 4. Rejection-based uniform sampling in $L_{\mathbf{x}} \backslash L_{\mathbf{x}'}$

in dimension i, and $a_i = 1$ identifies the overlapping interval in dimension i. Using this definition, the volume of $A_{\mathbf{a}}$ is computed as:

$$A_{\mathbf{a}} = \prod_{i=1}^{N} (2la_i + (1 - 2a_i) \cdot |x_i - x'_i|) \text{ , } \mathbf{a} \in \{0,1\}^N, \tag{15}$$

which can be used to determine the sampling probability for each region as:

$$P(A_{\mathbf{a}}) = \frac{A_{\mathbf{a}}}{L_{\mathbf{x}} - A_{\mathbf{1}}} \text{ , } \mathbf{a} \in \{0,1\}^N \backslash \{1\}^N. \tag{16}$$

Note that $A_{\mathbf{1}}$ is the region of full overlap, i.e., A. The algorithm of Figure 5 uses this partitioning to directly sample uniformly in the region $L_{\mathbf{x}} \backslash A$. For each sample it will first select a box $A_{\mathbf{a}}$ using the probabilities (16) and then take a sample $\mathbf{x}_s$ uniformly from that box. This scheme has a complexity of $O(2^N)$ (i.e., proportional to the number of boxes $A_{\mathbf{a}}$) making it only practical for search spaces of smaller dimension sizes ($N \lesssim 10$). However, it is not dependent on the distance between $\mathbf{x}$ and $\mathbf{x}'$ and has a constant expected running time over an optimization loop of e.g. a $(1+1)$-ES.

```
input: $L_{\mathbf{x}} = [\mathbf{x} - \mathbf{l}, \mathbf{x} + \mathbf{l}]$, $L_{\mathbf{x}'} = [\mathbf{x}' - \mathbf{l}, \mathbf{x}' + \mathbf{l}]$
1: $\mathbf{X} \leftarrow \emptyset$
2: for $i = 1$ to $m$ do
3:     select $A_{\mathbf{a}}$, $\mathbf{a} \in \{0,1\}^N \backslash \{1\}^N$ with probability
   $P(A_{\mathbf{a}}) = A_{\mathbf{a}} \backslash (L_{\mathbf{x}} - A)$
4:     $\mathbf{x}_s \sim U(A_{\mathbf{a}})$
5:     $\mathbf{X} \leftarrow \mathbf{X} \bigcup \{\mathbf{x}_s\}$
6: end for
7: return X
```

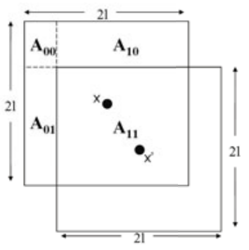


Fig. 5. Partitioning the non-overlap regions for uniform sampling in $L_{\mathbf{x}} \backslash L_{\mathbf{x}'}$

5 Exploiting Symmetry

Besides the fact that we can either avoid the region A in case of uniform noise, or at least reuse the samples for the evaluation of $\mathbf{x}$ and $\mathbf{x}'$ in case of other noise distributions, there is also a symmetry in the regions $L_{\mathbf{x}}\backslash A$ and $L_{\mathbf{x}'}\backslash A$. Given a perturbation $\mathbf{x}_d \sim U(-\mathbf{l}, \mathbf{l})$ we note that:

$$\mathbf{x} + \mathbf{x}_d \in L_{\mathbf{x}}\backslash A \Leftrightarrow \mathbf{x}' - \mathbf{x}_d \in L_{\mathbf{x}'}\backslash A. \tag{17}$$

Hence, when using the sampling algorithms of Figure 4 and 5, it is only required to run them once for every pair $\mathbf{x}$ and $\mathbf{x}'$, because we can deduce a set $\mathbf{X}'$ from $\mathbf{X}$. Moreover, as shown in [6], it is advantageous to use the same perturbations for each individual in a population, which is possibly due to the absence of an inherent bias toward selecting the solution with the smallest perturbation distances (this effect is described in [3]).

6 A Simple Demonstration for the (1 + 1)-ES

To demonstrate how the previous considerations could be integrated into practical optimization schemes, as an example we study a $(1+1)$-ES with the 1/5th success rule for stepsize adaptation [9] and implement two versions which incorporate a robustness evaluation scheme, shown in Figure 6. The $(1+1)$-ES, being a simple algorithm useful for theoretical purposes only, lends itself well for incorporation of a comparison scheme that follows the previous considerations.

The first scheme (left) is a benchmark scheme which uses the straightforward Monte-Carlo approach. In each iteration a set $\mathbf{X}_d$ of m perturbations is drawn from $[-\mathbf{l}, \mathbf{l}]$. The set $\mathbf{X}_d$ is used to generate the sets $\mathbf{X}_o$ and $\mathbf{X}_p$ which are the perturbations of the parent and the offspring used to obtain the robustness approximations (note that the parent is re-evaluated every generation).

The second scheme (right) takes for the offspring a set of m samples from $L_{\mathbf{x}_o}\backslash A$, and uses the perturbations of $\mathbf{X}_o$ to generate $\mathbf{X}_p$ by exploiting the symmetry (discussed in Section 5). The sets $\mathbf{X}_o$ and $\mathbf{X}_p$, which now contain only samples in $L_{\mathbf{x}_o}\backslash A$ and $L_{\mathbf{x}_p}\backslash A$ respectively, are used to compare the parent and the offspring (note that we avoid the use of $\hat{f}_{\mathit{eff}}$, as we no longer approximate the effective fitness, but rather compare the parent and the offspring).

It should be noted that Latin Hypercube Sampling yields better results than Monte-Carlo sampling [6]. Although both schemes could be adapted for this, Monte-Carlo sampling is used here for the sake of the simplicity of the example. Also, a note of caution is in order regarding the performance of the $(1+1)$-ES on noisy fitness functions. As noted in [1] and [2], the $(1+1)$-ES can stagnate or even show divergent behavior when the noise level is sufficiently high. As both schemes are still based on sampling, noise is inevitable. Moreover, as shown in Section 3, for the normal sampling approach, the signal-to-noise ratio is related to the stepsize and it can be expected that this method will stagnate at a certain point in time. The question is: (when) will the overlap avoiding scheme stagnate?

```
1: initialize: $t \leftarrow 0$, $\mathbf{x}_p \leftarrow \mathbf{x}^{(\text{init})}$, $\sigma \leftarrow \sigma^{(\text{init})}$, $s \leftarrow 0$, $n \leftarrow N$, $c \leftarrow 0.85^{1/5n}$
2: while not terminate do
3:     $\mathbf{z}_o \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4:     $\mathbf{x}_o \leftarrow \mathbf{x}_p + \sigma \mathbf{z}_o$
5:     $\mathbf{X}_d \leftarrow m$ samples from $[-\mathbf{l}, \mathbf{l}]$
6:     $\mathbf{X}_o \leftarrow \bigcup_{\mathbf{x}_s \in \mathbf{X}_d} \{\mathbf{x}_o + \mathbf{x}_s\}$
7:     $\mathbf{X}_p \leftarrow \bigcup_{\mathbf{x}_s \in \mathbf{X}_d} \{\mathbf{x}_p + \mathbf{x}_s\}$
8:     $\hat{f}_{\mathit{eff}_o} \leftarrow \frac{1}{m} \sum_{\mathbf{x}_s \in \mathbf{X}_o} f(\mathbf{x}_s)$
9:     $\hat{f}_{\mathit{eff}_p} \leftarrow \frac{1}{m} \sum_{\mathbf{x}_s \in \mathbf{X}_p} f(\mathbf{x}_s)$
10:    if $\hat{f}_{\mathit{eff}_o} \leq \hat{f}_{\mathit{eff}_p}$ then
11:        $\mathbf{x}_p \leftarrow \mathbf{x}_o$
12:        $s \leftarrow s + 1$
13:    end if
14:    if $(t \mod n == 0)$ then
15:        $p_s \leftarrow s/n$
16:        $\sigma \leftarrow \begin{cases} \sigma/c & , p_s < 1/5 \\ \sigma \cdot c & , p_s > 1/5 \\ \sigma & , p_s = 1/5 \end{cases}$
17:        $s \leftarrow 0$
18:    end if
19:    $t \leftarrow t + 1$
20: end while
```

```
1: initialize: $t \leftarrow 0$, $\mathbf{x}_p \leftarrow \mathbf{x}^{(\text{init})}$, $\sigma \leftarrow \sigma^{(\text{init})}$, $s \leftarrow 0$, $n \leftarrow 5$, $c \leftarrow 0.85^{1/5n}$
2: while not terminate do
3:     $\mathbf{z}_o \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4:     $\mathbf{x}_o \leftarrow \mathbf{x}_p + \sigma \mathbf{z}_o$
5:     $\mathbf{X}_o \leftarrow m$ samples from $L_{\mathbf{x}_o} \backslash A$
6:     $\mathbf{X}_d \leftarrow \bigcup_{\mathbf{x}_s \in \mathbf{X}_o} \{\mathbf{x}_s - \mathbf{x}_o\}$
7:     $\mathbf{X}_p \leftarrow \bigcup_{\mathbf{x}_s \in \mathbf{X}_d} \{\mathbf{x}_p - \mathbf{x}_d\}$
8:     $\hat{f}_o \leftarrow \frac{1}{m} \sum_{\mathbf{x}_s \in \mathbf{X}_o} f(\mathbf{x}_s)$
9:     $\hat{f}_p \leftarrow \frac{1}{m} \sum_{\mathbf{x}_s \in \mathbf{X}_p} f(\mathbf{x}_s)$
10:    if $\hat{f}_o \leq \hat{f}_p$ then
11:        $\mathbf{x}_p \leftarrow \mathbf{x}_o$
12:        $s \leftarrow s + 1$
13:    end if
14:    if $(t \mod n == 0)$ then
15:        $p_s \leftarrow s/n$
16:        $\sigma \leftarrow \begin{cases} \sigma/c & , p_s < 1/5 \\ \sigma \cdot c & , p_s > 1/5 \\ \sigma & , p_s = 1/5 \end{cases}$
17:        $s \leftarrow 0$
18:    end if
19:    $t \leftarrow t + 1$
20: end while
```

Fig. 6. Two (1 + 1)-ES schemes for finding robust optima using normal sampling in $L_{\mathbf{x}_o}$ and $L_{\mathbf{x}_p}$ (left) and sampling focused on $L_{\mathbf{x}_o} \backslash A$ and $L_{\mathbf{x}_p} \backslash A$ (right)

The experiments are performed on the sphere problem $f(\mathbf{x}) = \sum_{i=1}^{N} (x_i + \delta_i)^2$, like in [3]. The settings are: $N = 10$ (i.e., 10D), $\mathbf{x} \in [-10, 10]^N$, and $\boldsymbol{\delta} \sim U(-\mathbf{1}, \mathbf{1})$. A high evaluation budget of 2×10^6 is used and both algorithms use $m = 2N$. As the algorithmic setting is only a demonstration of the use of the concept introduced in this paper, we will only show the results of one run. Figure 7 shows the results.

Whereas the approach implementing the normal sampling stagnates after ca. $5 \cdot 10^5$ evaluations, there is no sign of stagnation for the approach implementing the focused sampling. Hence, although it still uses $m = 20$ samples for each evaluation, the focused sampling approach remains making progress. Supported also by the stepsize plots, these empirical results suggest linear convergence.

The error plots show the frequency of false negatives and false positives versus the number of generations (these frequencies are computed over a window of 1000 generations). For both approaches, the error frequencies stay at the same levels at a certain point in time. For the normal sampling approach, this can be related to the stagnation of the stepsize (i.e., the error rate is coupled to the noise ratio, which is directly coupled to the stepsize). However, for the focused resampling approach the error rates stay at the same levels even with the stepsize decreasing.

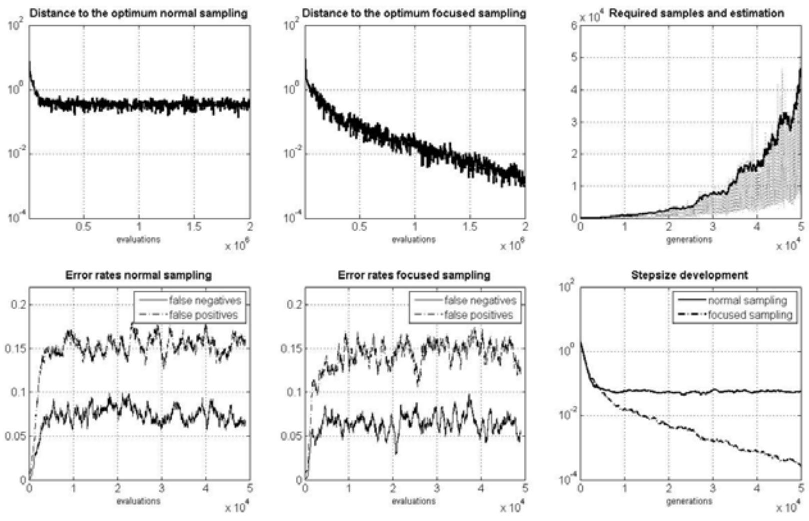


Fig. 7. Left column: distance to the optimum and ordering error frequency of the normal sampling approach. Middle column: distance to the optimum and ordering error frequency of the focused sampling approach. Top right: the required number of samples and an upper bound estimator for the required number of samples of the focused sampling run. Bottom right: the stepsize development for both approaches.

Finally, we show for the focused sampling method the number of samples that were required in order to obtain m samples in $L_{\mathbf{x}} \backslash A$. Note that the implementation used for these experiments uses the simple rejection method of the algorithm of Figure 4. The thick solid line shows for every generation the expected upper bound of the number of samples, computed using the simple rule-of-thumb of (14), with $r = \sigma\sqrt{N}$. This plot shows how this rule-of-thumb accurately determines the upper bound for the required number of samples, but also that in many cases fewer samples suffice.

7 Conclusions and Outlook

This paper has introduced a new view on robustness evaluation where the focus is not on trying to obtain accurate robustness estimates for individual candidate solutions, but rather to compare the solutions of (successive) populations with respect to robustness. It has shown how the distance between two solutions relates to the amount of redundant sampling when dealing with uniform input noise. Also, two sampling schemes have been introduced that target the regions where there is no overlap between and a proof of concept was given.

The experiments on the (1+1)-ES show how the proposed idea can successfully be integrated in common optimization algorithms. A topic for future research is to also include this concept in population based schemes that work with sets of candidate solutions rather than two. An implementation for tournament selection can easily be derived, but for $(\mu \overset{+}{,} \lambda)$-selection, more sophisticated schemes are required. For $(\mu \overset{+}{,} \lambda)$-schemes, samples of overlapping regions can at least be reused when evaluating candidate solutions, but specifically targeting non-overlapping regions will become computationally more expensive.

Another question is to what extent this sampling strategy can be adopted in cases of other input noise distributions. In these cases, it might not be possible or desirable to reject the region of overlap. It is a topic for further study to compare how the overlapping and non-overlapping regions relate to each other when determining the relative fitness differences between candidate solutions.

Acknowledgments. J. Kruisselbrink acknowledges financial support of the Netherlands Organization for Scientific Research (NWO); dossier no. 612.066.618.

References

1. Beyer, H.-G.: Towards a theory of 'evolution strategies'. some asymptotical results from the (1,+lambda)-theory. Evolutionary Computation 1, 165–188 (1993)
2. Beyer, H.-G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering 186(2-4), 239–267 (2000)
3. Beyer, H.-G., Sendhoff, B.: Evolution strategies for robust optimization. In: IEEE Congress on Evolutionary Computation (CEC 2006), pp. 1346–1353 (2006)
4. Beyer, H.-G., Sendhoff, B.: Robust optimization - a comprehensive survey. Computer Methods in Applied Mechanics and Engineering 196(33-34), 3190–3218 (2007)
5. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Parallel Problem Solving from Nature (PPSN V), pp. 119–128 (1998)
6. Branke, J.: Reducing the sampling variance when searching for robust solutions. In: Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 235–242 (2001)
7. Jin, Y., Branke, J.: Evolutionary optimizationin uncertain environments - a survey. IEEE Transaction on Evolutionary Computation 9(3), 303–317 (2005)
8. Paenke, I., Branke, J., Jin, Y.: Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Transactions on Evolutionary Computation 10(4), 405–420 (2006)
9. Schwefel, H.-P.: Evolution and Optimum Seeking: The Sixth Generation. John Wiley & Sons, Inc., Chichester (1993)
10. Tsutsui, S.: A comparative study on the effects of adding perturbations to phenotypic parameters in genetic algorithms with a robust solution searching scheme. In: IEEE Systems, Man, and Cybernetics Conference (SMC 1999), pp. 585–591 (1999)
11. Tsutsui, S., Ghosh, A.: Genetic algorithms with a robust solution searching scheme. IEEE Transactions on Evolutionary Computation 1(3), 201–208 (1997)
12. Tsutsui, S., Ghosh, A.: Effects of adding perturbations to phenotypic parameters in genetic algorithms for searching robust solutions. In: Advances in Evolutionary Computing: Theory and Applications, pp. 351–365 (2003)

Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis

Olaf Mersmann, Mike Preuss, and Heike Trautmann

TU Dortmund University, Dortmund, Germany
mike.preuss@cs.uni-dortmund.de
{trautmann,olafm}@statistik.tu-dortmund.de

Abstract. We present methods to answer two basic questions that arise when benchmarking optimization algorithms. The first one is: which algorithm is the 'best' one? and the second one: which algorithm should I use for my real world problem? Both are connected and neither is easy to answer. We present methods which can be used to analyse the raw data of a benchmark experiment and derive some insight regarding the answers to these questions. We employ the presented methods to analyse the BBOB'09 benchmark results and present some initial findings.

Keywords: evolutionary optimization, benchmarking, BBOB test set, multidimensional scaling, consensus ranking.

1 Introduction

The last years have seen several competitions for optimization algorithms at *evolutionary computation* (EC) conferences, possibly starting with the CEC'05 [11], and currently most notably continued with the black-box optimization algorithm benchmarking (BBOB) competitions at GECCO 2009 and 2010 [2]. However, this bears two main questions: a) Given a number of comparison results on different functions, what is the 'best' algorithm? and b) How can we transfer benchmarking results onto real-world situations?

To answer the first question, we turn to existing benchmarking theory, and especially to consensus ranking procedures. Answering the second question is surely harder. While it is relatively easy to control the settings of a benchmark experiment and to enforce every problem property we can possibly imagine, not much is usually known about a real-world problem we may have to deal with. The only possible solution for achieving a good algorithm-problem matching may be to extract meaningful high-level empirical properties and approach the matching from two sides: a) find out which algorithms perform especially well on certain property combinations and b), develop ways to cheaply and automatically extract problem properties from a concrete problem instance. The latter may be termed *Exploratory Landscape Analysis* (ELA) and is our long-term objective which is not explicitly attempted in this work. However, in order to achieve any progress in this directions, we need a good set of properties and to resolve the first issue of detecting and evaluating algorithm-property dependencies.

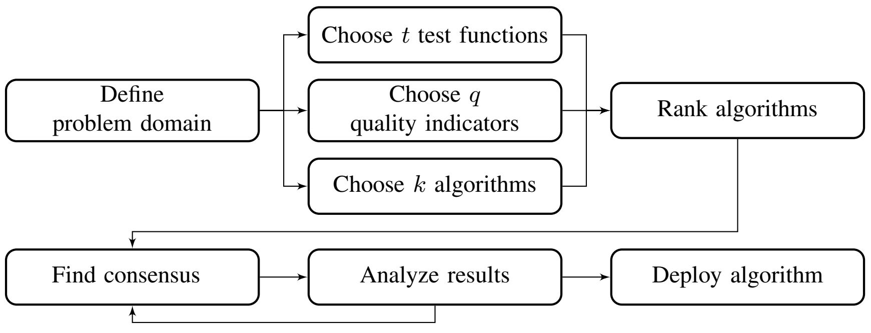


Fig. 1. Flow chart describing the steps involved in a benchmark experiment

After introducing benchmarking theory in Sec. 2 we will focus on the BBOB'09 test in Sec. 3 which is grouped into 5 categories using predefined properties. We will analyse if the measured performance of the benchmarked algorithms is in line with this grouping and will include additional properties into the analysis. Furthermore, we will explicitly make the distinction of low (2 and 3) and high (5-20) dimensions[1] and analyse how similar or different the algorithms behave with respect to these groups. Conclusions are given in Sec. 4.

2 Benchmarking Theory

The outcome of benchmarking experiments and competitions strongly depends on chosen performance measures and ranking procedures ([7],[8]). The general setup of a benchmark experiment is shown in Fig. 1.

Initially a *problem domain* needs to be defined. This step is crucial since it restricts the domain to which any conclusions made in later steps can possibly generalize. Next, t test functions with a known global optimum value or a known bound that should be achieved are chosen. Finding good test functions is a hard problem since they should be distributed in a 'uniform' fashion in the space of all possible functions from the problem domain. To judge the performance of an optimization result, q (ideally independent) quality indicators are chosen. Finally, a set of candidate optimization procedures needs to be determined.

The next step is to determine the number of independent runs of each of the k algorithms on the t test functions with respect to the trade-off between speed and accuracy. In practice 10 to 25 repetitions are a good rule of thumb. The result of this are $t \times q \times k$ quality indicator estimates. Using these, individual rankings of the algorithm for each quality indicator and test function will be constructed.

2.1 Individual Ranking

Benchmarking theory is mainly based on the theoretical framework of relations and orders [4] from which a formal definition of a *ranking* of a set of items can be derived

[1] The BBOB'09 data for 40 dimensions is not complete and therefore discarded.

[8]. We will use $\succ$, where $a \succ b$ reads as "a better than or equal to b", to denote this ranking and the underlying relation.

Initially, without loss of generality, we will consider the case of a fixed test function f and quality criterion I to be maximized. If we only had two algorithms, a_1 and a_2, then we could define $\succ$ by saying $a_1 \succ a_2$ (a_1 is better than or equal to a_2) if $I(a_1(f)) \geq I(a_2(f))$. Generalizing this result to a higher number of algorithms is straight forward. We use the order induced by I on the algorithms as our ranking. The main disadvantage of our simple definition of $\succ$ is that we must *estimate* the value of a quality criterion from several runs of the optimization algorithm. This means that $I(a, f)$ is a random variable whose distribution is unknown. One way to deal with this is to use classical statistical hypothesis tests to define $\succ$. Details of this procedure are given in [7] and [8]. The main obstacle is that we need $\succ$ to be transitive and antisymmetric in order for it to define a meaningful ordering.

2.2 Consensus Ranking

In case the *best* algorithm out of a given set should be determined the problem of building a consensus of a number of individual rankings of the considered algorithm arises. Ideally, a consensus should be non-dictatorial, universal, Pareto efficient and fulfill the *Independence of irrelevant alternatives* (IIA) criterion as well as the majority criterion (see [7] or [8] for formal definitions). Unfortunately, all criteria cannot be met simultaneously. Thus, consensus approaches yield different results with respect to the criteria chosen to be fulfilled.

Generally, we can differentiate between positional and optimization based methods. It can be shown that the Borda count method [1], as one of the oldest consensus methods, is optimal under all positional consensus methods [9]. An algorithm is assigned one point for each algorithm that it weakly dominates, i.e. the algorithms that are not better than the algorithm considered. The Borda score results by taking the sum of these values. Optimization based methods transform the consensus ranking task into an optimization problem based on a suitable distance measure between the individual rankings. The consensus ranking comes out as the ranking which minimizes the median or mean distance to all individual rankings (*SD approach*, [6]). Advantages and drawbacks of the introduced consensus methods are summarized in [10].

3 Benchmarking Results for the BBOB09-Testset

3.1 Suggested Problem Properties

The BBOB'09 test set is built according to problem properties, a) separable problems, b) low or moderate conditioned problems, c) high conditioned and unimodal problems, d) multi-modal problems with adequate global structure, and e) multi-modal problems with weak global structure. We suggest to add more properties applied to the BBOB'09 test set in Table 1. It gets evident that some property values are sparsely populated, e.g. only one test function has plateaus.

Multi-modality refers to the number of local optima of a problem. In practical applications, many problems are not unimodal (convex) as favoured by most classical optimization algorithms.

Global structure is what remains after deleting all non-optimal points. For Rastrigins problem, we obtain a perforated parabola which is unimodal. Problems without global basin structure are more difficult because one virtually needs to look in every corner.

Separability means a problem may be partitioned into subproblems which are then of lower dimensionality and should be considerably easier to solve. However, for an unknown problem, information about its separability may be scarce.

Variable scaling can make a problem behave differently in each dimension. It can be essential to perform small steps in some dimensions, and large ones in others, which is due to the non-spherical form of basins of attraction. Note that scaling may differ between different basins of attraction.

Search space homogeneity refers to a search space without phase transitions. Its overall appearance is similar in different search space areas. Most benchmark problems are of this type.

Basin size homogeneity means the size relation (largest to smallest) of all basins of attraction (e.g. [12] postulated that size differences influence problem hardness).

Global to local optima contrast refers to the difference between global and local peaks in comparison to the average fitness level of a problem. It thus determines if very good peaks are easily recognized as such.

Plateaus can make the life of optimization algorithms a lot harder as they do not provide any information about good directions to turn to. However, in the BBOB'09 test set, this property is largely unused.

3.2 Algorithm Analysis

Initially, a ranking for each test function / dimension combination is calculated using the *expected running time* (ERT, [2]) which is shown in Fig. 2 and 3. It is evident that there is a change in the general performance of the algorithms going from three to five dimensions. Especially in the parallel coordinate plot we recognize a deterioration of performance for some algorithms as the dimension rises.

Looking at the distribution of these ranks separately for the two and three as well as the 5 to 20 dimensional functions as shown in Fig. 4, we see two very different consensus rankings of the algorithms. The order of the algorithms was chosen by decreasing mean rank, this coincides with the Borda consensus for the two dimension groups. In lower dimensions the Nelder-Mead type algorithms perform best while only ranking in the middle for higher dimensional problems. The same behaviour can be observed for some other classical algorithms such as the Rosenbrock procedure. On the other hand, BFGS performs better for higher dimensional problems. Here, the effort invested for estimating the gradient obviously pay off. Generally, the order of the algorithms gets more stable for higher dimensions, so that the differences between dimensions 5/10/20 are much smaller than the ones between 2/3/5 (see Fig. 2). We therefore do not present a consensus ranking over all dimensions.

3.3 Function-Set Analysis

We have seen that the algorithms perform quite differently in different dimensions. Can we expect similar behaviour for different classes of functions from the test function set?

Table 1. Classification of the noiseless functions based on their properties (multi-modality, global structure, separability, variable scaling, homogeneity, basin-sizes, global-local contrast, plateaus). Predefined groups are separated by horizontal lines.

Function	multim.	gl.-struc.	separ.	scaling	homog.	basins	gl.-loc.	plat.
1: Sphere	none	none	high	none	high	none	none	none
2: Ellipsoidal separable	none	none	high	high	high	none	none	none
3: Rastrigin separable	high	strong	none	low	high	low	low	none
4: Büche-Rastrigin	high	strong	high	low	high	med.	low	none
5: Linear Slope	none	none	high	none	high	none	none	none
6: Attractive Sector	none	none	high	low	med.	none	none	none
7: Step Ellipsoidal	none	none	high	low	high	none	none	small
8: Rosenbrock	low	none	none	none	med.	low	low	none
9: Rosenbrock rotated	low	none	none	none	med.	low	low	none
10: Ellipsoidal high conditioned	none	none	none	high	high	none	none	none
11: Discus	none	none	none	high	high	none	none	none
12: Bent Cigar	none	none	none	high	high	none	none	none
13: Sharp Ridge	none	none	none	low	med.	none	none	none
14: Different Powers	none	none	none	low	med.	none	none	none
15: Rastrigin multimodal	high	strong	none	low	high	low	low	none
16: Weierstrass	high	med.	none	med.	high	med.	low	none
17: Schaffer F7	high	med.	none	low	med.	med.	high	none
18: Schaffer F7 moderately ill-cond.	high	med.	none	high	med.	med.	high	none
19: Griewank-Rosenbrock	high	strong	none	none	high	low	low	none
20: Schwefel	med.	deceptive	none	none	high	low	low	none
21: Gallagher 101 Peaks	med.	none	none	med.	high	med.	low	none
22: Gallagher 21 Peaks	low	none	none	med.	high	med.	med.	none
23: Katsuura	high	none	none	none	high	low	low	none
24: Lunacek bi-Rastrigin	high	weak	none	low	high	low	low	none

And more importantly, can we use any insight gained from the benchmark experiment to choose a good algorithm for a real world optimization problem? For this, we will use the distance measure SD introduced for the SD/L consensus method to calculate the distances between the 144 rankings as a way to quantify how similar the algorithms performed. We focus our analysis on the 5 to 20 dimensional functions, as they are more challenging than the low dimensional ones. Additionally, algorithm ranks are much more consistent on these functions, which should simplify the analysis.

Ideally, we would like to identify groups from this reduced function set that lead to similar algorithm rankings. We then could reduce the size of the function set by only including one prototype from each group. In addition, new functions similar to functions in this group would probably result in similar algorithm performance. We could therefore try to guess a good algorithm from the function group the new problem belongs to.

Two approaches are used to retrieve groups or clusters from the distance matrix. First, we project the high-dimensional data onto a lower dimensional space by means of multidimensional scaling (MDS, [3]) in order to visualize the relationship between observations. The MDS embeds the observations into the lower dimensional space while attempting to retain the distance between data points. Thus, the starting point for any MDS algorithm is a distance matrix[2] over all observations of a dataset. Starting from

[2] Usually this is a similarity matrix, but it is trivial to compute one out of the other.

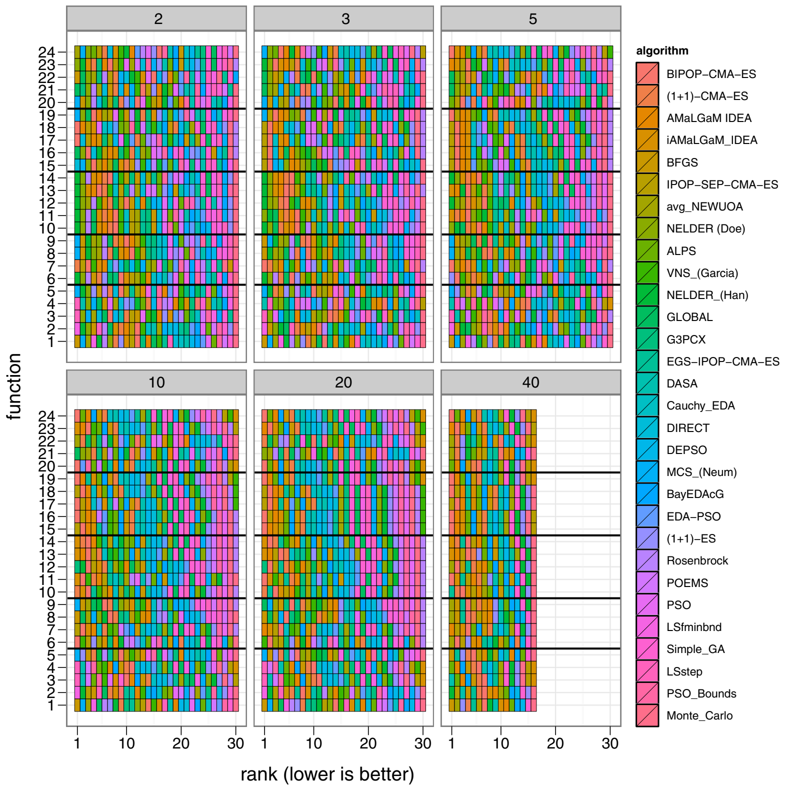


Fig. 2. Benchplot for each dimension by function. The color indicates the algorithm and the position of its rank in the ranking. For higher dimensions, a clear structure emerges because many algorithms are not able to reliably solve such problems with the given number of FEs. The black lines separate the 5 function groups as defined by the BBOB'09 organizers.

this matrix, a geometrical representation of the relationship is created while preserving the input distances or distance as accurately as possible. This is formalized as the optimization problem of minimizing a loss function over the difference between the input and output distance matrix. Next, the clustering algorithm PAM [5] is employed to find clusters in the data, based on the distance matrix. For this dataset 2 clusters were determined to be optimal, yielding an average silhouette width of ≈ 0.44. These clusters were then correlated with the identified function properties (see Table 1). Note that this silhouette width value is not exceptionally high, meaning that the clustering is not seen as very good (that would be the case for values near 1).

The two dimensional MDS of the distance matrix in Fig. 5 shows that a test function will generally lead to similar performance (close proximity) regardless of the dimension. The first MDS component seems to be adequate to describe the clustering. Observations with a value smaller than -50 are assigned to cluster one, the rest to cluster two. This however does not indicate how the clustering relates to the properties of the test

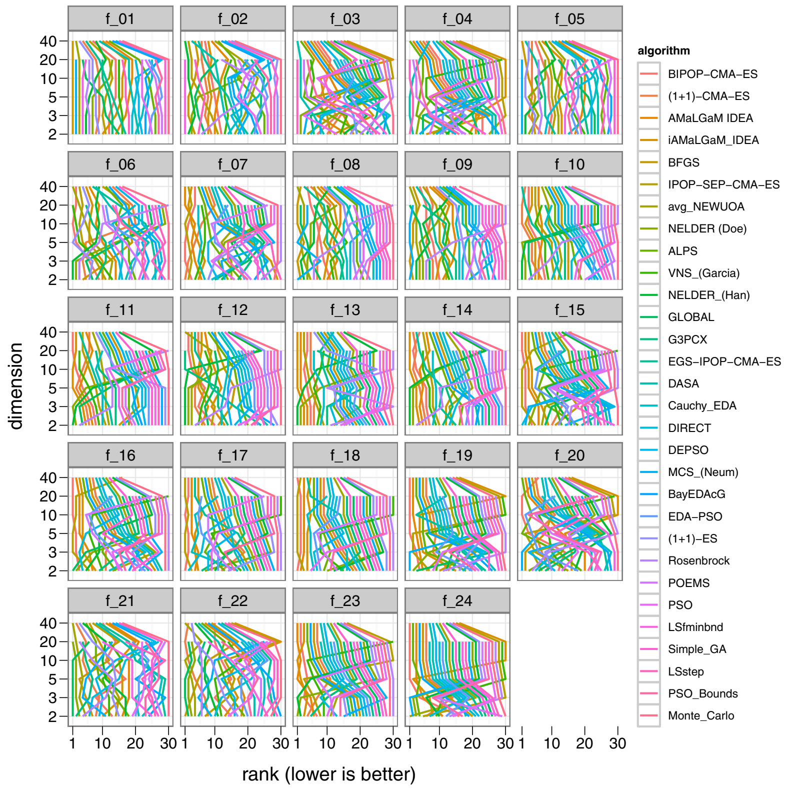


Fig. 3. Parallel coordinate plot for each function, showing the rank of each algorithm as the number of dimensions rise. Some algorithms dramatically loose ground as the number of dimensions rises (diagonal lines in upper right direction).

functions. Therefore, we look at the cross tabulation of the cluster against the function properties. To determine if there is any correlation between the property and the clustering, a Fisher Test was performed for the null hypothesis that the two are independent. For the four properties shown in Table 2 this hypothesis was rejected at the 1% level.

To further model the unknown decision boundary of the clustering with respect to the function properties, a classification tree (Fig. 6) was constructed using the four properties identified by the cross tabulation as having a relationship with the clustering. It comes as no surprise that the first split separates the highly multi-modal problems from all others. The matched 9 problems (27 observations in 5/10/20 dimensions) obviously require a specific type of algorithm, whether no, low, and medium multi-modality go together. Interestingly, the second split considers low variable scaling as a separate function group. Possibly, this reflects that some algorithms have basic means to adapt to rescaled variables where others do not possess such means. The third split between none and deceptive (Schwefel function) global structure is not that surprising. Global structure needs multi-modality, and after removing highly multi-modal functions, the Schwefel function is the only one that possesses a global structure (according to our

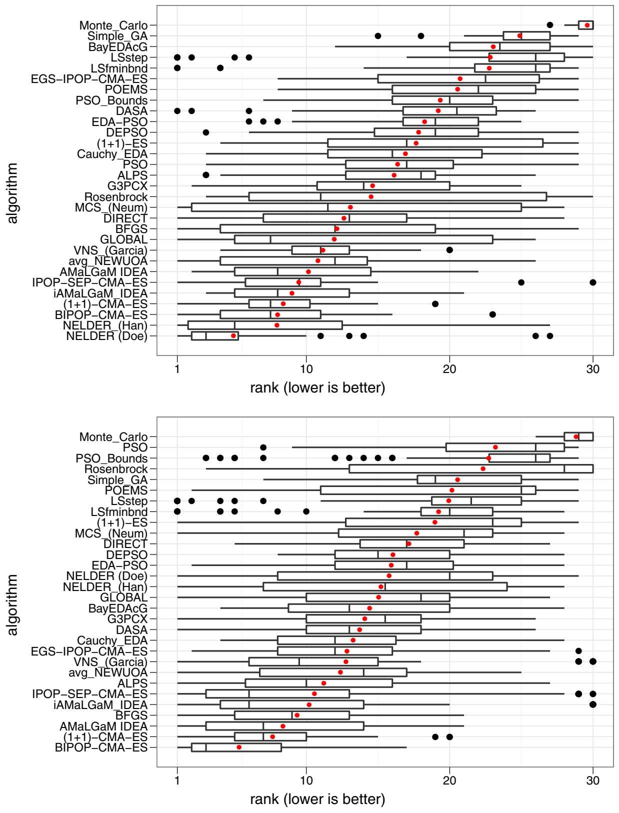


Fig. 4. Boxplot of the ranks for each algorithm in {2,3} (top) and {5,10,20} dimensions (bottom). The algorithms are ordered by mean rank which is marked by a red dot. This corresponds to the Borda ranking method. In low dimensions, the classical Nelder-Mead algorithm performs quite well.

Table 2. Cross tabulation of Multi-modality (a), Global Structure (b), Variable Scaling (c), Global / Local contrast (d) against the clusters found via PAM

(a)

	1	2
none	14	16
low	9	0
medium	3	3
high	0	27

(b)

	1	2
none	26	19
weak	0	3
medium	0	9
strong	0	12
deceptive	0	3

(c)

	1	2
none	12	9
low	1	26
medium	6	3
high	7	8

(d)

	1	2
none	14	16
low	9	24
medium	3	0
high	0	6

classification). It could be interesting to add more functions with low multi-modality but some global structure to the set. Basically, we obtain four different classes of functions with respect to algorithm performance.

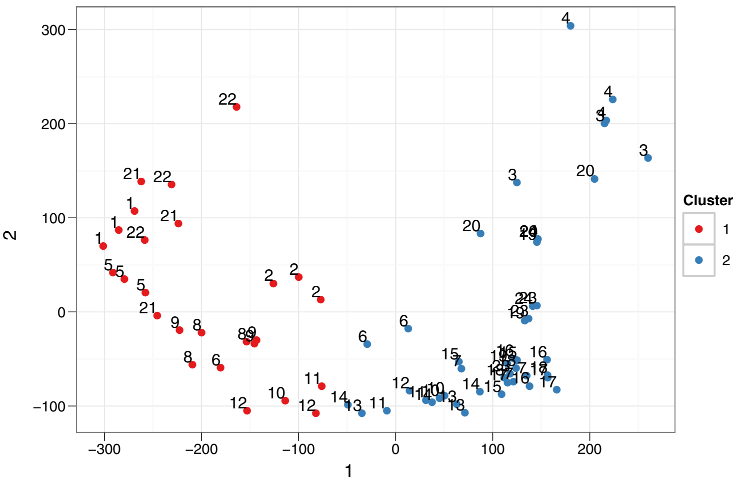


Fig. 5. MDS of the distance matrix obtained from the 5 to 20 dimensional test problems. Color denotes assigned clusters, numbers refer to function numbers. The 5, 10 and 20 dimensional instances of a test function tend to lie in close proximity.

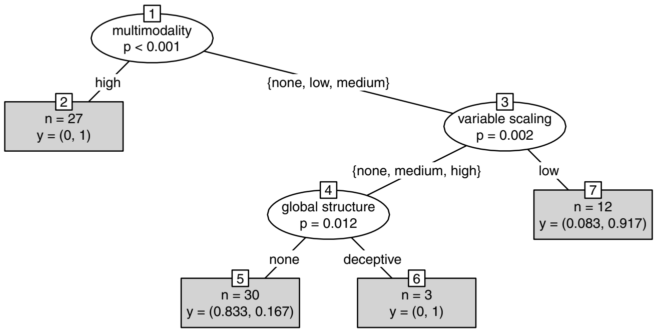


Fig. 6. Decision tree modeling the cluster boundary for the 5 to 20 dimensional function set. The vector y reflects the proportion of observations in each cluster class.

4 Conclusions and Outlook

In this article, we have shown how a combination of benchmarking methods and classical statistical exploratory data analysis can be used to gain insight into the true performance of a set of algorithms under test. Furthermore, we demonstrated how the structure of the function set can be explored. This leads us to define different groups or clusters of functions for which the ranking of the algorithms was essentially the same or very similar. To describe these groups we used decision trees for modeling the unknown cluster boundary. In the future, we would like to extend this work by including additional, measured features, and using the generated decision trees to develop heuristics for supporting practitioners in the choice of an optimization procedure that works well for their problem type.

Acknowledgements. This work was partly supported by the Collaborative Research Center SFB 823 of the German Research Foundation.

References

1. de Borda, J.C.: Mémoire sur les élections au scrutin. Historie de l'Académie Royale des Sciences (1781)
2. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2009: Experimental setup. Tech. Rep. RR-6828, INRIA (2009), http://hal.inria.fr/inria-00362649/en/
3. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, New York (2001)
4. Hunter, D.J.: Essentials of Discrete Mathematics. Jones and Bartlett Publishers, Boston (2008)
5. Kaufman, L., Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley Interscience, New York (1990)
6. Kemeny, J.G., Snell, J.L.: Mathematical Models in the Social Siences. MIT Press, Cambridge (1972)
7. Mersmann, O.: Benchmarking evolutionary multiobjective optimization algorithms using R. Bachelor Thesis, TU Dortmund (2009), http://www.statistik.tu-dortmund.de/~olafm/files/ba.pdf
8. Mersmann, O., Trautmann, H., Naujoks, B., Weihs, C.: Benchmarking evolutionary multiobjective optimization algorithms. In: Ishibuchi, H., et al. (eds.) Congress on Evolutionary Computation (CEC). IEEE Press, Piscataway (2010)
9. Saari, D.G.: The optimal ranking method is the Borda Count. Discussion Paper 638, Northwestern University, Center for Mathematical Studies in Economics and Management Science (1985), http://ideas.repec.org/p/nwu/cmsems/638.html
10. Saari, D.G., Merlin, V.R.: A geometric examination of Kemeny's Rule. Social Choice and Welfare 17, 2000 (1997)
11. 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. Tech. rep., Nanyang Technological University, Singapore (May 2005), http://www.ntu.edu.sg/home/EPNSugan
12. Törn, A., Ali, M., Viitanen, S.: Stochastic Global Optimization: Problem Classes and Solution Techniques. Journal of Global Optimization 14(4), 437–447 (1999)

One-Point Geometric Crossover

Alberto Moraglio

School of Computing and Center for Reasoning, University of Kent, Canterbury, UK
A.Moraglio@kent.ac.uk

Abstract. Uniform crossover for binary strings has a natural geometric interpretation that allows us to generalize it *rigorously* to any search space endowed with a notion of distance and any representation [6]. In this paper, we present an analogous characterization for one-point crossover and explicitly derive formally specific one-point crossovers for a number of well-known representations.

1 Introduction

Geometric crossover [6] is a representation-independent formalization of crossover which requires the offspring to be in the metric segment between parents for some distance. This formalization encompasses many recombination operators across representations [5]. The geometric framework sees recombination operators acting on representations as dual and equivalent to suitably formalized versions of combination operators acting on the neighborhood structure (e.g., path-relinking). Early studies on the relation between crossover and path-relinking in the space of permutations linked with a notion of distance were pioneered by Reeves [7]. The geometric interpretation of crossover operators is interesting as it clarifies what distance/search space is associated with them and how the search operator can be seen as navigating the search space (i.e., sampling offspring in the space-specific segment between parents). This, in turns, clarifies what the fitness landscape induced by a certain crossover operator is, and how the search of this operator relates to the "geographic" structure of the fitness landscape. Ultimately, this characterization may form the basis for a unified, representation-independent theory of evolutionary algorithms [5].

The definition of geometric crossover is rather coarse as it does not consider the actual probability of a particular offspring of being sampled. More finely grained subclasses of geometric crossover can be defined by specifying a probability distribution of the offspring over the segment. Perhaps, the most natural among such subclasses is the uniform geometric crossover, in which offspring are drawn from a uniform distribution over the metric segment between parents. In previous work [6], we have shown that all mask-based crossovers for binary strings are geometric crossovers and that, in particular, uniform crossover [8] is a uniform geometric crossover, as the offspring strings are drawn from a uniform distribution over the metric segment between parent strings under Hamming distance. This equivalence allows us to generalize naturally uniform crossover for binary strings to any representation by making use of the definition of uniform geometric crossover and replacing the Hamming distance with another distance defined on the target representation. For example, the uniform geometric crossover for the swap distance between permutations (i.e., the minimum number of swaps needed to transform a permutation in the other) requires the offspring permutations to be drawn uniformly from the metric segment between parent permutations. This is a formal specification of a recombination operator on permutations rather than the actual operator. To turn this specification into a procedural definition that tells how to manipulate parent permutations to obtain offspring permutations, one can rely on the observation that picking offspring in the segment between two parent permutations is equivalent to generating partially sorted permutations on a minimal sorting trajectories obtained while sorting one parent permutation into the other parent permutation using swaps. Therefore, this operator can be implemented by adapting a traditional sorting algorithm [5]. Importantly, the uniform crossover for binary strings and the recombination for permutations above are fundamentally the same recombination operator instantiated to two different spaces as both share exactly the same geometric definition, that of uniform geometric crossover.

One-point crossover for binary strings [2] selects a common crossover point uniformly at random on the length of the parent strings and produces two offspring by swapping the tails of the parent strings after the crossover point. Whereas it was possible to generalize uniform crossover across representations due essentially to its highly symmetric definition, the situation seems different for one-point crossover as cutting and swapping tails is an operation which relies much on the special characteristic of binary strings of being a vector. In this paper, we show that also one-point crossover has a simple characterization in geometric terms which allows us to generalize it rigorously to *any* representation. We then specify it for a number of well-known representations.

2 Generalization of One-Point Crossover

In the following, we generalize a version of one-point crossover for binary strings that returns only one offspring, in which the head comes from the first parent and the tail comes from the second parent. To do this, we first show how shortest paths between two binary strings in the Hamming space can be constructed (see figure 1). The sequence s of binary strings $s_0, s_1, ..., s_5$ is generated from parent a and b using the bit ordering p which specifies when to exchange the bits between parents. Note that, in the sequence s, the strings s_1, s_2 and s_3 coincide. The bottom part of the figure reports the sequence s' obtained form s after removing repetitions.

Theorem 1. *Every bit ordering generates a shortest path between a and b in the Hamming space. All shortest paths can be generated by using all possible bit orderings and there are $HD(a,b)!$ distinct shortest paths between a and b.*

Proof. We have a few remarks that will lead to prove this theorem. First, the sequence s' forms a path in the Hamming space as two consecutive strings in

```
 p: 1 3 5 2 4
 a: 1 1 1 1 1
 b: 0 1 0 1 0

s0: 1 1 1 1 1
s1: 0 1 1 1 1
s2: 0 1 1 1 1
s3: 0 1 1 1 1
s4: 0 1 1 1 0
s5: 0 1 0 1 0

sequence without repetitions:

    1 1 1 1 1
    0 1 1 1 1
    0 1 1 1 0
    0 1 0 1 0
```

Fig. 1. Example of shortest path generation

the sequence differ in exactly one bit. This is a shortest path as its length equals the Hamming distance between a and b. This holds for any choice of the parent strings and bit ordering. Second, the fact that in the sequence s there are subsequences of repeated elements has its origin in the fact that at some positions the strings a and b do not differ. When the contents at those positions are exchanged the newly generated string equals the previous string in the sequence. So, if we restrict the bit ordering that generates the sequence to those positions in which a and b differ, we can generate the sequence s' directly without repetitions. By definition, the number of positions in which a and b differ is $HD(a, b)$, so the reduced bit ordering p' needs to be an order of $HD(a, b)$ elements. Third, distinct reduced bit orderings produce distinct sequences. Forth, all bit orderings defines all ways of how to apply sequentially all the differences between a and b to a to be turned into b. Hence, all reduced bit orderings, that is, all orders of application of these differences, account for all shortest paths between a and b. So we have $HD(a, b)!$ distinct shortest paths between a and b, that is the number of possible reduced orders on a and b.

Corollary 1. *All possible offspring of one-point crossover for binary strings are on a single geodesic (shortest path) in the Hamming space between parents.*

Proof. All the offspring generated by one-point crossover can be generated from the left to the right by exchanging one by one the contents of the two parents at each position. This is the same sequence obtained by applying the bit ordering $(1 \ldots n)$ to the two parents where n is the length of the parents.

The corollary makes precise and proves an idea of Whitley [9] who noticed earlier that the offspring of one-point crossover forms paths in the Hamming space between parents. Since the notion of geodesic is well-defined in every metric space, the previous theorem allows us to generalize one-point crossover to any metric space, as follows.

Definition 1. *(One-point geometric crossover) In one-point geometric crossover all offspring are on a single geodesic between parents under some distance.*

Notice that in general there may be more than one geodesic between two points (parents). The definition leaves deliberately the geodesic unspecified, but it requires that all offspring are on it. The reason we do not specify a specific geodesic is that since they are all indistinguishable from a distance viewpoint, we cannot specify anyone in particular using only the distance. Only using extra-information based on the underlying representation, we can refer to one geodesic in particular.

This necessary indetermination in the definition of one-point geometric crossover makes it an "over-generalization" of the one-point crossover for binary strings, as explained as follows. There are three distinct types of indetermination in this definition: (i) the specific geodesic is not specified, (ii) the specific probability distribution of the offspring on the geodesic is not specified and, (iii) the specific distance is not specified. Therefore, even when both the probability distribution of the offspring is fixed (e.g., uniform probability on the geodesic) and the operator is instantiated to a specific search space (i.e., the distance is fixed), the indetermination about the specific geodesic makes the one-point operator non-uniquely determined. This is unlike the case of uniform geometric crossover which is unique with respect to its underlying metric space, as the segment between two points, unlike a geodesic, is uniquely determined by these points.

As a consequence of this indetermination, when the geometric one-point crossover is instantiated to the space of binary strings under Hamming distance, it gives rise to a family of crossover operators, one operator for each geodesic between parents, rather than only to the original one-point crossover, which corresponds to a specific geodesic. This family of operators is completely characterized by generating offspring on shortest paths using any possible bit ordering. Every crossover of this family corresponds to a bit ordering, and the strings returned by the application of this ordering to the parent strings correspond to the offspring set of that specific crossover operator (for those parent strings).

When a space allows for a family of one-point geometric crossovers, rather than a single one, they are all indistinguishable from a distance viewpoint. So all are the "right" one-point geometric crossover for the specific space. We will see that in some spaces one member of this family may be preferable to others for reasons linked with the specific character of the underlying representation.

3 Euclidean and Manhattan Spaces

Since in the Euclidean space (endowed with Euclidean distance) a segment comprises a single geodesic, in this space there is only one possible specification of one-point geometric crossover. *Both uniform one-point geometric crossover, in which offspring are drawn uniformly at random on a geodesic, and uniform geometric crossover when specified for the Euclidean space pick points uniformly at random on the only geodesic. So they are equivalent.*

In the 2-D Manhattan space (endowed with Manhattan distance), a segment is a rectangle and its endpoints are two diagonally opposite corners. In higher dimensions, a segment is a hyper-rectangle. For this specific space, uniform geometric crossover picks uniformly offspring in the hyper-rectangle. In this space, a segment comprises infinitely many geodesics linking two points. The geodesics between two points are all monotonic curves joining the two points. *So in the Manhattan space there are infinitely many one-point crossovers, one for each monotonic curve connecting the two parents.*

4 Permutations

In previous work [5], we have shown that PMX, Cycle Crossover, Merge Crossover and others are geometric crossovers. In the introduction, we mentioned that geometric crossovers for permutations are naturally associated with sorting algorithms, giving rise to sorting crossovers [5]. In the following, we consider two different perspectives on one-point geometric crossover for permutations: sorting crossovers and cut-and-fill crossovers.

Let us consider sorting crossovers. Given two permutations, and a move on permutation, e.g., swap of two elements, deterministic sorting algorithms sort the elements of one permutation into the order of the elements of the other permutation always on the same minimal sorting trajectory out of all the possible minimal sorting trajectories. A minimal sorting trajectory corresponds to a geodesic, i.e., a shortest path, on the metric space induced by the sorting move (e.g., the swap move induces a space on permutations endowed with the swap distance). So, the associated deterministic sorting crossovers pick offspring always on the same geodesic between two parents. Hence, *deterministic sorting crossovers are one-point geometric crossovers*. In randomized sorting algorithms, the sorting trajectory is still minimal but non-deterministic. The *sorting crossovers based on randomized sorting algorithms, although being geometric crossovers, are not one-point geometric (under the space induced by the sorting move)*[1] because for two given permutations (parents), different applications of the sorting crossover may return partially ordered permutations (offspring) belonging to different sorting trajectories (geodesics).

Let us now consider cut-and-fill crossovers [1], which are intuitive extensions of one-point crossover for permutations, as follows. If the one-point crossover for binary strings is applied directly on permutations, the offspring so obtained are not permutations. So, this operator cannot be applied as it is but it can be easily adapted: the first parent is cut at a crossover point and the part before the cutting point is passed to the offspring as in the traditional one-point crossover; the second part is then filled in using the order in the second parent avoiding elements already present in the offspring before the crossover point. We call this crossover

[1] Proving that a recombination operator is not a one-point geometric crossover requires showing it for any choice of the underlying distance, and not only for a specific distance. See Moraglio's PhD thesis [5] for how to prove this type of general negative results.

insertion cut-and-fill crossover. Figure 2 (left) shows an example of this crossover. $P1$ and $P2$ are the parent permutations and O is the offspring permutation. The vertical bar in $P1$ indicates the crossover point, and the dashes indicate elements of parent $P2$ whose relative order is preserved in the offspring O. *The insertion cut-and-fill crossover is one-point geometric because it is equivalent to a sorting crossover based on the insertion move*: it is like sorting parent $P2$ into parent $P1$ using the insertion sort algorithm and stopping it when all the elements before the crossover point are sorted.

```
P1: a b c|d e f   P1: a b c|d e f
      -   - -
P2: c d b f e a   P2: c d b f e a
                      -         -
O:  a b c d f e       a d b f e c
                        - -
                      a b d f e c
                          -     -
                  O:  a b c f e d
```

Fig. 2. Insertion cut-and-fill crossover (left) and swap cut-and-fill crossover (right)

The connection between cut-and-fill crossover and sorting crossover suggests that more types of cut-and-fill crossovers can be defined depending on the type of move used as base of the sorting. So we can define a swap cut-and-fill crossover that sorts parent $P2$ into parent $P1$ using selection sort (swap-based minimal sort algorithm) and stopping it when the elements before the crossover point are sorted. *Since swap cut-and-fill crossover is based on a deterministic sorting algorithm, it is a one-point geometric crossover.* Figure 2 (right) gives an example of swap cut-and-fill crossover. Using the same parents and the same crossover point as for the insertion cut-and-fill crossover we obtain a different offspring (the dashes indicate the elements of parent $P2$ that have been swapped to match the order of parent $P1$). If, as a base of the cut-and-fill crossover, we use the adjacent swap move that is associated with the bubble sort algorithm, we obtain again the insertion cut-and-fill crossover. This happens because, apart from the number of moves required (an insertion is equivalent to a sequence of adjacent swaps), after ordering the elements of parent $P2$ into the order of parent $P1$ up to the crossover point the order of the remaining elements is the same as when using insertions or adjacent swaps.

5 Genetic Programming Trees

For GP trees, there are two recombination operators that can be thought as extensions of one-point crossover for binary strings to GP trees: Koza's subtree swap crossover [3] and one-point homologous crossover [4]. Koza's subtree swap crossover is not a geometric crossover [5]. So it is not a one-point geometric crossover either because this is a subclass of geometric crossover. Homologous one-point crossover aligns parent trees at the root and then cut-and-swap subtrees at the same position in the two parents. The family of homologous crossovers for GP trees [4] are geometric crossovers under Structural Hamming distance [5], so also one-point homologous crossover is. However, one-point homologous crossover is not a one-point geometric crossover [5]. After these two negative results, one may wonder how one-point geometric crossovers for GP trees look like. In the following we first present a theorem that helps detecting whether a crossover operator is a one-point geometric crossover. Then we consider a family of crossovers for GP trees that are a subclass of mask-based homologous crossover. Since all mask-based homologous crossovers for GP trees are geometric also this family of crossovers is geometric. Then we show that all these crossovers are one-point geometric.

Theorem 2. *The points $o_1, o_2, \ldots, o_n \in [a,b]_d$ belong to a single geodesic g linking a and b in the metric space d iff they can be ordered as $\bar{o}_1, \bar{o}_2, \ldots, \bar{o}_n$ such that $d(a,\bar{o}_1) + d(\bar{o}_1,\bar{o}_2) + \ldots + d(\bar{o}_{n-1},\bar{o}_n) + d(\bar{o}_n,b) = d(a,b)$.*

Proof. If such an order exists the length of the path g connecting a and b passing thought $o_1, o_2, \ldots, o_n$ is $d(a,b)$. So g is a shortest path linking a and b. If such an order does not exist, for each order of $\bar{o}_1, \bar{o}_2, \ldots, \bar{o}_n$ we have $d(a,\bar{o}_1) + d(\bar{o}_1,\bar{o}_2) + \ldots + d(\bar{o}_{n-1},\bar{o}_n) + d(\bar{o}_n,b) > d(a,b)$ for the triangular inequality. Then the path g connecting a and b passing thought $o_1, o_2, \ldots, o_n$ is larger than $d(a,b)$. So g is not a shortest path linking a and b. Since $o_1, o_2, \ldots, o_n \in [a,b]_d$, for each point there exists a geodesic linking a and b passing for that point. Therefore since there is no geodesic between a and b passing through all points $o_1, o_2, \ldots, o_n$ they must belong to distinct geodesics.

Definition 2. *(Ordered homologous crossover family) Let us define a total order on the nodes of the common region of two parent trees. The order is a (deterministic) function of the two parents. The offspring that the two parents can produce are those obtained, exchanging in the parents the node at position 1 in the order, plus those obtained by exchanging* simultaneously *the nodes at positions 1 and 2, plus those obtained by exchanging* simultaneously *nodes at positions 1, 2 and 3 and so on.*

For example, we can define a total order on the common region by numbering its nodes starting from the root and then visiting and numbering successive nodes of the common region using a breath-first strategy. Then generating uniformly a random number k between 1 and the number of nodes in the common region and exchanging in the two parents all nodes up to k. We term this crossover breath-first homologous geometric crossover. One could change the numbering strategy with a depth-first or bottom-up or any other strategy that visits all nodes of a tree *in a systematic and deterministic way* and obtain new geometric crossover belonging to the ordered homologous crossover family. Figure 5 illustrates the breath-first ordered homologous crossover. The crossover mask tells for each position of the common region from which of the parents to take the node or subtree to pass to the offspring. The crossover mask on the bottom left is valid because the nodes from 1 to 5 marked with 'X' will be passed to the offspring from

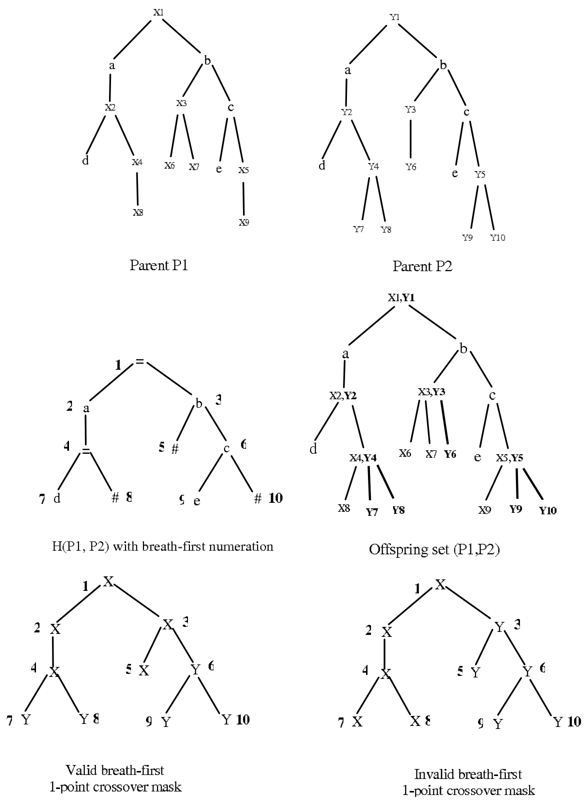


Fig. 3. Breath-first ordered homologous crossover for GP trees: (top) two parent trees P1 and P2; (center left) their associated hyperschema H(P1,P2) with nodes numbered in breath-first order; (center right) all the potential offspring applying homologous crossover to parents P1 and P2 (the part in bold means alternative content of the tree; in this case there are 5 independent binary alternatives, resulting in 32 possible offspring); (bottom left) a valid crossover mask for the breath-first ordered homologous crossover, and (bottom right) an invalid one

parent 1; the node from 6 to 10 marked with 'Y' will be passed to the offspring from parent 2. The crossover mask on the right is invalid for the breath-first ordered homologous crossover because the numbering of the nodes passed to the offspring from parent 1 (marked with 'X') is not an uninterrupted sequence (since X-marked nodes are: 1, 2, 4, 7 and 8).

Theorem 3. *Ordered homologous crossovers are one-point geometric crossovers.*

Proof. We prove it by showing that the offspring $o_1, o_2, \ldots, o_n$ generated respectively by exchanging nodes at position 1, and at positions 1 and 2, and at positions 1, 2 and 3 and so on respect the condition of theorem 2 to be on a geodesic. It is easy to see that the sequence of offspring is a cumulative sequence of independent syntactic differences. Since the metric SHD is a weighted Hamming distance it is an additive distance on the contribution of independent syntactic differences. So we have $d(a,b) = d(a,o_1) + d(o_1,b)$ and $d(a,b) = d(a,o_1) + d(o_1,o_2) + d(o_2,b)$ and $d(a,b) = d(a,o_1) + d(o_1,o_2) + d(o_2,o_3) + d(o_3,b)$ and so on. So we have $d(a,o_1) + d(o_1,o_2) + \ldots + d(o_{n-1},o_n) + d(o_n,b) = d(a,b)$.

The application of one-point geometric crossover to GP trees is instructive because it shows that operators that would have been intuitively understood as reasonable extensions of one-point geometric crossover for binary strings to GP trees, in fact, are not one-point geometric crossovers. This point deserves some attention. In intuitive terms, one-point geometric crossover generalizes the aspect of traditional one-point crossover for binary strings that the offspring must form a chain of gradual syntactic changes leading from one parent to the other parent. In particular, the aspect of the traditional one-point crossover that adjacent syntactic elements (adjacent loci in the string) are more likely to be passed together to the offspring is not captured by the generalization. This is essentially because this property is very specific of the binary string representation and cannot be defined in general geometric terms. One might enforce the adjacency property choosing a specific one-point geometric crossover out of the many possible for the representation at hand, if some notion of adjacency can be defined for the specific representation. For example, the breath-first order crossover introduced above has a property of syntactic adjacency as it can be understood as slicing the tree incrementally starting from the root (see also figure 5). Although this property makes this operator more in the spirit of one-point for binary strings, it is not more one-point geometric than any other operator belonging to the ordered homologous crossover family that does not have such a syntactic adjacency property.

6 Variable-Length Sequences

In previous work [5], we have introduced a class of alignment-based homologous operators for variable-length sequences in which parent sequences are aligned optimally on their contents before exchanging genetic material using a crossover mask on the alignment. This is a closer model of biological recombination at molecular level than traditional crossovers for binary strings. Then, we proved that this class of operators are geometric crossovers under edit distance for sequences. One-point homologous crossover for sequences is an operator belonging to the class of alignment-based homologous operators in which the crossover masks on the alignment are the traditional one-point mask of the one-point crossover for binary strings. The following result shows that one-point homologous crossover for sequences is a one-point geometric crossover.

Theorem 4. *One-point alignment-based homologous crossover is one-point geometric crossover under edit distance.*

Proof. An optimal edit transcript T contains a smallest set E of edit moves to transform parent u in parent v. A mask m selects a subset of edit moves $E_m \subseteq E$

from the transcript T to apply to u and produces the offspring z. z is on a shortest path for the geometricity of homologous crossover. Any homologous crossover operator for which all offspring are generated employing a set of masks m_i that forms a total order (when understood as vectors) generates offspring on a single shortest path between parents. This is because: (i) the sets of edit moves E_{m_i} corresponding to the masks m_i can be totally ordered under inclusion; (ii) the contribution of each edit move to the distance between parents is independent and additive; (iii) hence, when considered in this order, E_{m_i} generate a sequence of offspring z_i on a single shortest path between parents that incrementally leads from u to v. One-point alignment-based homologous crossover uses crossover masks that form a total order: $(00...0) < (10...0) < (11...0) < ... < (11...1)$. So all its offspring are on a single shortest path between parents. Hence it is one-point geometric.

7 Sets

In previous work [5], we have seen that there is a duality between the geometric crossover under Hamming distance for binary strings and the geometric crossover under ins/del edit distance for sets. In fact, these two crossovers, although being based on two different solution representations, are associated to isomorphic metric spaces (via the set indicator function), hence they are completely equivalent. Using the duality, in the following we will show the equivalent for sets of the one-point crossover for binary strings. Let us consider two binary strings of length 5, $p_1 = 11000$ and $p_2 = 01110$. Let $U = \{a, b, c, d, e\}$ be the universal set. The corresponding sets of p_1 and p_2 are $s_1 = \{a, b\}$ and $s_2 = \{b, c, d\}$. After removing repetitions, the geodesic path of the traditional one-point crossover (i.e., with bit ordering 12345) applied to parents p_1 and p_2 is 11000, 01000, 01100, 01110. The corresponding sequence of offspring sets is $\{a, b\}, \{b\}, \{b, c\}, \{b, c, d\}$. Notice that this sequence gradually transforms s_1 into s_2 a move at a time using the ins/del edit move. This is the interpretation of one-point crossover for sets.

8 Conclusions

One-point geometric crossover clarifies and makes rigorous the notion of one-point crossover across representations and formalizes the intuition behind it. It can be used to generate new one-point crossovers for new representations in a formal way without involving ad-hoc adaptations of the original concept. We have derived specific one-point crossovers for a number of well-known representations. Some of the derived operators correspond to pre-existing operators, others are new operators. Few pre-existing operators, which were conceived as analogues of the traditional one-point crossover for other representations, are not one-point geometric crossovers. In future work, we will derive properties common to all one-point crossovers, test the new operators experimentally and extend the geometric framework with the generalization of multi-point crossover.

References

1. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Natural Computing Series. Springer, Heidelberg (2003)
2. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
3. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
4. Langdon, W., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2002)
5. Moraglio, A.: Towards a geometric unification of evolutionary algorithms. PhD thesis, University of Essex (2007)
6. Moraglio, A., Poli, R.: Topological interpretation of crossover. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1377–1388 (2004)
7. Reeves, C.R., Yamada, T.: Genetic algorithms, path relinking, and the flowshop sequencing problem. Evolutionary Computation 6(1), 45–60 (1998)
8. Sywerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the Third International Conference on Genetic Algorithms (1989)
9. Whitley, D.: A genetic algorithm tutorial. Journal of Statistics and Computing 4, 65–85 (1994)

When Does Dependency Modelling Help? Using a Randomized Landscape Generator to Compare Algorithms in Terms of Problem Structure

Rachael Morgan and Marcus Gallagher

School of Information Technology and Electrical Engineering,
University of Queensland, Brisbane 4072, Australia
r.morgan4@uq.edu.au, marcusg@itee.uq.edu.au

Abstract. In this paper we extend a previously proposed randomized landscape generator in combination with a comparative experimental methodology to study the behaviour of continuous metaheuristic optimization algorithms. In particular, we generate landscapes with parameterised, linear ridge structure and perform pairwise comparisons of algorithms to gain insight into what kind of problems are easy and difficult for one algorithm instance relative to another. We apply this methodology to investigate the specific issue of explicit dependency modelling in simple continuous Estimation of Distribution Algorithms. Experimental results reveal specific examples of landscapes (with certain identifiable features) where dependency modelling is useful, harmful or has little impact on average algorithm performance. The results are related to some previous intuition about the behaviour of these algorithms, but at the same time lead to new insights into the relationship between dependency modelling in EDAs and the structure of the problem landscape. The overall methodology is quite general and could be used to examine specific features of other algorithms.

1 Introduction

An important research direction in evolutionary and metaheuristic optimization is to improve our understanding of the relationship between algorithms and the optimization problems that they are applied to. In a general sense, an algorithm can be expected to perform well if the assumptions that it makes, either explicit or implicit, are well-matched to the properties of the search landscape or solution space of a given problem or set of problems.

While it is possible to carry out theoretical investigations of performance and behaviour (e.g. by assuming that the problem has a known analytical form), it is also useful to take a systematic and rigorous approach to the experimental analysis of algorithms. To this end, randomised problem or landscape generators have some favourable properties which can be used to gain insights into the behaviour of metaheuristic optimizers with respect to some underlying properties of the problem instances generated [1].

In this paper we extend a previously proposed randomised landscape generator in combination with a methodology inspired by [2]. We do pairwise performance comparisons of algorithms on 2D test problems with linear ridge structure to gain insight into problem difficulty for different algorithm instances. We then use this approach to investigate the specific issue of explicit dependency modelling in the Estimation of Multivariate Normal Algorithm (EMNA) compared to the Univariate Marginal Distribution Algorithm (UMDA_c) which does not model variable dependencies. The overall methodology is quite general and can be used to examine experimentally the specific features of other algorithms.

An outline of the paper is as follows. Sec. 2 gives an overview of the previous work that provides a basis for the methodology in this paper. The extension of the landscape generator to incorporate linear ridge structure and some illustrative experiments are presented in Sec. 3. In Sec. 4 the extended generator and methodology are used to study the relationship between dependencies in problem variables and the modelling in UMDA_c and EMNA. Sec. 5 concludes the paper.

2 Using a Landscape Generator to Actively Study the Relationship between Problems and Algorithms

Experimental research in metaheuristics is receiving increasing attention in the literature as a means of evaluating and comparing the performance of newly proposed and existing algorithms. This includes the development of large-scale competitions and associated sets of benchmark test problems (e.g at recent Genetic and Evolutionary Computation Conferences (GECCO) and Congress on Evolutionary Computation (CEC)). Several different types of test problems have been used for the evaluation of algorithms, including constructed analytical functions, real-world problem instances or simplified versions of real-world problems and problem/landscape generators [3, 4, 5]. Different problem types have their own characteristics, however it is usually the case that complementary insights into algorithm behaviour result from conducting larger experimental studies using a variety of different problem types.

Max-Set of Gaussians (MSG) [3] is a randomised landscape generator that specifies test problems as a weighted sum of Gaussian functions. By specifying the number of Gaussians and the mean and covariance parameters for each component, a variety of test landscape instances can be generated. The topological properties of the landscapes are intuitively related to (and vary smoothly with) the parameters of the generator.

Langdon and Poli use Genetic Programming (GP) to evolve landscapes for the evaluation and comparison of metaheuristics [2]. Individuals in the GP are candidate landscapes, represented and evolved as 2D polynomial functions. The fitness function for the GP is the performance difference between two specified algorithms that are run on a landscape. Consequently, landscapes found by the GP are optimization problems where one of the algorithms significantly outperforms the other. The results show that considerable new insights can be gained into the behaviour of the algorithms tested and their parameter settings. The methodology is generally applicable to compare other metaheuristic optimization algorithms.

An interesting possibility is to combine the advantages of a randomised landscape generator with an active search for landscapes that maximise performance difference between algorithms. This approach allows greater control over the types of landscapes generated through the parameterisation of the MSG generator, compared to using a

GP to evolve arbitrary polynomial functions. Experiments can be conducted while systematically and incrementally varying the landscape parameters. If a parameterisation is found that produces significant performance difference between two algorithms, a large number of problem instances can be generated with known topological features for analysis and further experimentation.

3 Extending the Landscape Generator to Incorporate Ridge Structure

In this paper we consider 2D continuous optimization problems:

$$\max_f f(\mathbf{x});\ \mathbf{x} = (x_1, x_2) \in I\!R^2;\ f : I\!R^2 \to I\!R$$

A symmetric boundary constraint is implemented such that $\mathbf{x} \in [-1, 1]^2$ by rejecting any search points generated by an algorithm that lie outside the feasible region.

3.1 Constructing Linear Ridges in Randomized Landscapes

Many real world optimization problems are defined over variables with significant dependency relationships. This suggests objective fitness function landscapes with correlation structure or ridges in their contours. In [3] parameterisations of the MSG generator are described that generate localised dependencies, with peaks either uniformly distributed around the space or in a "big valley" structure. However, these rarely lead to global dependency structure and cannot be controlled directly from the generator parameters. We propose an extension to the MSG generator that incorporates linear ridge structure as follows. In two dimensions, a line through the search space is given by

$$ax_1 + bx_2 + c = 0 \tag{1}$$

Our aim is to generate linear ridges positioned randomly in the search space, with a random angle to the coordinate axes. A ridge can be formed by positioning a number of Gaussian components such that their means are distributed along a line. This can be done by generating two points along the bounds of the search space, where each point is along a different boundary. The two points are then used to solve Eqn. 1 for a, b and c. Then, the mean points of the n Gaussians are determined by generating values of x_1 from $U[-1, 1]$ and using Eqn. 1 to find the respective values of x_2.

The orientation (rotation angle) of each Gaussian component on the ridge is determined via its covariance structure. Firstly, an orientation angle is randomly generated. This would impose a homogeneous structure on the ridge with every local peak at the same orientation (between completely aligned with, or orthogonal to the linear ridge). To add further variety, the orientation of each component is subsequently adjusted by a small amount of noise. Examples of ridge landscapes resulting from this method are shown in Fig. 1.

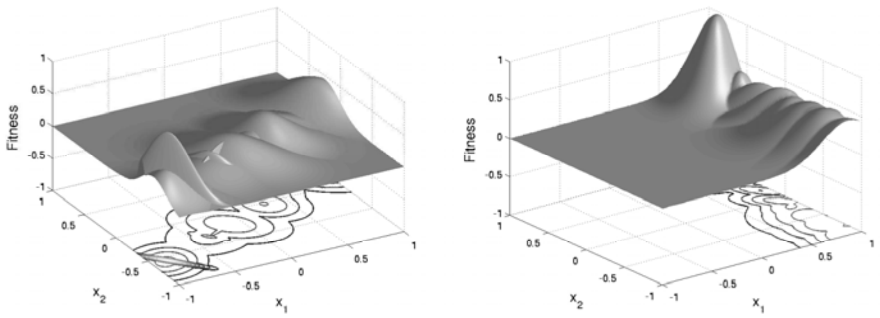


Fig. 1. Example ridge-structured landscape instances from the extended MSG generator

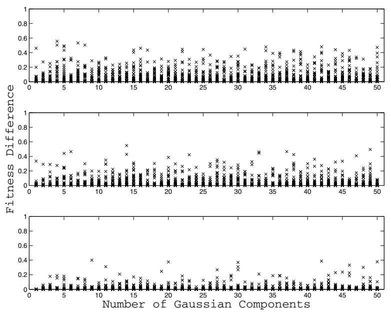
3.2 Illustrative Experiments

To examine the effect of ridge structure on algorithm performance, we compared the Direct algorithm [6], UMDA_c (see Sec. 4) and an implementation of Simulated Annealing on random (i.e. with component mean values uniformly distributed in the feasible search space), big valley and ridge structured landscapes over a varying number of components. For each value, we ran these algorithms on 30 problem instances, with 30 random restarts per instance. Fig. 2(a) shows the average performance difference between Direct and UMDA_c over restarts for all problem instances. We see that Direct and UMDA_c perform quite similarly on big valley, but quite differently on random landscapes. On ridge landscapes, the difference is somewhere in between. Performance is also not strongly related to the number of Gaussian components, except perhaps when the number of components equals 1. In this case, the difference is consistently very small for big valley landscapes, since the global peak will be biased towards the centre of the search space. This is not true for random and ridge landscapes.

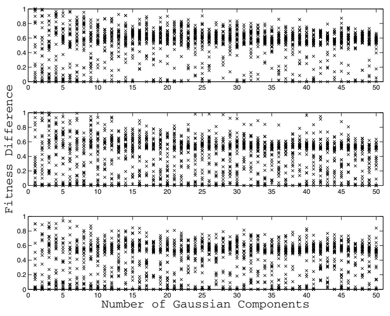
Fig. 2(b) shows the performance difference between Direct and Simulated Annealing. There is some concentration of points close to zero performance difference, that is, trials where the two algorithms performed almost identically (e.g. both found the global optimum). However, a larger fraction of the results is distributed around a performance difference value of approximately 0.6. Not surprisingly, this is strongly related to the structure of the generated landscapes. The generator includes specification of a threshold between the maximum height of local optima (for these results 0.5) and the height of the global optimum (1.0). This threshold will appear in the results of many different algorithms for these landscapes, as most algorithms tend to converge to either the global or a local optimum.

4 Comparing EMNA and UMDA_c in Terms of Landscape Dependency Structure

EDAs are a class of metaheuristic optimization algorithms that build and use a probabilistic model to direct the search process [7, 8]. For continuous problems, the most commonly used model is a Gaussian or Normal distribution with specified covariance



(a) Direct vs UMDA$_c$



(b) Direct vs Simulated Annealing

Fig. 2. Performance comparison results with varying numbers of components in the landscapes generated. Top: random landscapes; Middle: ridge landscapes; Bottom: "big valley" landscapes.

structure. The continuous Univariate Marginal Distribution Algorithm (UMDA$_c$) uses a diagonal covariance matrix, corresponding to a factorised product of univariate Normal distributions. The Estimation of Multivariate Normal (EMNA) algorithm uses a full covariance matrix corresponding to an unrestricted multivariate Normal distribution [7].

One of the major issues that has been explored across EDA research, and has motivated the work in Sec. 3 has been the incorporation of dependency modelling in the probabilistic model of the algorithms. The general assumption is that many real world optimization problems are defined over variables that have unknown dependency relationships between them. Therefore, a model that has the ability to capture and exploit dependencies between problem variables can be expected to provide good performance on such problems. This argument has been experimentally verified several times in the context of developing new algorithms for both continuous and binary problems. If such a model works well for a given optimization problem, it suggests that there are features present in the fitness landscape that the model is able to fit well, but there are few reported studies that specifically analyse the relationship between landscape properties and dependency modelling in EDAs.

4.1 Experimental Results on Ridge Landscapes

In this Section we use the landscape generator described above to evaluate and compare the performance of UMDA$_c$ and EMNA. Our assumption is that linear ridges on the landscape result from a very simple and direct dependency relationship between x_1 and x_2. A set of experiments was carried out as follows. The rotation angle of components in the landscapes (see Sec. 3) was varied between 0 and 45 degrees with increments of 1 degree with random noise of $\pm$ 5 degrees. At each angle, 30 randomised landscapes were generated and 30 trials of each algorithm were conducted on each landscape. Each algorithm used a population size of 50, selection threshold of 0.8 and was run for 50 generations.

Fig. 3 shows the mean fitness difference between UMDA$_c$ and EMNA in terms of best fitness values found on each landscape instance. Counter to our assumption

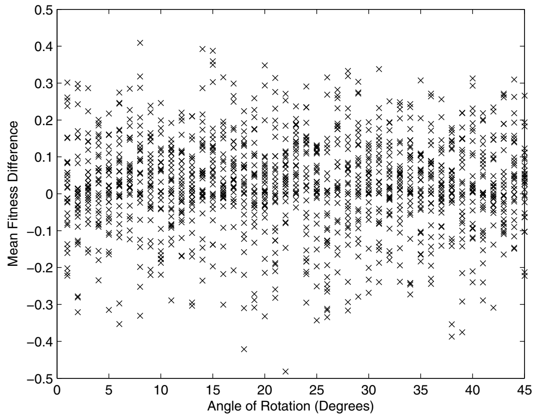


Fig. 3. Fitness difference between UMDA$_c$ and EMNA with varying rotation angle between the ridge and the coordinate axes in the generated landscapes. Fitness difference overall is not correlated with the angle of the ridge. Points tend to be distributed with fitness difference above 0, indicating that UMDA$_c$ actually outperforms EMNA on average across these landscapes.

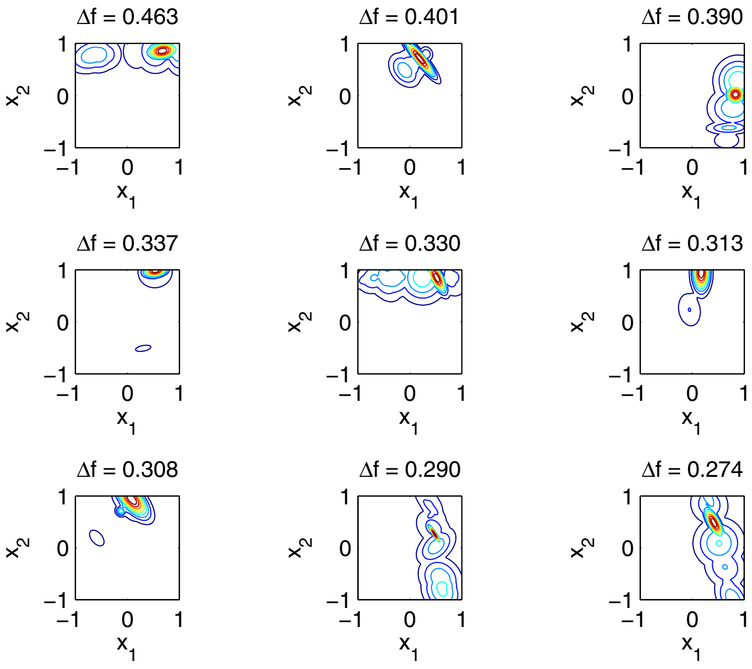


Fig. 4. Example landscape instances where UMDA$_c$ outperforms EMNA

and intuition, the results show no obvious trend between the angle of rotation and the fitness difference of the two algorithms. More surprisingly, UMDA$_c$ tends to *outperform* EMNA regardless of the rotation angle of the ridge, with fitness differences concentrated between 0 and 0.1 and skewed in favour of UMDA_c (positive values). Our expectation was that the full covariance model of EMNA would be more capable of capturing the dependencies within the ridge-structured landscapes.

One benefit of using a landscape generator is that it is possible to analyse results on specific landscape instances. Each data point in Fig. 3 represents a performance difference between the two algorithms averaged over 30 trials. To further investigate the above results, we selected three samples of the landscape instances from Fig. 3 corresponding to the maximum, minimum and approximately zero fitness differences.

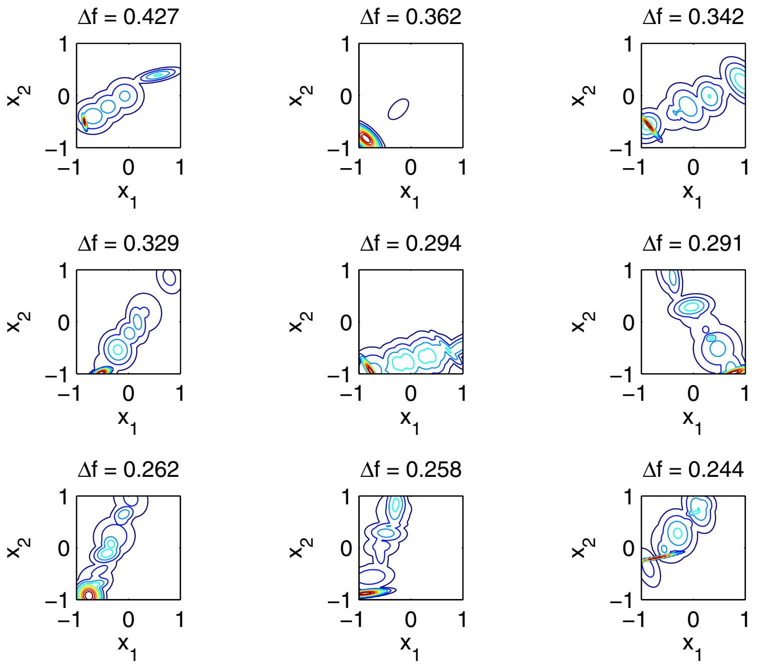


Fig. 5. Example landscape instances where EMNA outperforms UMDA_c

Fig. 4 shows 9 example landscape instances where UMDA_c significantly outperforms EMNA. Ridges in these landscapes tend to be axis aligned, but exhibit significant variation in height along the top of the ridge. Global peaks are relatively narrow compared to other peaks in the landscape and are strongly positioned toward the boundaries of the search space.

In contrast, Fig. 5 shows landscape instances where EMNA significantly outperforms UMDA_c. Ridges in these landscapes are more diagonal than those in Fig. 4, reflecting stronger correlation between x_1 and x_2. The global peaks in Fig. 5 tend to be highly elliptical, as well as being narrow and located near the boundary of the search space.

Fig. 6 shows landscape instances where the performance of EMNA and UMDA_c is almost identical. Global peaks within these landscape are much more regular than those in Figs. 4 and 5 in the sense that they have wider variance, are less elliptical and are

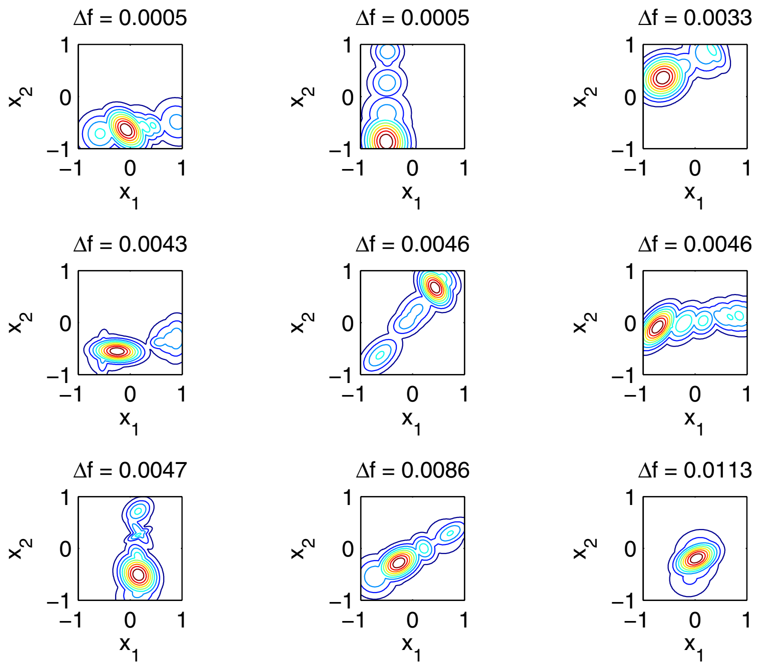


Fig. 6. Example landscape instances where UMDA_c and EMNA perform almost equally

positioned closer to the centre of the search space. Most of the landscapes in Fig. 6 have ridges that are closely aligned to the coordinate axes.

It is clear from Figs. 4-6 that the performance difference between the algorithms is strongly influenced by identifiable features of the landscape. The main topological features that we have observed above are the regularity, position and orientation of the ridge as well as the position and orientation of the global peak relative to the ridge and how elliptical it is. It seems likely that a number of factors are responsible for the performance differences observed in the above experiments. The summary of all results in Fig. 3 focus on a single factor (i.e. orientation) in isolation, but no trend is observed because of the variability in the generated landscape instances contributed by other factors. When these factors are identified and controlled or constrained, clearer performance difference trends may be seen. From Fig. 5, when EMNA outperforms UMDA_c, the landscape does tend to have a diagonal ridge in agreement with our initial assumption. But this is in combination with a global peak that is relatively small, located close to the search space boundary and is highly elliptical in a direction orthogonal to the ridge itself. In contrast, the landscapes where UMDA_c outperforms EMNA (Fig. 4) also tend to have a small global peak located close to the boundary, but the ridges are closely axis-aligned. Furthermore, the global peak is less elliptical, and the ridge itself is often closer to the boundary.

4.2 Discussion and Related Work

The results above can be related to what is already known about the behaviour of UMDA_c and EMNA. It has been shown that the modelling in EMNA does not lead to efficient progress on a linear correlated slope function: the variance of the model extends orthogonally to the direction of increasing fitness (i.e. in the worst possible direction) [9, 10]. This is equally true of UMDA_c if the contours of the slope are axis-aligned, but since the UMDA_c model cannot completely capture a linear dependence, it should outperform EMNA on a correlated slope. Our general observation that UMDA_c tends to outperform EMNA on ridge landscapes (Fig. 3) supports this reasoning.

For univariate or factorisable problems (i.e. uncorrelated variables), UMDA_c is also known to converge prematurely on any monotonic or flat function [11, 12, 13, 14] while convergence is fast on a unimodal function when the model is ”close enough” to the optimum. In the landscapes generated above, a ridge on the search space boundary is globally similar to a monotonic slope (in the direction orthogonal to the ridge), while a ridge through the centre of the search space is more like a unimodal function along most directions through the search space. In the above experiments we have observed the tendency of both algorithms to become stuck on the side of a ridge close to the boundary (such as Fig. 1(b)). However, in practice the behaviour of the algorithms is affected by the interaction of several different factors: ridge location and rotation, global peak size, orientation (with respect to the ridge direction) and eccentricity (see the discussion above). Additional experiments are required to study the interactions of these factors relative to algorithm performance.

5 Conclusions

This paper proposes a general experimental methodology, combining a randomised landscape generator with an active search for problems that differentiate between algorithms. More specifically, an existing generator was extended to incorporate global ridge dependency structure. This was used to investigate the relationship between problem variables and dependency modelling in simple continuous EDAs. The results give insight into the relationship between algorithm performance and landscape structure. They also indicate that, even for parameterised benchmark functions, the experimental behaviour of metaheuristics is a highly complex function of the properties of the problem landscape.

Although the methodology presented is general, the experiments described above were limited to 2D problems and the algorithmic parameter settings used. Examining higher-dimensional problems and integrating the variation of algorithm parameters are avenues for future work. A longer-term goal of this type of exploratory work is to be able to more precisely categorise or quantify the relationship between landscape structure and algorithm behaviour.

References

[1] Rardin, R.L., Uzsoy, R.: Experimental evaluation of heuristic optimization algorithms: A tutorial. Journal of Heuristics 7, 261–304 (2001)

[2] Langdon, W., Poli, R.: Evolving problems to learn about particle swarm optimizers and other search algorithms. IEEE Transactions on Evolutionary Computation 11(5), 561–578 (2007)

[3] Gallagher, M., Yuan, B.: A general-purpose, tunable landscape generator. IEEE Transactions on Evolutionary Computation 10(5), 590–603 (2006)

[4] Gaviano, M., Kvasov, D., Lera, D., Sergeyev, Y.: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software (TOMS) 29(4), 469–480 (2003)

[5] MacNish, C.: Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. Connection Science 19(4), 361–385 (2007)

[6] Jones, D., Perttunen, C., Stuckman, B.: Lipschitzian optimization without the lipschitz constant. Journal of Optimization Theory and Application 79(1), 157–181 (1993)

[7] Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer, Dordrecht (2002)

[8] Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Computational Optimization and Applications 21(1), 5–20 (2002)

[9] Hansen, N.: The CMA evolution strategy: a comparing review. In: Towards a New Evolutionary Computation, pp. 75–102 (2006)

[10] Bosman, P., Grahl, J., Thierens, D.: Enhancing the Performance of Maximum–Likelihood Gaussian EDAs Using Anticipated Mean Shift. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 133–143. Springer, Heidelberg (2008)

[11] González, C., Lozano, J.A., Larrañaga, P.: Mathematical modelling of UMDA$_c$ algorithm with tournament selection. Behaviour on linear and quadratic functions. International Journal of Approximate Reasoning (2002)

[12] Grahl, J., Minner, S., Rothlauf, F.: Behaviour of UMDA$_\mathrm{c}$, with truncation selection on monotonous functions. In: Proc. Congress on Evolutionary Computation (CEC 2005), pp. 2553–2559. IEEE, Los Alamitos (2005)

[13] Yuan, B., Gallagher, M.: A mathematical modelling technique for the analysis of the dynamics of a simple continuous EDA. In: Congress on Evolutionary Computation (CEC), pp. 5734–5740. IEEE, Los Alamitos (2006)

[14] Yuan, B., Gallagher, M.: Convergence analysis of UMDA$_c$ with finite populations: a case study on flat landscapes. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 477–482. ACM, New York (2009)

First-Improvement vs. Best-Improvement Local Optima Networks of NK Landscapes

Gabriela Ochoa[1], Sébastien Verel[2], and Marco Tomassini[3]

[1] School of Computer Science, University of Nottingham, Nottingham, UK
[2] INRIA Lille - Nord Europe and University of Nice Sophia-Antipolis, France
[3] Information Systems Department, University of Lausanne, Lausanne, Switzerland

Abstract. This paper extends a recently proposed model for combinatorial landscapes: *Local Optima Networks (LON)*, to incorporate a first-improvement (greedy-ascent) hill-climbing algorithm, instead of a best-improvement (steepest-ascent) one, for the definition and extraction of the basins of attraction of the landscape optima. A statistical analysis comparing best and first improvement network models for a set of NK landscapes, is presented and discussed. Our results suggest structural differences between the two models with respect to both the network connectivity, and the nature of the basins of attraction. The impact of these differences in the behavior of search heuristics based on first and best improvement local search is thoroughly discussed.

1 Introduction

The performance of heuristic search algorithms crucially depends on the structural aspects of the spaces being searched. An improved understanding of this dependency, can facilitate the design and further successful application of these methods to solve hard computational search problems. Local optima networks (LON) have been recently introduced as a novel model of combinatorial landscapes [6,7,8]. This model allows the use of complex network analysis techniques [5] in connection with the study of fitness landscapes and problem difficulty in combinatorial optimisation. The model, inspired by work in the physical sciences on energy surfaces [3], is based on the idea of compressing the information given by the whole problem configuration space into a smaller mathematical object which is the graph having as vertices the optima configurations of the problem and as edges the possible weighted transitions between these optima (see Figure 1). This characterization of landscapes as networks has brought new insights into the global structure of the landscapes studied, particularly into the distribution of their local optima. Moreover, some network features have been found to correlate and suggest explanations for search difficulty on the studied domains. The study of local optima networks has also revealed new properties of the basins of attraction.

The current methodology for extracting LONs requires the exhaustive exploration of the search space, and the use of a best-improvement (steepest-ascent) local search algorithm from each configuration. In this paper, we are interested in exploring how the network structure and features of a given landscape will change, if a first-improvement (greedy-ascent) local search algorithm is used instead for extracting the basins and transition probabilities. This is apparently simple but, in reality, requires a careful redefinition of the concept of a basin of attraction. The new notions will be presented in the

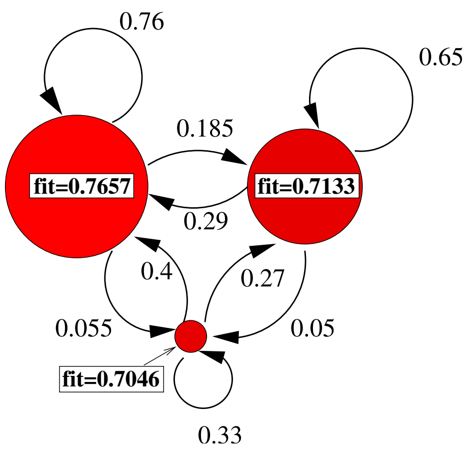


Fig. 1. Visualisation of the weighted local optima network of a small NK landscape ($N = 6$, $K = 2$). The nodes correspond to the local optima basins (with the diameter indicating the size of basins, and the label "fit", the fitness of the local optima). The edges depict the transition probabilities between basins as defined in the text.

next section. Following previous work [7,8], we use the well-known family of NK landscapes [4] as an example, as it allows the exploration of landscapes of tunable ruggedness and search difficulty.

The article is structured as follows. Section 2, includes the relevant definitions and algorithms for extracting the LONs. Section 3 describes the experimental design, and reports the analysis of the extracted networks, including a study of both their basic features and connectivity, and the nature of the basins of attraction of the local optima. Finally, section 4 discusses our main findings and suggest directions for future work.

2 Definitions and Algorithms

A Fitness landscape is a triplet (S, V, f) where S is a set of potential solutions i.e. a search space, $V : S \longrightarrow 2^S$, a neighborhood structure, is a function that assigns to every $s \in S$ a set of neighbors $V(s)$, and $f : S \longrightarrow R$ is a fitness function that can be pictured as the *height* of the corresponding solutions. In our study, the search space is composed by binary strings of length N, therefore its size is 2^N. The neighborhood is defined by the minimum possible move on a binary search space, that is, the 1-move or bit-flip operation. In consequence, for any given string s of length N, the neighborhood size is $|V(s)| = N$. The $HillClimbing$ algorithm to determine the local optima and therefore define the basins of attraction, is given in Algorithm 1. It defines a mapping from the search space S to the set of locally optimal solutions S^*.

First-improvement differs from best-improvement local search, in the way of selecting the next neighbor in the search process, which is related with the so-called *pivot-rule*. In best-improvement, the entire neighborhood is explored and the best solution is returned, whereas in first-improvement, a solution is selected uniformly at random from the neighborhood (see Algorithm 1).

Algorithm 1. Best-improvement (left) and first-improvement (right) algorithms.

```
Choose initial solution $s \in S$
repeat
    choose $s' \in V(s)$, such that $f(s') = max_{x \in V(s)} f(x)$
    if $f(s) < f(s')$ then
        $s \leftarrow s'$
    end if
until $s$ is a Local optimum
```

```
Choose initial solution $s \in S$
repeat
    choose $s' \in V(s)$ using a predefined random ordering

    if $f(s) < f(s')$ then
        $s \leftarrow s'$
    end if
until $s$ is a Local optimum
```

First, let us define the standard notion of a local optimum.

Local optimum (LO). A local optimum, which is taken to be a maximum here, is a solution s^* such that $\forall s \in V(s)$, $f(s) \leq f(s^*)$.

Let us denote by h, the stochastic operator that associates to each solution s, the solution obtained after applying one of the hill-climbing algorithms (see Algorithms 1) for a sufficiently large number of iterations to converge to a LO. The size of the landscape is finite, so we can denote by LO_1, LO_2, $LO_3 \ldots, LO_p$, the local optima. These LOs are the vertices of the *local optima network*.

Now, we introduce the concept of basin of attraction to define the edges and weights of our network model. Note that for each solution s, there is a probability that $h(s) = LO_i$. We denote $p_i(s)$ the probability $P(h(s) = LO_i)$. We have that for:

Best-improvement: for a given solution s, there is a (single) local optimum, and thus an i, such that $p_i(s) = 1$ and $\forall j \neq i, p_j(s) = 0$.

First-improvement: for a given solution s, it is possible to have several local optima, and thus several $i_1, i_2, \ldots, i_m$, such that $p_{i_1}(s) > 0, p_{i_2}(s) > 0, \ldots, p_{i_m}(s) > 0$.

For both models, we have, for each solution $s \in S$, $\sum_{i=1}^{n} p_i(s) = 1$.

Following the definition of the LON model in neutral fitness landscapes [8], we have that:

Basin of attraction. The basin of attraction of the local optimum i is the set $b_i = \{s \in S \mid p_i(s) > 0\}$. This definition is consistent with our previous definition [7] for the best-improvement case.

The size of the basins of attraction can now be defined as follows:

Size of a basin of attraction. The size of the basin of attraction of a local optimum i is $\sum_{s \in \mathcal{S}} p_i(s)$.

Edge weight. We first reproduce the definition of edge weights for the non-neutral landscape, and best-improvement hill-climbing [7]: For each solutions s and s', let $p(s \rightarrow s')$ denote the probability that s' is a neighbor of s, *i.e.* $s' \in V(s)$. Therefore, we define below: $p(s \rightarrow b_j)$, the probability that a configuration $s \in S$ has a neighbor in a basin b_j, and $p(b_i \rightarrow b_j)$, the total probability of going from basin b_i to basin b_j, which is as the average over all $s \in b_i$ of the transition probabilities to solutions $s^{'} \in b_j$ (where $\sharp b_i$ is the size of the basin b_i):

$$p(s \rightarrow b_j) = \sum_{s^{'} \in b_j} p(s \rightarrow s^{'}), \qquad p(b_i \rightarrow b_j) = \frac{1}{\sharp b_i} \sum_{s \in b_i} p(s \rightarrow b_j)$$

For first and best improvement hill-climbing, we have defined the probability $p_i(s)$ that a solution s belongs to a basin i. We can, therefore, modify the previous definitions to consider both types of network models:

$$p(s \rightarrow b_j) = \sum_{s^{'} \in b_j} p(s \rightarrow s^{'}) p_j(s^{'}), \qquad p(b_i \rightarrow b_j) = \frac{1}{\sharp b_i} \sum_{s \in b_i} p_i(s) p(s \rightarrow b_j)$$

In the best-improvement, we have $p_k(s) = 1$ for all the configurations in the basin b_k. Therefore, the definition of weights for the best-improvement case is consistent with the previous definition. Now, we are in a position to define the weighted local optima network:

Local optima network. The weighted local optima network $G_w = (N, E)$ is the graph where the nodes are the local optima, and there is an edge $e_{ij} \in E$, with weight $w_{ij} = p(b_i \rightarrow b_j)$, between two nodes i and j if $p(b_i \rightarrow b_j) > 0$.

According to our definition of edge weights, $w_{ij} = p(b_i \rightarrow b_j)$ may be different than $w_{ji} = p(b_j \rightarrow b_i)$. Thus, two weights are needed in general, and we have an oriented transition graph.

3 Analysis of the Local Optima Networks

The NK family of landscapes [4] is a problem-independent model for constructing multimodal landscapes that can gradually be tuned from smooth to rugged. In the model, N refers to the number of (binary) genes in the genotype (i.e. the string length) and K to the number of genes that influence a particular gene. By increasing the value of K from 0 to $N-1$, NK landscapes can be tuned from smooth to rugged. The K variables that form the context of the fitness contribution of gene s_i can be chosen according to different models. The two most widely studied models are the *random neighborhood* model, where the K variables are chosen randomly according to a uniform distribution among the $n-1$ variables other than s_i, and the *adjacent neighborhood* model, in which the K variables that are closest to s_i in a total ordering $s_1, s_2, \ldots, s_n$ (using periodic boundaries). No significant differences between the two models were found in [4] in terms of the landscape global properties, such as mean number of local optima or autocorrelation length. Similarly, our preliminary studies on the characteristics of the NK landscape optima networks, did not show noticeable differences between the two neighborhood models. Therefore, we conducted our full study on the more general random model.

In order to minimize the influence of the random creation of landscapes, we considered 30 different and independent landscapes for each combination of N and K parameter values. In all cases, the measures reported, are the average of these 30 landscapes.

The study considered landscapes with $N \in \{14, 16\}$ and $K \in \{2, 4, \ldots, N-1\}$, which are the largest possible parameter combinations that allow the exhaustive extraction of local optima networks. Both best-improvement and first-improvement local optima networks (b-LON and f-LON, respectively) were extracted and analyzed.

3.1 Network Features and Connectivity

This section reports the most commonly used features to characterise complex networks, in both the f-LON and b-LON models.

Table 1. NK landscapes network properties. Values are averages over 30 random instances, standard deviations are shown as subscripts. n_v and n_e represent the number of vertexes and edges, $\bar{C}^w$, the mean weighted clustering coefficient. $\bar{Y}$ represent the mean disparity coefficient, $\bar{d}$ the mean path length, and $\bar{d}_{best}$ the mean path length to the global optimum (see text for definitions).

K	$\bar{n}_v$	$\bar{n}_e/\bar{n}_v^2$		$\bar{C}^w$	$\bar{Y}$		$\bar{d}$		$\bar{d}_{best}$	
	$N=14$									
	both	b-LON	f-LON	b-LON	b-LON	f-LON	b-LON	f-LON	b-LON	f-LON
2	14_{6}	0.89	1.00	$0.98_{0.015}$	$0.367_{0.0934}$	$0.172_{0.0977}$	76_{194}	28_{18}	13_{6}	10_{6}
4	70_{10}	0.64	1.00	$0.92_{0.013}$	$0.148_{0.0101}$	$0.048_{0.0079}$	89_{6}	86_{7}	26_{8}	23_{11}
6	184_{15}	0.37	1.00	$0.79_{0.014}$	$0.093_{0.0031}$	$0.025_{0.0017}$	119_{3}	140_{6}	44_{9}	49_{16}
8	350_{22}	0.21	1.00	$0.66_{0.015}$	$0.070_{0.0020}$	$0.017_{0.0008}$	133_{2}	183_{4}	67_{10}	95_{20}
10	585_{22}	0.12	1.00	$0.54_{0.009}$	$0.058_{0.0010}$	$0.014_{0.0004}$	139_{1}	218_{3}	84_{11}	141_{26}
12	896_{22}	0.07	1.00	$0.46_{0.004}$	$0.052_{0.0006}$	$0.013_{0.0002}$	140_{1}	247_{2}	102_{11}	196_{42}
13	$1,085_{20}$	0.06	1.00	$0.42_{0.004}$	$0.050_{0.0006}$	$0.013_{0.0002}$	139_{1}	259_{1}	104_{9}	218_{38}
	$N=16$									
	both	b-LON	f-LON	b-LON	b-LON	f-LON	b-LON	f-LON	b-LON	f-LON
2	33_{15}	0.81	1.00	$0.96_{0.024}$	$0.326_{0.0579}$	$0.110_{0.0590}$	56_{14}	39_{11}	16_{5}	12_{5}
4	178_{33}	0.60	1.00	$0.92_{0.017}$	$0.137_{0.0111}$	$0.033_{0.0064}$	126_{8}	127_{13}	35_{9}	32_{13}
6	460_{29}	0.32	1.00	$0.79_{0.015}$	$0.084_{0.0028}$	$0.016_{0.0014}$	170_{3}	215_{8}	60_{15}	70_{23}
8	890_{33}	0.17	1.00	$0.65_{0.010}$	$0.062_{0.0011}$	$0.011_{0.0004}$	194_{2}	282_{5}	83_{13}	118_{26}
10	$1,470_{34}$	0.09	1.00	$0.53_{0.007}$	$0.050_{0.0006}$	$0.009_{0.0002}$	206_{1}	340_{3}	112_{15}	183_{30}
12	$2,254_{32}$	0.05	1.00	$0.44_{0.003}$	$0.043_{0.0003}$	$0.008_{0.0001}$	207_{1}	380_{2}	143_{16}	271_{48}
14	$3,264_{29}$	0.03	1.00	$0.38_{0.002}$	$0.040_{0.0003}$	$0.008_{0.0001}$	203_{1}	411_{1}	158_{13}	351_{51}
15	$3,868_{33}$	0.02	1.00	$0.35_{0.002}$	$0.039_{0.0004}$	$0.008_{0.0000}$	200_{1}	423_{1}	162_{13}	391_{87}

Number of nodes and edges: The 2nd column of Table 1, reports the number of nodes (local optima),n_v, for all the studied landscapes. The b-LONs and f-LONs have the same local optima, since both local search algorithms, although using a different pivot-rule, are based on the bit-flip neighborhood. The networks, however, have a different number of edges, as can be appreciated in the 3rd and 4th columns of Table 1, which report the number of edges normalized by the square of the number of nodes. Clearly, the number of edges is much larger for the f-LONs. This number is always the square of the number of nodes, which indicates that the f-LONs are complete graphs. It is worth noticing, however, that many of the edges have very low weights (see Figure 2). For the b-LON model, the number of edges decrease steadily with increasing values of K.

Clustering coefficient or transitivity: The *clustering coefficient* of a network is the average probability that that two neighbors of a given node are also neighbors of each other. In the language of social networks, the friend of your friend is likely also to be your friend. The standard clustering coefficient [5] does not consider weighted edges. We thus used the *weighted clustering* measure proposed by [1]. The 5^{th} column of table 1 lists the average coefficients of the b-LONs for all N and K. It is apparent that the clustering coefficients decrease regularly with increasing K, which indicates that either there are less transitions between neighboring basins for high K, and/or the transitions are less likely to occur. On the other hand, the f-LONs correspond to complete networks; the calculation of the clustering coefficients revealed that $\forall i$, $c^w(i) = 1.0$ (not shown in the Table). Therefore, the f-LON is densely connected for all values of K.

Disparity: The *disparity* measure proposed in [1], $Y(i)$, gauges the heterogeneity of the contributions of the edges of node i to the total weight. Columns 6^{th} and 7^{th} in Table 1 depict the disparity coefficients, for both network models, respectively. The heterogeneity decreases with increasing values of K. This reflects that with high values of K, the transitions to other basins tend to become equally likely, an indication of a more random structure (and thus a difficult search). It can also be seen that the weights for the f-LON model are less heterogenous (more uniform) than for the b-LON one.

Shortest path length: Another standard metric to characterize the structure of networks is the shortest path length (number of link hobs) between two nodes on the network. In order to compute this measure on the optima network of a given landscape, we considered the expected number of bit-flip mutations to pass from one basin to the other. This expected number can be computed by considering the inverse of the transition probabilities between basins. More formally, the distance between two nodes is defined by $d_{ij} = 1/w_{ij}$ where $w_{ij} = p(b_i \rightarrow b_j)$. Now, we can define the length of a path between two nodes as being the sum of these distances along the edges that connect the respective basins. Columns 9^{th} and 7^{th} in Table 1 report this measure on the two network models. In both cases, the shortest path increases with K, however, for the b-LON the growth stagnates for larger K values. The paths are considerably longer for the f-LON, with the exception of the lowest values of K. Some paths are more relevant from the point of view of a stochastic local search algorithm following a trajectory over the maxima network. Therefore, columns 10^{th} and 11^{th} in Table 1, report the shortest path length to the global optimum from all the other optima in the landscape. The trend is clear, the path lengths to the optimum increase steadily with increasing K, and similarly, the first-improvement network shows longer paths. This suggest that a larger number of hops will be needed to find the global optimum when a first-improvement local search is used. We must consider, however, that the number of evaluations needed to explore a basin, would be N times lower for first-improvement than for best-improvement.

Outgoing weight distribution: The standard topological characterization of (unweighed) networks is obtained by its degree distribution. The degree of a node is defined as its number of neighbours, and the degree distribution of a network is the distribution over the frequencies of different degrees over all nodes in the network. For weighted networks, a characterization of weights is obtained by the *connectivity and weight distributions* $p_{in}(w)$ and $p_{out}(w)$ that any given edge has incoming or outgoing weight w. In our study, for each node i, the sum of outgoing edge weights is equal to 1 as they represent transition probabilities. So, an important measure is the weight w_{ii} of self-connecting edges (remaining in the same node). We have the relation: $w_{ii} + s_i = 1$.

Figure 2, reports the outgoing weight distributions $p_{out}(w)$ (in log-scale on x-axis) of both the f-LON and b-LON networks on a selected landscape with $K = 6$, and $N = 16$. One can see that the weights, i.e. the transition probabilities to neighboring basins are small. The distributions are far from uniform or Poissonian, they are not close to power-laws either. We couldn't find a simple fit to the curves such as stretched exponentials or exponentially truncated power laws. It can be seen that the distributions differ for the first and best LON models. There is a larger number of edges with low weights for the f-LONs than for the b-LONs. Thus, even though the f-LONs are more densely connected (indeed they are complete graphs) many of the edges have very low weights. For both networks models, we found that the w_{ii} weights (i.e. the probabilities of remaining in the same basin after a bit-flip mutation) are much higher when compared to those w_{ij} with $j \neq i$. The w_{ii} are much lower for the first than for the best LON. In particular, in the studied b-LON, for $K = 2$, around 50% of the random bit-flip mutations will produce a solution within the same basin of attraction, whereas this figure is of less than 20% in the f-LON. Indeed, in this case, for K greater than 4, the probabilities of remaining in the same basin fall below 10%, which suggests that escaping from local optima would be easier for a first-improvement local searcher.

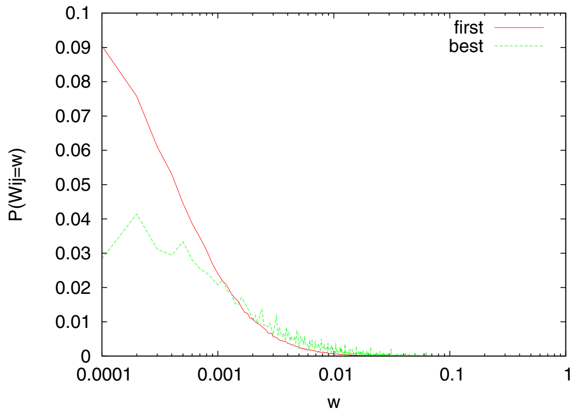


Fig. 2. Probability distribution of the network weights w_{ij} for outgoing edges with $j \neq i$ (in logscale on x-axis) for $N = 16$, $K = 6$. Averages on 30 independent landscapes.

3.2 Basins of Attraction Features

The previous section studied and compared the basic network features and connectivity of the first and best LONs. The exhaustive extraction of the networks, also produced detailed information of the corresponding basins of attraction. Therefore, this section discusses the most relevant of the basin's features.

Size of the global optimum basin: When exploring the average size of the global optimum basin of the f-LONs, we found that they decrease exponentially with increasing ruggedness (K values). This is consistent with the results for the b-LON on these landscapes [7]. Moreover, the basins sizes for both networks are similar, with those of f-LON being slightly smaller. This may suggest that for the the same number of runs, the success rate of a first-improvement heuristic would be lower. One needs to consider, however, that the number of evaluations per run is smaller in this case.

Basin sizes of the two network models: A comparative study of the basin sizes of the two network models revealed that they are highly correlated. Only the smallest basins of the f-LON model are larger in size when compared to the corresponding smallest basins in the b-LON model.

Basin size and fitness of local optima: Fig. 3 reports the correlation coefficients ρ between the networks' basin sizes and their fitness, for both the first and best LONs, and landscapes with $N = 16$ and all the K values. It can be observed that there is a strong correlation between fitness and basin sizes for both types of networks. Indeed, for $K \leq 10$, the correlation is over $\rho > 0.8$. For rugged landscapes, $K > 8$, the f-LON shows reduced and decreasing coefficients as compared to the b-LON.

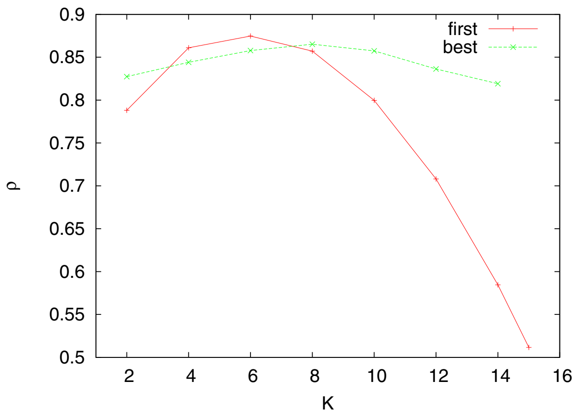


Fig. 3. Average of the correlation coefficient between the fitness of local optima and their corresponding basin sizes on 30 independent landscapes for both f-LON and b-LON ($N = 16$, and all the K values)

Number of basins per solution on the f-LONs: According to the definition of basins (see section 2), for the f-LON, a given solution may belong to a set of basins. Fig. 4 (a) shows the average number of basins to which a solution belongs (i.e. $\sharp\{i \mid p_i(s) > 0\}$). It can be observed that for $N = 16$ and $K = 4$, a solution belongs to nearly 70% of the total number of basins, whereas for $K = 14$, a solution belongs to less than 30% of the total number of basins. On average, a solution belongs to less basins for high K than for low K. An exploration of the average number of basin per solution, according to the solution fitness value (Fig. 4 (b), for $N = 16$) reveals a striking difference. While low fitness solutions belong to nearly all basins, high fitness solutions belong to at most one basin. The figure suggest the presence of a phase transition, in which the threshold of the transition is lower for high K than for low K. This suggests that the structure of the

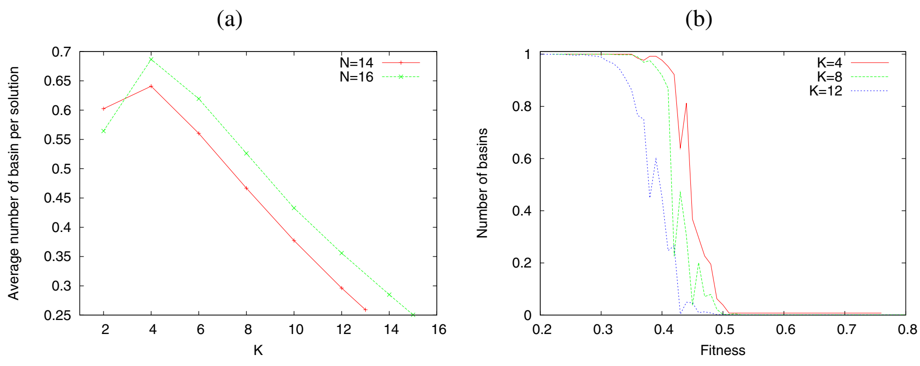


Fig. 4. (a) Average number of basins to which a solution belongs. (b) For $N = 16$ and 3 selected values of K, the number of basins per solution according to the solution fitness value. Averages on 30 independent landscapes.

f-LON network for solutions with high fitness, resembles that of the b-LON, whereas the topology is different with respect to solutions with low fitness.

4 Discussion

We have extended the recently proposed *Local Optima Network (LON)* model to analyze the structural differences between first and best improvement local search, in terms of the local optima network connectivity and the nature of the corresponding basins of attraction. The results of the analysis, on a set of NK landscapes can be summarized as follows. The impact of landscape ruggedness (K value) on the network features is similar for both models. First-improvement induces a densely connected network (indeed a complete network), while this is not the case on the best-improvement model. However, many of the edges in the f-LON networks have very low weights. In particular, the self-connections (i.e. the probabilities of remaining in the same basin after a bit-flip mutation), are much smaller in the f-LON than in the b-LON model, which suggests that escaping from local optima would be easier for a first-improvement local searcher. The path lengths between local optima, and between any optima and the global optimum, are generally larger in f-LON than in b-LON networks. We must consider, however, that the number of evaluations needed to explore a basin, would be N times lower for first-improvement than for best-improvement. We, therefore, suggest that first-improvement is a better heuristic for exploring NK landscapes. Our preliminary empirical results support this insight, a detailed account of them will be presented elsewhere due to space restrictions. Most of our work on the local optima model has been based on binary spaces and NK landscapes. However, we have recently started the exploration of permutation search spaces, specifically the Quadratic Assignment Problem (QAP) [2], which opens up the possibility of analyzing other permutation based problems such as the traveling salesman and the permutation flow shop problems. Our current definition of transition probabilities, although very informative, produces highly connected networks, which are not easy to study. Therefore, we are currently considering alternative definitions and threshold values for the connectivity. Finally, although the local optima network model is still under development, we argue that it offers an alternative view of combinatorial fitness landscapes, which can potentially contribute to both our understanding of problem difficulty, and the design of effective heuristic search algorithms.

Acknowledgment

Gabriela Ochoa gratefully acknowledge the British Engineering and Physical Sciences Research Council (EPSRC) for financial support under grant number EP/D061571/1.

References

1. Barthélemy, M., Barrat, A., Pastor-Satorras, R., Vespignani, A.: Characterization and modeling of weighted networks. Physica A 346, 34–43 (2005)
2. Daolio, F., Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of the quadratic assignment problem. In: Proceedings of the 2010 Congress on Evolutionary Computation, CEC 2010, 3145–3152 (2010)
3. Doye, J.P.K.: The network topology of a potential energy landscape: a static scale-free network. Phys. Rev. Lett. 88, 238701 (2002)
4. Kauffman, S.A.: The Origins of Order. Oxford University Press, New York (1993)
5. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
6. Ochoa, G., Tomassini, M., Verel, S., Darabos, C.: A study of NK landscapes' basins and local optima networks. In: Genetic and Evolutionary Computation Conference, GECCO 2008, pp. 555–562. ACM, New York (2008)
7. Tomassini, M., Verel, S., Ochoa, G.: Complex-network analysis of combinatorial spaces: The NK landscape case. Phys. Rev. E 78(6), 066114 (2008)
8. Verel, S., Ochoa, G., Tomassini, M.: Local optima networks of NK landscapes with neutrality. IEEE Transactions on Evolutionary Computation (to appear)

Differential Mutation Based on Population Covariance Matrix

Karol Opara[1] and Jarosław Arabas[2]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
[2] Institute of Electronic Systems, Warsaw University of Technology, Poland
karol@opara.waw.pl, jarabas@elka.pw.edu.pl

Abstract. In this paper we analyze the impact of mutation schemes using many difference vectors in Differential Evolution (DE) algorithm. We show that for an infinite (sufficiently large) number of difference vectors, distribution of their sum weakly converges to a normal distribution. This facilitates theoretical analysis of DE and leads to introduction of a mutation scheme generalizing differential mutation using multiple difference vectors. The novel scheme uses Gaussian mutation with covariance matrix proportional to the covariance matrix of the current population instead of calculating difference vectors directly. Such modification, called DE/rand/∞, and its hybridization with DE/best/1 were tested on the CEC 2005 benchmark and performed comparable or better than DE/rand/1. Both modified mutation schemes may be easily incorporated into other DE variants. In this paper we provide theoretical analysis, discussion of obtained mutation distributions, and experimental results.

Keywords: differential evolution, differential mutation, covariance matrix, mutation distribution.

1 Introduction

Differential Evolution (DE) is a simple yet powerful algorithm performing unconstrained global minimization in continuous spaces. It differs from other evolutionary algorithms in the sense that distance and direction information from the current population is used to guide the search process via differential mutation [2]. An outline of the basic DE algorithm is given as Algorithm 1. With P_i^t we denote the i-th individual from the population P^t in iteration t. A detailed description of recent advances in DE and a comparative study of state-of-the-art DE variants can be found in [5].

Price and Storn [2,6] proposed a few variants of DE algorithm, denoted as DE/X/Y/Z, where X stands for differential mutation method, Y denotes the number of difference vectors, and Z indicates crossover method. In this contribution we concentrate on two basic and most popular variants [8], DE/rand/1/bin and DE/best/1/bin, where 'bin' stands for binomial crossover.

Algorithm 1. Differential Evolution

```
Initialize parameters: $C_r$, $F$, and $N_p$
Initialize population $P^0$, $t \leftarrow 0$
**while** stop condition not met **do**
  **for all** $i \in \{1, 2, ..., N_p\}$ **do**
    $u_i \leftarrow \texttt{mutation}(F; i, P^t)$
    $o_i \leftarrow \texttt{crossover}(CR; P_i^t, u_i)$
    **if** $f(o_i) \leq f(P_i^t)$ **then**
      $P_i^{t+1} \leftarrow o_i$
    **else**
      $P_i^{t+1} \leftarrow P_i^t$
    **end if**
  **end for**
  $t \leftarrow t + 1$
**end while**
Return arg $\min_i f(P_i^t)$
```

Creating a mutant u_i, which will be later recombined with target vector P_i^t is a two-step process. First, a donor vector $P_{i_1}^t$ is selected. In DE/rand/1 it is chosen with uniform distribution in P^t, while in DE/best/1 scheme it is set to the best point in P^t. Next, two other vectors $P_{i_2}^t$ and $P_{i_3}^t$ are drawn with uniform distribution from the current population. Indices i_1, i_2 and i_3 are required to be pairwise distinct and different from population index i. This restriction protects from creating zero-length difference vectors, which could be harmful for population diversity. Mutant u_i is defined as

$$u_i \leftarrow P_{i_1}^t + F \cdot (P_{i_2}^t - P_{i_3}^t), \tag{1}$$

where scaling factor F is a parameter usually set to $0.4 \leq F < 1$, see [2,6,8].

Among various variants of DE, there are some that use two difference vectors to define mutant, $u_i \leftarrow P_i^t + F(P_{i_1}^t - P_{i_2}^t) + F(P_{i_3}^t - P_{i_4}^t)$, see [2,5,6]. Encouraged by reports about beneficial influence of multiparent recombination in classical evolutionary algorithms [1], in this paper we analyze multiple difference vector DE and generalize it to the case of infinite number of parents. In the following sections we provide theoretical analysis, discussion of obtained mutation distributions, and experimental results.

2 Differential Mutation Distribution

Choosing a random point from population P^t may be interpreted as a realization of a random variable X_{P^t} depicting uniform random distribution within the population P^t. Consider a random variable $V_{P^t}^{i,i_1}$ corresponding to possible values of difference vectors for population P^t, target vector P_i^t and donor vector $P_{i_1}^t$. It may be expressed as a product of a scaling factor F and two randomly chosen points distinct from target vector, donor vector and each other

$$V_{P^t}^{i,i_1} = F \cdot \left(X_{P^t \setminus \{P_i^t, P_{i_1}^t\}} - X_{P^t \setminus \{P_i^t, P_{i_1}^t, P_{i_2}^t\}} \right).$$

For each P^t, i, and i_1, distributions of $V_{P^t}^{i,i_1}$ have zero mean and are symmetric about the origin, as for each difference vector $F \cdot (P_{i_2}^t - P_{i_3}^t)$ an opposite vector $F \cdot (P_{i_3}^t - P_{i_2}^t)$ exists. Except for very small populations, random variables $V_{P^t}^{i,i_1}$ and $V_{P^t}^{j,j_1}$ have very simillar distributions and can be jointly approximated by a random variable V_{P^t},

$$V_{P^t}^{i,j} \approx V_{P^t} = F \cdot (X_{P^t} - X_{P^t})\,. \tag{2}$$

This reflects the case, in which indices i, i_1, i_2 and i_3 are not required to be pairwise distinct. Consequently, covariance matrix $\operatorname{cov}\left(V_{P^t}^{i,i_1}\right)$ of any random variable $V_{P^t}^{i,i_1}$ can be approximated by

$$\operatorname{cov}(V_{P^t}^{i,j}) \approx \operatorname{cov}(V_{P^t}) = 2F^2 \cdot \operatorname{cov}(X_{P^t}). \tag{3}$$

3 Multivector Differential Mutation

Differential mutation using k difference vectors can be expressed as

$$u_i = P_{i_1}^t + v' \text{ where } v' \sim {V'}_{P^t}^{i,i_1} = \sum_{j=1}^{k} v_j, \text{ and } v_j \sim V_{P^t}^{i,i_1}. \tag{4}$$

Direct application of this formula causes, however, some inconvenience. Mutation range, measured by the covariance matrix $\operatorname{cov}({V'}_{P^t}^{i,i_1})$ of difference vector distribution, increases along with the number of difference vectors

$$\operatorname{cov}({V'}_{P^t}^{i,i_1}) = k \operatorname{cov}(V_{P_i^t}^{i,i_1}).$$

To compensate for that, scheme (4) can be redefined, so that covariance matrices were equal. In this way, mutation range is equivalent to DE variants with one difference vector.

$$u_i = P_{i_1}^t + \frac{1}{\sqrt{k}} v' \text{ where } v' \sim {V'}_{P^t}^{i,i_1} = \sum_{j=1}^{k} v_j, \text{ and } v_j \sim V_{P^t}^{i,i_1} \tag{5}$$

An offspring vector u_i is therefore created by adding a scaled sum of k vectors derived independently from identical distribution $V_{P^t}^{i,i_1}$ to the donor vector $P_{i_1}^t$. For $k \to \infty$ one can apply the Central Limit Theorem to formula (5) and conclude, that

$$\frac{1}{\sqrt{k}} v' = \frac{1}{\sqrt{k}} \sum_{i=1}^{k} v_i \xrightarrow{\mathcal{D}} \mathcal{N}(0, \operatorname{cov}(V_{P^t}^{i,i_1})).$$

This means, that for sufficiently large k, the empirical cumulative distribution function (cdf) of $\frac{1}{\sqrt{k}} v'$ tends to the cdf of a multivariate normal distribution $\mathcal{N}(0, \operatorname{cov}(V_{P^t}^{i,i_1}))$. Hence, for large k, mutation using k difference vectors (5) is equivalent to adding a normally distributed random variable with zero mean and covariance matrix $\mathrm{cov}(V_{P^t}^{i,i_1})$. This is an important observation, as it simplifies formulas describing difference vector distribution and facilitates theoretical analysis of DE. Furthermore, this result leads to the definition of a generalized multivector mutation scheme

$$u_i \leftarrow P_{i_1}^t + v_\infty, \text{ where } v_\infty \sim \mathcal{N}\left(0, \mathrm{cov}\left(V_{P^t}^{i,i_1}\right)\right). \tag{6}$$

However, direct application of formula (6) would require computing difference vector distributions $V_{P^t}^{i,i_1}$ and their covariance matrices for each pair of target and donor vector, which would be very time-consuming. To address these problems, we used approximate relation (3) obtaining

$$u_i \leftarrow P_{i_1}^t + \sqrt{2}F \cdot v_\infty, \text{ where } v_\infty \sim \mathcal{N}\left(0, \mathrm{cov}(X_{P^t})\right). \tag{7}$$

In formula (7), covariance matrix of the current population distribution is computed only once per iteration. Moreover, the distribution of v_∞ is continuous and the mutation range can be tuned with scaling factor F. Since convergence in distributions happens for $k \to \infty$, we called the above mutation scheme DE/rand/∞. Proportional dependence between covariance matrix of the current population and the covariance matrix used in (7) exploits contour fitting [6] properties of DE.

4 Discussion

The idea of sampling mutants from a random distribution dependent on the current population is widely used in Estimation of Distribution Algorithms [4]. It is also present in Evolution Strategies, where points undergo a Gaussian mutation [2]. Moreover, DE with multiple difference vectors or mutation scheme defined by formula (7), implicitly adapts mutation covariance matrix, which resembles some solutions used explicitly in a state of the art CMA-ES algorithm [3].

To illustrate differences between DE/rand/1 and DE/rand/∞ we define a two-dimensional fitness function f, which is a weighted sum of three Gaussian peaks with means $M_1 = [-1, 7]^T$, $M_2 = [-9, -2]^T$, $M_3 = [5, -6]^T$, covariance matrices $C_1 = [5, 2; 2, 1]$, $C_2 = [1, -1; -1, 4]$, $C_3 = [5, -2; -2, 1]$ and weights $w_1 = 2$, $w_2 = 1$, $w_3 = 2$

$$f(x) = \sum_{i=1}^{3} w_i \frac{1}{2\pi \det(C_i)^{1/2}} \exp\left(-\frac{1}{2}(x - M_i)^T C_i^{-1} (x - M_i)\right)$$

DE is invariant with respect to order preserving transformations, i.e. it takes into account the order between points rather than their fitness values. Consequently, to give better impression of the fitness function shape, contour lines in Fig. 1, and 2 are not equally spaced, but are more densely distributed at low values.

Figure 1 presents population located on three competing optima in the search space and appropriate distributions of difference vectors for DE/rand/1, DE/rand/2 and DE/rand/∞. To enhance visual analysis of the density of difference vectors, the number of different points plotted in each figure is equal to

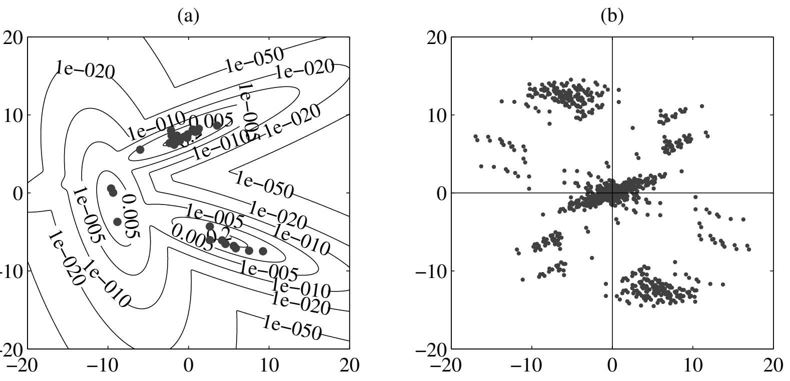
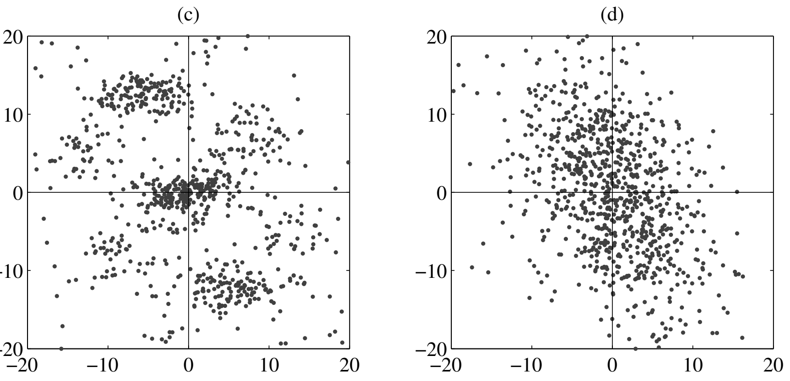



Fig. 1. Difference vector distributions for population located on competing optima (a), and mutation methods: (b) – DE/rand/1, (c) – DE/rand/2, (d) – DE/rand/∞

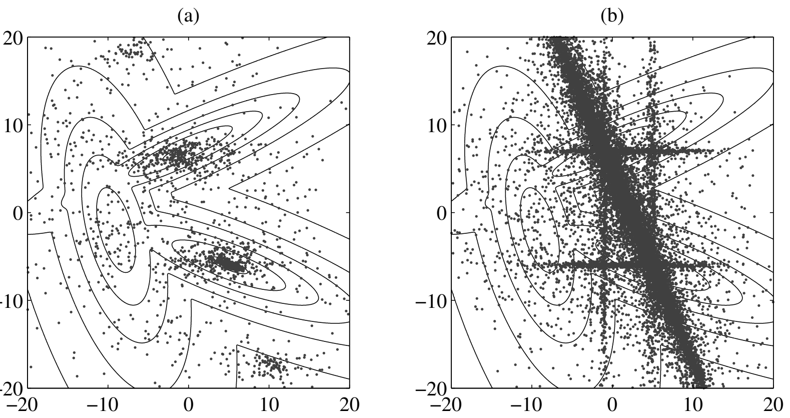


Fig. 2. Trace of points generated in a single run of algorithms (a) – DE/rand/1/bin, (b) – DE/rand/∞/bin

$N_p(N_p - 1)$, which is the number of all possible pairings of points with different indices and equals the number of difference vectors in DE/rand/1 scheme. In case of DE/rand/2 and DE/rand/∞, plotted points were randomly chosen. In case of DE/rand/1, Fig. 1b, distribution concentrates nearby the origin and in a few separate groups depicting the difference vectors between points located in distinct competing optima. For DE/rand/2, Fig. 1c, such groups are blurred, and the spread of the difference vectors is wider. Mutation scheme DE/rand/∞, Fig. 1d, explores the whole area between competing optima rather than only separated groups of points. This property is clearly visible in Fig. 2, where a set of all points sampled by DE/rand/1/bin and DE/rand/∞/bin is plotted for a single run of appropriate algorithm. In Fig. 2a one can notice groups of points sampled away from optima, which are a side effect of presence of competing optima. In Fig. 2b instead of those groups, narrow Gaussian shapes appear. The 'crosses' nearby the optima are a consequence of binomial crossover.

Large range of the DE/rand/∞ mutation is beneficial, as far as global search is concerned, but decreases exploitation properties – in Fig. 1d much less points are located nearby the origin than in Fig. 1b showing difference vector distribution for DE/rand/1. This may affect the precision of locating a optimum. To deal with this problem we decided to introduce a modified mutation scheme called hybrid DE/rand/∞. For each target point we draw a random number from the uniform distribution in $(0, 1)$. If it is lower than a certain parameter c_b, we perform DE/rand/∞ mutation, otherwise DE/best/1 mutation. We used $c_b = 0.8$, as our preliminary study showed it provides good balance between exploration and exploitation. An outline of Hybrid DE/rand/∞ mutation is depicted as Algorithm 2. The choice of DE/best/1 seems natural for DE and yields good results. However, DE/rand/∞ may be also hybridized with other exploitative mutation operators or even local search methods.

Algorithm 2. Hybrid DE/rand/∞ mutation

```
if rand() $< c_b$ then
    /** DE/rand/$\infty$ mutation **/
    $u_i \leftarrow P^t_{i_1} + \sqrt{2}F \cdot v_i$ where $v_i \sim \mathcal{N}(0, \mathrm{cov}(P^t))$
else
    /** DE/best/1 mutation **/
    Choose randomly indices $i_2$ and $i_3$, such that $i$, $i_2$ and $i_3$ be pairwise distinct
    $u_i \leftarrow \arg\min_i f(P^t_i) + F \cdot (P^t_{i_2} - P^t_{i_3})$
end if
```

5 Experimental Study

To compare performance of the proposed mutation schemes we decided to use CEC 2005 benchmark, see [9]. It contains 25 problem definitions (functions) and appropriate evaluation criteria. Functions F1-F5 are unimodal, F6-F12 are basic multimodal, and F13-F25 are multimodal composition functions composed from the functions F1-F12. In case of F7 and F25 no bounds are assumed, and only the

Table 1. Comparison of error values obtained by DE/rand/1/bin and the two proposed algorithms, DE/rand/∞/bin and Hybrid DE/rand/∞/bin, using Wilcoxon rank sum test for ten-dimensional and thirty-dimensional CEC 2005 functions

Ten-dimensional functions													
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
DE/rand/∞/bin	·	·	+	+	+	+	−	·	−	−	−	·	−
Hybrid DE/rand/∞/bin	·	·	+	+	+	+	·	+	−	·	−	+	·
	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	
DE/rand/∞/bin	·	−	−	−	·	·	·	·	·	+	·	·	
Hybrid DE/rand/∞/bin	+	−	·	·	·	·	·	+	·	+	·	·	

Thirty-dimensional functions													
	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
DE/rand/∞/bin	+	+	+	+	+	+	+	·	−	·	·	·	−
Hybrid DE/rand/∞/bin	+	+	+	+	+	+	+	·	−	·	·	·	−
	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	
DE/rand/∞/bin	·	·	−	·	−	−	−	·	·	·	+	+	
Hybrid DE/rand/∞/bin	·	·	·	·	·	·	·	·	·	·	+	+	

initialization area is indicated. In some cases the global minimum is located on bounds. A detailed description of function characteristics end evaluation criteria can be found in [9].

For each fitness function under investigation, we performed 25 independent runs of DE/rand/1/bin, DE/rand/∞/bin and Hybrid DE/rand/∞/bin, each time setting the same initial population for all variants. The maximal number of function evaluations was set to 10^5 for ten-dimensional functions and to $3 \cdot 10^5$ for thirty-dimensional ones, [9]. After each run, we calculated the difference between function values in the best solution and in the global minimum, obtaining a population of 25 error values. To save space, in tables 2 and 3 we present only the final statistics of the error values for ten- and thirty-dimensional functions: median and interquartile range (IQR), as well as mean and standard deviation.

For all investigated DE variants we set the same parameter values (scaling factor $F = 0.9$, crossover rate $C_r = 0.9$, population size $N_p = 30$) following the tuning chosen for DE/rand/1/bin during CEC 2005 competition [7]. Therefore, DE/rand/∞/bin algorithm was run with potentially suboptimal settings. We also applied reflection as a method to handle boundary constraints, and for DE/rand/ 1/bin method we obtained results which are consistent with those reported in [7].

To compare results for the tested mutation versions we used a two-sided Wilcoxon rank sum test for equal medians at significance level 0.05. Results of the test are reported in Tab. 1. In each cell, we put '+' whenever one of the introduced algorithms had significantly lower median than DE/rand/1/bin, '−' when it had greater median, and '·' when there was no statistically significant difference.

Table 2. Statistics of optimization error values obtained after 10^5 function evaluations within 25 independent runs of DE/rand/1/bin, DE/rand/∞/bin and Hybrid DE/rand/∞/bin algorithms for ten-dimensional CEC 2005 benchmark functions

		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**
DE/rand/1/bin	**Median**	0.0e+00	0.0e+00	4.7e−07	5.7e−14	0.0e+00	2.8e−13	7.9e−02	2.0e+01	9.9e−01	9.9e+00	6.7e−02	1.9e−11	8.6e−01
	IQR	0.0e+00	0.0e+00	8.3e−06	2.0e−13	4.5e−13	1.5e−12	7.4e−02	9.7e−02	9.9e−01	1.0e+01	1.8e+00	5.8e−10	7.0e−01
	Mean	0.0e+00	4.5e−15	2.5e−05	2.3e−13	7.3e−13	3.9e−12	1.0e−01	2.0e+01	8.0e−01	1.5e+01	1.4e+00	5.9e+01	1.0e+00
	Std	0.0e+00	1.6e−14	6.6e−05	3.9e−13	1.4e−12	1.3e−11	5.9e−02	1.0e−01	9.9e−01	9.6e+00	2.6e+00	2.0e+02	4.9e−01
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	3.6e+00	4.1e+02	1.1e+02	1.1e+02	3.0e+02	3.0e+02	3.0e+02	5.0e+02	7.7e+02	5.6e+02	2.0e+02	3.8e+02	
	IQR	2.8e−01	3.5e+02	1.3e+01	1.3e+01	1.3e+02	0.0e+00	1.3e+02	0.0e+00	1.6e+01	2.3e−12	0.0e+00	3.6e+00	
	Mean	3.5e+00	3.1e+02	1.1e+02	1.2e+02	4.2e+02	4.0e+02	4.2e+02	5.0e+02	7.6e+02	5.7e+02	2.0e+02	3.8e+02	
	Std	4.9e−01	1.7e+02	1.3e+01	2.2e+01	2.2e+02	2.0e+02	2.2e+02	4.9e−13	9.8e+01	3.2e+01	0.0e+00	2.9e+00	

		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**
DE/rand/∞/bin	**Median**	0.0e+00	0.0e+00	6.5e−11	0.0e+00	0.0e+00	0.0e+00	5.9e−01	2.0e+01	2.6e+01	3.1e+01	9.1e+00	4.8e−09	2.5e+00
	IQR	0.0e+00	0.0e+00	1.4e−10	0.0e+00	0.0e+00	0.0e+00	1.5e−01	1.6e−01	3.7e+00	6.3e+00	9.0e−01	1.0e+01	6.9e−01
	Mean	0.0e+00	0.0e+00	1.2e−10	0.0e+00	0.0e+00	0.0e+00	5.7e−01	2.0e+01	2.6e+01	3.2e+01	8.9e+00	1.7e+02	2.5e+00
	Std	0.0e+00	0.0e+00	1.4e−10	0.0e+00	0.0e+00	0.0e+00	9.2e−02	9.9e−02	3.1e+00	4.8e+00	8.1e−01	5.3e+02	4.5e−01
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	3.6e+00	4.8e+02	1.7e+02	1.8e+02	3.0e+02	3.0e+02	3.0e+02	5.0e+02	7.7e+02	5.6e+02	2.0e+02	3.8e+02	
	IQR	1.7e−01	2.3e+02	1.6e+01	1.5e+01	5.0e+02	5.0e+02	5.0e+02	0.0e+00	4.7e+00	0.0e+00	0.0e+00	5.5e+00	
	Mean	3.6e+00	4.0e+02	1.7e+02	1.8e+02	4.8e+02	4.4e+02	4.8e+02	4.9e+02	7.4e+02	5.7e+02	2.0e+02	3.8e+02	
	Std	1.5e−01	1.2e+02	1.0e+01	1.6e+01	2.4e+02	2.3e+02	2.4e+02	4.0e+01	1.3e+02	3.2e+01	0.0e+00	3.4e+00	

		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**
Hybrid DE/rand/∞/bin	**Median**	0.0e+00	0.0e+00	2.3e−13	0.0e+00	0.0e+00	0.0e+00	1.2e−01	2.0e+01	5.0e+00	1.3e+01	1.5e+00	0.0e+00	1.1e+00
	IQR	0.0e+00	0.0e+00	5.5e−13	0.0e+00	0.0e+00	0.0e+00	1.2e−01	2.4e−01	4.2e+00	4.2e+00	4.5e+00	1.2e+01	1.8e+00
	Mean	0.0e+00	0.0e+00	8.7e−13	0.0e+00	0.0e+00	0.0e+00	1.6e−01	2.0e+01	5.4e+00	1.4e+01	3.0e+00	5.7e+01	1.6e+00
	Std	0.0e+00	0.0e+00	2.0e−12	0.0e+00	0.0e+00	0.0e+00	1.2e−01	1.5e−01	3.1e+00	4.7e+00	3.1e+00	1.6e+02	1.0e+00
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	3.4e+00	4.3e+02	1.1e+02	1.2e+02	3.0e+02	3.0e+02	3.0e+02	5.0e+02	7.7e+02	5.6e+02	2.0e+02	3.8e+02	
	IQR	3.6e−01	3.3e+02	1.8e+01	2.9e+01	5.0e+02	5.0e+02	5.0e+02	0.0e+00	2.7e+01	1.1e−13	0.0e+00	5.0e+00	
	Mean	3.3e+00	3.4e+02	1.1e+02	1.2e+02	5.2e+02	5.0e+02	4.6e+02	4.8e+02	7.5e+02	5.8e+02	2.0e+02	3.8e+02	
	Std	5.0e−01	1.6e+02	1.3e+01	2.4e+01	2.5e+02	2.5e+02	2.4e+02	6.6e+01	9.8e+01	5.4e+01	0.0e+00	3.2e+00	

Table 3. Statistics of optimization error values obtained after $3 \cdot 10^5$ function evaluations within 25 independent runs of DE/rand/1/bin, DE/rand/∞/bin and Hybrid DE/rand/∞/bin algorithms for thirty-dimensional CEC 2005 benchmark functions

DE/rand/1/bin		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
	Median	5.7e−14	5.6e−02	1.0e+06	1.6e+01	1.1e+02	1.2e+01	7.4e−03	2.1e+01	2.0e+01	1.9e+02	3.9e+01	1.6e+03	3.1e+00
	IQR	0.0e+00	6.8e−02	5.8e+05	3.1e+01	1.8e+02	1.0e+01	1.0e−02	7.0e−02	9.0e+00	1.6e+02	1.4e+00	2.7e+03	1.4e+00
	Mean	5.7e−14	1.2e−01	1.1e+06	2.4e+01	1.7e+02	2.3e+01	6.4e−03	2.1e+01	2.4e+01	1.4e+02	3.7e+01	2.6e+03	3.5e+00
	Std	0.0e+00	2.0e−01	4.2e+05	2.7e+01	1.6e+02	2.7e+01	7.4e−03	5.0e−02	1.2e+01	8.3e+01	6.8e+00	3.1e+03	9.3e−01
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	1.4e+01	4.0e+02	1.7e+02	2.6e+02	9.0e+02	9.0e+02	9.0e+02	5.0e+02	9.0e+02	5.3e+02	2.0e+02	2.1e+02	
	IQR	2.8e−01	5.0e+01	1.8e+02	3.3e+01	3.8e−01	4.4e−01	4.4e−01	0.0e+00	1.8e+01	6.0e−04	3.1e−12	7.1e−01	
	Mean	1.3e+01	3.6e+02	1.6e+02	2.7e+02	9.0e+02	9.0e+02	9.0e+02	5.0e+02	9.0e+02	5.3e+02	2.0e+02	2.1e+02	
	Std	2.4e−01	1.0e+02	1.1e+02	5.9e+01	2.5e−01	6.1e−01	6.1e−01	1.9e−13	1.2e+01	6.9e−03	1.6e−12	5.4e−01	

DE/rand/∞/bin		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
	Median	0.0e+00	8.2e−04	5.0e+05	7.7e−01	2.1e+01	2.7e−01	1.1e−13	2.1e+01	1.8e+02	2.0e+02	3.9e+01	1.4e+03	1.7e+01
	IQR	5.7e−14	8.6e−04	3.9e+05	8.4e−01	2.8e+01	4.5e−01	5.7e−13	5.5e−02	1.3e+01	1.7e+01	1.5e+00	7.5e+03	1.3e+00
	Mean	1.8e−14	1.1e−03	5.6e+05	9.5e−01	3.4e+01	7.3e−01	1.8e−03	2.1e+01	1.8e+02	2.0e+02	3.9e+01	2.8e+04	1.7e+01
	Std	2.7e−14	9.3e−04	2.3e+05	8.1e−01	3.9e+01	1.3e+00	4.2e−03	4.9e−02	9.4e+00	1.1e+01	1.3e+00	6.9e+04	9.1e−01
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	1.3e+01	4.0e+02	2.3e+02	2.5e+02	9.1e+02	9.1e+02	9.1e+02	5.0e+02	8.9e+02	5.3e+02	2.0e+02	2.1e+02	
	IQR	2.4e−01	0.0e+00	1.1e+01	1.6e+01	3.7e−01	2.8e−01	5.1e−01	0.0e+00	2.3e+01	5.2e−04	0.0e+00	8.4e−01	
	Mean	1.3e+01	3.9e+02	2.4e+02	2.8e+02	9.1e+02	9.1e+02	9.1e+02	5.0e+02	9.0e+02	5.3e+02	2.0e+02	2.1e+02	
	Std	1.5e−01	9.3e+01	4.4e+01	7.7e+01	4.6e−01	2.0e−01	8.0e−01	1.2e−13	1.4e+01	7.5e−01	6.3e−13	5.8e−01	

Hybrid DE/rand/∞/bin		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
	Median	0.0e+00	7.6e−05	3.6e+05	3.0e−01	9.3e+00	1.2e−03	2.8e−14	2.1e+01	3.3e+01	2.0e+02	3.9e+01	3.5e+02	1.6e+01
	IQR	5.7e−14	6.5e−05	2.5e+05	5.3e−01	1.1e+01	1.1e−02	2.8e−14	6.8e−02	9.5e+00	2.2e+01	1.7e+00	4.3e+03	1.5e+01
	Mean	1.8e−14	1.4e−04	4.0e+05	5.7e−01	2.5e+01	2.1e−01	1.6e−03	2.1e+01	3.3e+01	1.9e+02	3.8e+01	2.2e+03	1.0e+01
	Std	2.7e−14	2.7e−04	1.8e+05	6.3e−01	5.4e+01	8.1e−01	3.8e−03	4.9e−02	8.5e+00	3.7e+01	5.1e+00	3.0e+03	7.3e+00
		F14	**F15**	**F16**	**F17**	**F18**	**F19**	**F20**	**F21**	**F22**	**F23**	**F24**	**F25**	
	Median	1.3e+01	4.0e+02	2.2e+02	2.5e+02	9.0e+02	9.0e+02	9.0e+02	5.0e+02	8.9e+02	5.3e+02	2.0e+02	2.1e+02	
	IQR	2.0e−01	2.2e+02	4.7e+01	3.0e+01	6.9e−01	3.2e−01	5.5e−01	0.0e+00	2.5e+01	7.7e−04	0.0e+00	6.4e−01	
	Mean	1.3e+01	3.4e+02	2.5e+02	2.7e+02	9.0e+02	9.0e+02	9.0e+02	5.1e+02	8.9e+02	5.3e+02	2.0e+02	2.1e+02	
	Std	2.3e−01	1.3e+02	1.1e+02	6.5e+01	1.1e+00	5.7e−01	8.8e−01	6.0e+01	1.4e+01	4.4e−04	2.9e−14	4.5e−01	

Results presented in the Tab. 1-3 indicate that DE/rand/∞ speeds up the DE algorithm on unimodal and simple multimodal functions, while it may have disadvantageous impact on some multimodal functions. Addition of an exploitative factor, as in the Hybrid DE/rand/∞, improves the overall performance and leads to comparable or better solutions than DE/rand/1.

6 Conclusions and Outlook

We showed that the difference vector distribution for the differential mutation using multiple difference vectors tends to a normal one, whose covariance matrix is proportional to the covariance matrix of the current population. This result is a generalization of DE/rand/k mutation schemes for large k and facilitates theoretical analysis of DE simplifying complicated formulas depicting differential mutation. Direct application of theoretical results led to the definition of a mutation scheme, which has continuous distribution of difference vectors, although the population size is finite. Hybridization of this operator with DE/best/1 yields an algorithm performing equally as good or better than DE/rand/1 on a wide range of functions. DE/rand/∞ mutation is easy to implement, and may be incorporated into various DE variants presumably improving their performance.

References

1. Eiben, A.E.: Multi-parent recombination. In: Back, T., Fogel, D., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation, pp. C3.7:1–C3.7:10. IOP Publishing/Oxford University Press (1995)
2. Engelbrecht, A.P.: Computational Intelligence an Introduction. Wiley, Chichester (2007)
3. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
4. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. In: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
5. Neri, F., Tirronen, V.: Recent advances in differential evolution: a survey and experimental analysis. Artif. Intell. Rev. 33(1-2), 61–106 (2010)
6. Price, K.V., Storn, R.M., Lampien, J.A.: Differential evolution a practical approach to global optimization. Springer, Heidelberg (2005)
7. Ronkkonen, J., Kukkonen, S., Price, K.V.: Real-parameter optimization with differential evolution. In: IEEE Congress on Evol. Comp., pp. 506–513 (2005)
8. Storn, R.M.: Differential evolution homepage (2010) (last used: April 2010)
9. Suganthan, P., Hansen, N., Liang, J., Deb, K., Chen, Y., Auger, A., Tiwari, S.: Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report (2005) (last used: April 2010)

General Lower Bounds for the Running Time of Evolutionary Algorithms

Dirk Sudholt

International Computer Science Institute, Berkeley, CA 94704, USA

Abstract. We present a new method for proving lower bounds in evolutionary computation based on fitness-level arguments and an additional condition on transition probabilities between fitness levels. The method yields exact or near-exact lower bounds for LO, OneMax, and all functions with a unique optimum. All lower bounds hold for every evolutionary algorithm that only uses standard mutation as variation operator.

1 Introduction

Rigorous running time analysis has emerged as an important and active area in evolutionary computation. Results obtained by mathematical arguments help to judge the performance of evolutionary algorithms (EAs) on interesting problems and they can be used to compare different algorithms in a rigorous way. Running time analyses have been performed for many pseudo-Boolean functions [1] as well as for many problems from combinatorial optimization [2].

We contribute to this development with a new method for proving lower bounds for the running time of stochastic search algorithms. This method is applied to a very broad class of evolutionary algorithms for pseudo-Boolean optimization. The resulting bounds are not only tight in an asymptotic sense. They contain best possible leading constants when compared to upper bounds for the well-known (1+1) EA (see page 126 for a definition). We also make an effort towards stating bounds with precise constants for all involved terms, without resorting to asymptotic notation.

2 Previous Work

There is a long history of results on pseudo-Boolean optimization, including lower bounds. Already Droste, Jansen, and Wegener [3] presented a lower bound of $\Omega(n \log n)$ for the (1+1) EA on every n-bit pseudo-Boolean function with unique global optimum. The constant factor preceding the $n \log n$-term is $1/2 \cdot (1 - e^{-1/2}) \approx 0.196$. Wegener [1] mentions a lower bound $(1 - \varepsilon) \cdot n \ln n - cn$ where $\varepsilon > 0$ is an arbitrarily small constant and the constant $c > 0$ depends on ε. Doerr, Fouz, and Witt [4] presented a lower bound $(1 - o(1))en \ln n$ for the (1+1) EA on the function $\text{OneMax}(x) := \sum_{i=1}^{n} x_i$. This result was extended by Doerr, Johannsen, and Winzen [5]. They proved that the same bound holds for the (1+1) EA on every function with a unique global optimum.

The function LeadingOnes, shortly $\text{LO}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_i$, is another popular test function that counts the number of leading ones in the bit string. Droste, Jansen, and Wegener [3] showed that the running time of the (1+1) EA on LO is at least $c_1 n^2$ with probability $1 - 2^{-\Omega(n)}$, for some constant $c_1 > 0$.

Black-box complexity of search algorithms as introduced by Droste, Jansen, and Wegener [6] is another method for proving lower bounds. These bounds hold for all algorithms in a black-box setting where only the class of functions to be optimized is known, but the precise instance is hidden from the algorithm. Their results imply that *every* black-box algorithm needs at least $\Omega(n/\log n)$ function evaluations to optimize OneMax and LO (or, to be more precise, straightforward generalizations to function classes). Very recently Lehre and Witt [7] presented a more restricted black-box model. If only unary operators are used that are unbiased w. r. t. bit values and bit positions, every black-box algorithm needs $\Omega(n \log n)$ function evaluations for every function with a unique global optimum.

Recently, drift analysis has received a lot of attention [8,9,10]. Assume a non-negative potential function such that the optimum is reached only if the potential is 0. If the expected decrease ("drift") of the potential in one generation is bounded from below, an upper bound on the expected optimization time follows. Conversely, an upper bound on the drift implies lower bounds on the expected optimization time. If there is a drift pointing away from the optimum on a part of the potential's domain then exponential lower bounds can be shown [8].

In this work we show that also a more direct approach is sufficient for proving good lower bounds. We introduce the new lower-bound method in Section 4, followed by applications for LO in Section 5 and OneMax in Section 6. Section 7 transfers the last result to all functions with a unique optimum.

3 Preliminaries

The presentation in this work is for maximization problems, but it can be easily adapted for minimization. The technique for proving lower bounds will be applied to a very general class of evolutionary algorithms. It contains all EAs that generate $\mu \in \mathbb{N}$ individuals uniformly at random and afterwards only use standard mutations to generate offspring (see Algorithm 1). The optimization time is given by the time index t that counts the number of function evaluations.

The parent selection mechanism is very general as any mechanism based on the time index t and fitness values of previous search points may be used.

Algorithm 1. Scheme of a mutation-based EA

```
1: create $\mu$ individuals $x_1, \dots, x_\mu \in \{0,1\}^n$ uniformly at random.
2: let $t := \mu$.
3: loop
4:    select a parent $x \in \{x_1, \dots, x_t\}$ according to $t$ and $f(x_1), \dots, f(x_t)$.
5:    create $x_{t+1}$ by copying $x$ and flipping each bit independently with prob. $1/n$.
6:    let $t := t + 1$.
7: end loop
```

Any mechanism for managing a population fits in this framework. This includes parent populations and offspring populations with arbitrary selection strategies and even parallel evolutionary algorithms with spatial structures and migration.

The (1+1) EA is a well-known special case with population size $\mu = 1$. It maintains a single individual x and in every iteration it creates x' by standard mutation of x and replaces x by x' if $f(x') \geq f(x)$ (for maximization problems). We denote by (1+1) EA$_\mu$ a generalization of the (1+1) EA that is initialized with a best individual out of μ individuals generated uniformly at random.

We review the *fitness-level method*, also known as the *method of f-based partitions* [1]. It yields upper bounds for EAs whose best fitness value in the population never decreases. We call these algorithms *elitist EAs*. The *optimization time* is the number of function evaluations until a global optimum is found.

Theorem 1 (Fitness-level method for proving upper bounds). *For two sets $A, B \subseteq \{0,1\}^n$ and fitness function f let $A <_f B$ if $f(a) < f(b)$ for all $a \in A$ and all $b \in B$. Consider a partition of the search space into non-empty sets $A_1, \dots, A_m$ such that $A_1 <_f A_2 <_f \cdots <_f A_m$ and A_m only contains global optima. For a mutation-based EA $\mathcal{A}$ we say that $\mathcal{A}$ is in A_i or on level i if the best individual created so far is in A_i. Consider some elitist EA $\mathcal{A}$ and let s_i be a lower bound on the probability of creating a new offspring in $A_{i+1} \cup \cdots \cup A_m$, provided $\mathcal{A}$ is in A_i. Then the expected optimization time of $\mathcal{A}$ on f (without the cost of initialization) is bounded by*

$$\sum_{i=1}^{m-1} P(\mathcal{A} \text{ starts in } A_i) \sum_{j=i}^{m-1} \frac{1}{s_i} \;\leq\; \sum_{i=1}^{m-1} \frac{1}{s_i}.$$

The *canonical partition* is the one in which A_i contains exactly all search points with fitness i. It is known that for LO the method applied to the canonical partition yields an upper bound of $\sum_{i=0}^{n-1} en = en^2$ for the (1+1) EA since the probability of finding an improvement is lower bounded by the probability of flipping the first bit with value 0. This probability is at least $1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$. For OneMax we get an upper bound of $\sum_{i=0}^{n-1} en/(n-i) = en \sum_{i=1}^{n} 1/i \leq en \ln n + O(n)$ for the (1+1) EA since on level i there are $n-i$ 1-bit mutations that flip a 0-bit to 1 and hence improve the fitness.

4 Lower Bounds with Fitness-Levels

The best lower bounds with fitness-level arguments known so far were presented by Wegener in [1], assuming fitness levels $A_1, \dots, A_m$ for some fitness function f:

Lemma 1. *Let u_i be an upper bound on the probability of an EA $\mathcal{A}$ creating a new offspring in $A_{i+1} \cup \cdots \cup A_m$, provided $\mathcal{A}$ is in A_i (where "$\mathcal{A}$ is in A_i" is defined as in Theorem 1). Then the expected optimization time of $\mathcal{A}$ on f is at least*

$$\sum_{i=1}^{m-1} P(\mathcal{A} \text{ starts in } A_i) \frac{1}{u_i}.$$

The resulting lower bounds are very weak since we only look at the time it takes to leave the initial fitness level and then pessimistically assume that the optimum is found.

Making an additional assumption about the transition probabilities between fitness levels allows for much better lower bounds. In the following theorem $\gamma_{i,j}$ reflects the conditional probability of jumping from level i to level j, given that the algorithm leaves level i.

Theorem 2. *Consider a partition of the search space into non-empty sets $A_1, \ldots, A_m$ such that only A_m contains global optima. For a mutation-based EA $\mathcal{A}$ we again say that $\mathcal{A}$ is in A_i or on level i if the best individual created so far is in A_i. Let the probability of traversing from level i to level j in one step be at most $u_i \cdot \gamma_{i,j}$ and $\sum_{j=i+1}^{m} \gamma_{i,j} = 1$. Assume that for all $j > i$ and some $0 < \chi \leq 1$ it holds*

$$\gamma_{i,j} \geq \chi \sum_{k=j}^{m} \gamma_{i,k}. \tag{1}$$

Then the expected optimization time of $\mathcal{A}$ on f is at least

$$\sum_{i=1}^{m-1} P(\mathcal{A} \text{ starts in } A_i) \cdot \left(\frac{1}{u_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{u_j} \right) \tag{2}$$

$$\geq \sum_{i=1}^{m-1} P(\mathcal{A} \text{ starts in } A_i) \cdot \chi \sum_{j=i}^{m-1} \frac{1}{u_j}. \tag{3}$$

If the same fitness levels are used and $s_i = u_i$ for all levels then (3) matches the upper bound from Theorem 1 up to a factor of χ.

Proof of Theorem 2. The second bound immediately follows from the first one since $0 \leq \chi \leq 1$. Let E_i be the minimum expected remaining optimization time, where the minimum is taken for all possible histories $x_1, \ldots, x_t$ of previous search points with $x_1, \ldots, x_t \in A_1 \cup \cdots \cup A_i$. By definition $E_1 \geq E_2 \geq \cdots \geq E_m = 0$ as the conditions on the histories are subsequently relaxed. By the law of total expectation the unconditional expected optimization time is at least $\sum_{i=1}^{m-1} \mathrm{P}(\mathcal{A} \text{ starts in } A_i) \cdot E_i$, hence we only need to bound E_i. The probability of leaving level i is at most u_i and the waiting time for this event is at least $1/u_i$. In case level i is left, the remaining time depends on the new level. We have

$$E_i \geq \frac{1}{u_i} + \sum_{j=i+1}^{m-1} \gamma_{i,j} \cdot E_j.$$

Assume for an induction that for all $j > i$ it holds $E_j \geq \frac{1}{u_j} + \chi \sum_{k=j+1}^{m-1} \frac{1}{u_k}$. Then E_i is at least

$$\frac{1}{u_i} + \sum_{j=i+1}^{m-1} \gamma_{i,j} \cdot \left(\frac{1}{u_j} + \chi \sum_{k=j+1}^{m-1} \frac{1}{u_k} \right) = \frac{1}{u_i} + \sum_{j=i+1}^{m-1} \frac{1}{u_j} \left(\gamma_{i,j} + \chi \sum_{k=i+1}^{j-1} \gamma_{i,k} \right)$$

where the equality holds since on the left-hand side every term $1/u_k$ in the inner sum appears for all summands $i+1, \dots, j-1$ in the outer sum, weighted by $\gamma_{i,k}\chi$. Using the preconditions on the γ-values, the last term equals

$$\frac{1}{u_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{u_j} \left(\frac{\gamma_{i,j}}{\chi} + 1 - \sum_{k=j}^{m} \gamma_{i,k} \right) \geq \frac{1}{u_i} + \chi \sum_{j=i+1}^{m-1} \frac{1}{u_j}. \qquad \square$$

Note that in order to apply the theorem, we only have to find an upper bound $u_i \cdot \gamma_{i,j}$ on the probability of jumping from level i to level j. In particular, we can allow ourselves some slack in the definition of u_i, which can make it much easier to prove the desired condition. Also note that the theorem does not require the sets A_i to form fitness levels. The only requirement is that all global optima are contained in A_m. Furthermore, "global optima" can be replaced by any other optimization goal such as finding high-fitness individuals.

5 A Lower Bound for LeadingOnes

Our first application is for LO as here the γ-values can be estimated in a very natural way.

Theorem 3. *Let X_μ be a random variable that describes the maximum* LO*-value among μ individuals created independently and uniformly at random. For every $n \geq 2$ the expected optimization time of every mutation-based EA on* LO *is at least*

$$\sum_{i=0}^{n-1} P(X_\mu = i) \cdot n \left(\left(1 - \frac{1}{n}\right)^{-i} + \frac{1}{2} \sum_{j=i+1}^{n-1} \left(1 - \frac{1}{n}\right)^{-j} \right) \tag{4}$$

$$\geq \frac{e-1}{2} \cdot n^2 - 4n \log n. \tag{5}$$

Proof. Consider the canonical partition and assume that the algorithm is on level $i < n$. This implies that in the best individual created so far the first $i+1$ bits are predetermined. In addition, in all individuals created so far the bits at positions $i+2, \dots, n$ have not contributed to the fitness yet. These bits have been initialized uniformly at random and they have been subjected to random mutations. It is easy to see that this again results in uniform random bits. More precisely, the probability that a specific bit j with $j \geq i+2$ in a specific individual has a specific bit value 0 or 1 is exactly $1/2$ (see the proof of Theorem 17 in Droste, Jansen, and Wegener [3]).

Consider an individual x that has been selected as parent among the created individuals. Let $\text{LO}(x) = j \leq i$. We bound the probability of creating an offspring with k leading ones for some $i+1 \leq k \leq n$. One necessary condition is that the first j leading ones do not flip, which happens with probability $(1-1/n)^j$. The bit at position $j+1$ is 0, hence it must be flipped. All bits at positions $j+2, \dots, i+1$ must obtain the value 1 in the offspring. This probability is determined by the number of ones among these bits. But clearly $(1-1/n)^{i-j}$ is a lower bound on this probability since this reflects the best-case scenario that all these bits are 1 in the parent. (Since $n \geq 2$ the probability of flipping a bit is not larger than the probability of not flipping it.) The last necessary condition is to create exactly $k-1-i$ ones among at positions $i+2, \dots, n$. By the preceding arguments on the "randomness" of these bits, the probability of creating exactly $k-1-i$ ones is $2^{-k+i} := \gamma_{i,k}$ if $k < n$ and $2^{-k+i+1} := \gamma_{i,k}$ if $k = n$. Putting everything together, we have that $\left(1-\frac{1}{n}\right)^i \cdot \frac{1}{n} \cdot \gamma_{i,k}$ is an upper bound on the probability of jumping to level k.

Checking the condition on the γ-values, $\sum_{k=i+1}^{n} \gamma_{i,k} = \sum_{k=i+1}^{n-1} 2^{-k+i} + 2^{-n+i+1} = 1$ and for all $i < j \leq n$ condition (1) holds since

$$\sum_{k=j}^{n} \gamma_{i,k} = \sum_{k=j}^{n-1} 2^{-k-i-1} + 2^{-n-i} = 2^{-j+i+1} = 2\gamma_{i,j}.$$

Setting $\chi = 1/2$, the preconditions for Theorem 2 are fulfilled. Using $u_i := (1-1/n)^i \cdot 1/n$, this proves the bound

$$\sum_{i=0}^{n-1} \mathrm{P}\left(X_\mu = i\right) \cdot \left(n \cdot \left(1-\frac{1}{n}\right)^{-i} + \frac{1}{2} \sum_{j=i+1}^{n-1} n \cdot \left(1-\frac{1}{n}\right)^{-j}\right)$$

and hence (4). Due to the lack of space we only sketch how to derive the second bound (5). We assume $n \geq 21$ as otherwise the claimed bound is negative. If $\mu \geq en^2$ the lower bound follows from the initialization. Otherwise, with probability at least $1-1/n$ the algorithm starts on a fitness level at most $4\log n$. Estimating the summand for $i = 4\log n$ and multiplying by $1-1/n$ yields the claimed bound. The calculations show that the leading constant $(e-1)/2$ is best possible. □

Note that a term $-\Theta(n \log n)$ is, in general, necessary since with, say, $\mu = n$ an EA will start with an average of $\Theta(\log n)$ leading ones in the best search point.

For the (1+1) EA$_\mu$ $u_i\gamma_{i,j}$ is the exact probability of jumping from fitness level i to level $j > i$. Also recall that all conditions (1) on the $\gamma_{i,j}$-values hold with equality. Going through the proof of Theorem 2 again, we find that in this special setting all estimations are, in fact, equalities. This shows that the first lower bound on LO is exact for the (1+1) EA$_\mu$. This proves that among all mutation-based EAs the (1+1) EA$_\mu$ is an optimal algorithm for the function LO. For $\mu = 1$ we get the following.

Corollary 1. *The expected optimization time of the (1+1) EA on* LO *is exactly*

$$\sum_{i=0}^{n-1} 2^{-i-1} \cdot n \left(\left(1-\frac{1}{n}\right)^{-i} + \frac{1}{2} \sum_{j=i+1}^{n-1} \left(1-\frac{1}{n}\right)^{-j}\right) = \frac{\left(\frac{n}{n-1}\right)^{n-1} + \frac{1}{n} - 1}{2} \cdot n^2.$$

The factor preceding n^2 converges to $(e-1)/2$ from below. To the author's knowledge this is the first time the leading constant for the (1+1) EA on LO has been stated explicitly. This result also shows that additional information about transition probabilities between fitness levels is also useful for proving better *upper* bounds.

6 A Lower Bound for OneMax

Theorem 4. *The expected optimization time of every mutation-based EA on* OneMax *is at least* $en\ln n - 2n\log\log n - 16n$*.*

Proof. Assume that $n \geq 34$ as otherwise the claim is trivial. If $\mu \geq 2en\ln n$ then the probability that the first $2en\ln n$ search points generated during initialization find the optimum is at most $2en\ln n \cdot 2^{-n} \leq 1/2$, which establishes the lower bound $en\ln n$. In the following we assume $\mu \leq 2en\ln n$. Let $\ell = \lceil n - n/\log n\rceil$. Consider the following partition $A_\ell, \dots, A_n$. Let $A_i = \{x \mid |x|_1 = i\}$ for $i > \ell$ and A_ℓ contain all remaining search points. With probability at least $1 - 2en\log n \cdot \sum_{i=0}^{\log n}\binom{n}{i}2^{-n} \geq 1 - 1/(\log n)$ for $n \geq 34$ the initial population only contains individuals on the first fitness level. Consider a situation where the algorithm is on fitness level i, i. e., the best-so-far search point has had up to i ones. The probability of reaching any specific higher fitness level $j > i$ is maximal if an individual with i ones is selected as parent. (See Lemma 11 in [5] for a formal proof.)

For $j > i$ let $p_{i,j}$ be the probability of the event that mutating an individual with i ones results in an offspring that contains j ones. A necessary condition for this event is that in this mutation either $j-i$ 0-bits flip to 1 and no 1-bit flips to 0 or that at least $j-i+1$ 0-bits flip to 1. This yields

$$\begin{aligned}
p_{i,j} &\leq \binom{n-i}{j-i}\cdot\frac{1}{n^{j-i}}\cdot\left(1-\frac{1}{n}\right)^{n-j+i} + \binom{n-i}{j-i+1}\cdot\frac{1}{n^{j-i+1}}\\
&\leq \left(\frac{n-i}{n}\right)^{j-i}\cdot\left(\left(1-\frac{1}{n}\right)^{n-j+i}+\frac{n-i}{n}\right).
\end{aligned}$$

For $i \geq \ell$ define

$$u'_i := \frac{n-i}{n}\cdot\left(\left(1-\frac{1}{n}\right)^{n-1}+\frac{n-i}{n}\right) \quad\text{and}\quad \gamma'_{i,j} := \left(\frac{n-i}{n-1}\right)^{j-i-1}$$

where the prime indicates that these will not be the final variables used in the application of Theorem 2. Observe that

$$\begin{aligned}
u'_i\gamma'_{i,j} &= \frac{n-i}{n}\cdot\left(\left(1-\frac{1}{n}\right)^{n-1}+\frac{n-i}{n}\right)\cdot\left(\frac{n-i}{n}\cdot\frac{n}{n-1}\right)^{j-i-1}\\
&= \left(\frac{n-i}{n}\right)^{j-i}\cdot\left(\left(1-\frac{1}{n}\right)^{n-j+i}+\frac{n-i}{n}\cdot\left(\frac{n}{n-1}\right)^{j-i-1}\right)
\end{aligned}$$

$$\geq \left(\frac{n-i}{n}\right)^{j-i} \cdot \left(\left(1-\frac{1}{n}\right)^{n-j+i} + \frac{n-i}{n}\right) \geq p_{i,j}.$$

Since Theorem 4 requires the $\gamma_{i,j}$-variables to sum up to 1, we consider the following normalized variables: $u_i := u'_i \cdot \sum_{j=i+1}^{n} \gamma'_{i,j}$ and $\gamma_{i,j} := \frac{\gamma'_{i,j}}{\sum_{j=i+1}^{n} \gamma'_{i,j}}$. As $u_i\gamma_{i,j} = u'_i\gamma'_{i,j} \geq p_{i,j}$, the conditions on the transition probabilities are fulfilled. The condition $\gamma_{i,j} \geq \chi \sum_{k=j}^{n} \gamma_{i,j}$ is equivalent to $\gamma'_{i,j} \geq \chi \sum_{k=j}^{n} \gamma'_{i,j}$ and

$$\sum_{k=j}^{n} \gamma'_{i,k} = \sum_{k=j}^{n} \left(\frac{n-i}{n-1}\right)^{k-i-1} \leq \left(\frac{n-i}{n-1}\right)^{j-i-1} \sum_{k=0}^{\infty} \left(\frac{n-i}{n-1}\right)^{k} = \gamma'_{i,j} \cdot \frac{n-1}{i-1}.$$

Using $i \geq n - n/\log n$, we have

$$\frac{i-1}{n-1} \geq \frac{n-n/\log n - 1}{n-1} = 1 - \frac{n/\log n - 1}{n-1} \geq 1 - \frac{2}{\log n}.$$

Hence, choosing $\chi := 1 - 2/\log n$ we obtain $\sum_{k=j}^{n} \gamma'_{i,k} \leq \gamma'_{i,j} \cdot (n-1)/(i-1) \leq \gamma'_{i,j}/\chi$ as required. Now that all conditions are verified, we proceed by estimating the variables u_i. Bounding the sum of the $\gamma'_{i,j}$-values as before,

$$\sum_{j=i+1}^{n} \gamma'_{i,j} \leq \sum_{j=0}^{\infty} \left(\frac{n-i}{n-1}\right)^{j} \leq \frac{n-1}{i-1} \leq \frac{1}{1-\frac{2}{\log n}}.$$

Hence

$$\begin{aligned}
u_i &\leq \frac{n-i}{n} \cdot \left(\left(1-\frac{1}{n}\right)^{n-1} + \frac{n-i}{n}\right) \cdot \frac{1}{1-\frac{2}{\log n}} \\
&\leq \frac{n-i}{en} \cdot \left(1 + \frac{1}{n-1} + \frac{e(n-i)}{n}\right) \cdot \frac{1}{1-\frac{2}{\log n}} \\
&\leq \frac{n-i}{en} \cdot \left(1 + \frac{3}{\log n}\right) \cdot \frac{1}{1-\frac{2}{\log n}} \\
&\leq \frac{n-i}{en} \cdot \frac{1}{1-\frac{3}{\log n}} \cdot \frac{1}{1-\frac{2}{\log n}} \leq \frac{n-i}{en} \cdot \frac{1}{1-\frac{5}{\log n}}.
\end{aligned}$$

Applying Theorem 2 and recalling that the algorithm is initialized on the first fitness level with probability at least $1 - \frac{1}{\log n}$ yields the upper bound

$$\left(1-\frac{1}{\log n}\right)\left(1-\frac{2}{\log n}\right)\sum_{i=\ell}^{n-1} \frac{en}{n-i}\left(1-\frac{5}{\log n}\right) \geq \left(1-\frac{8}{\log n}\right) en \sum_{i=1}^{\lfloor n-\ell \rfloor} \frac{1}{i}.$$

Since $\sum_{i=1}^{\lfloor r \rfloor} 1/i \geq \ln r$ for any $r \in \mathbb{R}^+$, the bound is at least

$$\begin{aligned}
\left(1-\frac{8}{\log n}\right) en \ln(n/\log n) &= \left(1-\frac{8}{\log n}\right) en(\ln(n) - \ln(\log n)) \\
\geq en \ln n - en\ln(\log n) - 8en \cdot \frac{\ln(n)}{\log n} &> en\ln n - 2n\log\log n - 16n. \qquad \square
\end{aligned}$$

We remark that the lower bound does not hold for all mutation operators. The biased mutation operator in [11] leads to a bound $\Theta(n)$ for the (1+1) EA on OneMax.

7 Generalization to All Functions with Unique Optimum

Using arguments by Doerr, Johannsen, and Winzen [5], we now transfer the lower bound for OneMax to all functions with a unique global optimum. This yields a more precise result than the $\Omega(n \log n)$ bound by Lehre and Witt [7].

In [5] the authors proved that the expected optimization time of the (1+1) EA on OneMax is not larger than the expected optimization time of the (1+1) EA on any other function with unique global optimum. Their proof extends to arbitrary mutation-based EAs. As space is limited, we only sketch the proof of the following theorem.

Theorem 5. *The expected number of function evaluations for every mutation-based EA $\mathcal{A}$ on every function f with a unique global optimum is at least $en \ln n - 2n \log\log n - 16n$.*

Sketch of Proof. Observe that all mutation-based EAs are invariant under transformations that consistently invert bit values for specific bits. Hence, we assume w. l. o. g. that 1^n is the global optimum of f. Let $E^f_{\mathcal{A}}$ denote the expected optimization time of $\mathcal{A}$ on f and assume that $\mathcal{A}$ has already created search points $x_1, \dots, x_t$. Let $E^f_{\mathcal{A}}(i)$ be the minimum expected optimization time for $\mathcal{A}$ on f given that $x_1, \dots, x_t \in A_0 \cup \dots \cup A_i$ with $A_0, \dots, A_n$ the canonical partition for OneMax. Observe that by definition, since the conditions are subsequently restricted, $E^f_{\mathcal{A}}(n) \leq E^f_{\mathcal{A}}(n-1) \leq \dots \leq E^f_{\mathcal{A}}(0)$. Further define a more specific and slightly modified quantity for the (1+1) EA$_\mu$: let $\widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i)$ be defined like $E^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i)$, but with the additional condition that the history $x_1, \dots, x_t$ contains at least one search point in A_i. Since we have only added a constraint, $\widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i) \geq E^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i)$.
Following Doerr, Johannsen, and Winzen [5], an inductive proof shows that for all i it holds $E^f_{\mathcal{A}}(i) \geq \widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i)$. Clearly $E^f_{\mathcal{A}}(n) \geq \widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(n) = 0$. Assume $E^f_{\mathcal{A}}(j) \geq \widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(j)$ for all $j > i$. Depending on the outcome of the next offspring creation, $E^f_{\mathcal{A}}(i)$ can be expressed by transition probabilities to higher fitness levels $k > i$ and $E^f_{\mathcal{A}}(k)$ as a lower bound on the conditional expected remaining time. By induction $E^f_{\mathcal{A}}(k) \geq \widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(k)$. Moreover, the best case for the transition probabilities is attained when a parent with i ones is mutated. This gives a lower bound for $E^f_{\mathcal{A}}(i)$ which exactly describes the expected running time of the (1+1) EA$_\mu$ on level i, hence $E^f_{\mathcal{A}}(i) \geq \widetilde{E}^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i) \geq E^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}(i)$.

As $\mathcal{A}$ and (1+1) EA$_\mu$ are initialized in the same way, they share the same distribution for the initial fitness level. We conclude $E^f_{\mathcal{A}} \geq E^{\text{OneMax}}_{(1+1)\text{ EA}_\mu}$ and the bound follows from Theorem 4 applied to (1+1) EA$_\mu$. $\square$

As a side result, the proof has also shown that the (1+1) EA$_\mu$ is an optimal algorithm for OneMax.

8 Conclusions

Using an adaptation of the fitness-level method, we have presented general lower bounds for the running time of mutation-based evolutionary algorithms in pseudo-Boolean optimization. The bounds for LO and OneMax are exact or exact up to lower-order terms when compared to upper bounds for the (1+1) EA. This is a rare occasion of results that are both very general and very precise at the same time. In addition, we have proven that among all mutation-based EAs the (1+1) EA$_\mu$ (for proper μ) is an optimal algorithm for LO and OneMax, with respect to the number of function evaluations.

The method itself is not restricted to the investigated setting. It is ready to be applied to other search spaces and further stochastic search algorithms.

Acknowledgment. The author was supported by a postdoctoral fellowship from the German Academic Exchange Service.

References

1. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In: Sarker, R., Yao, X., Mohammadian, M. (eds.) Evolutionary Optimization, pp. 349–369. Kluwer, Dordrecht (2002)
2. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4(3), 281–293 (2007)
3. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
4. Doerr, B., Fouz, M., Witt, C.: Quasirandom evolutionary algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 1457–1464. ACM, New York (2010)
5. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1967–1974. IEEE, Los Alamitos (2010)
6. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory of Computing Systems 39(4), 525–544 (2006)
7. Lehre, P.K., Witt, C.: Black box search by unbiased variation. In: Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 1441–1448. ACM, New York (2010)
8. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 82–91. Springer, Heidelberg (2008)
9. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)
10. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. In: Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 1449–1456. ACM, New York (2010)
11. Jansen, T., Sudholt, D.: Analysis of an asymmetric mutation operator. Evolutionary Computation 18(1), 1–26 (2010)

A Binary Encoding Supporting Both Mutation and Recombination

Karsten Weicker

HTWK Leipzig Applied University, Leipzig, Germany
weicker@imn.htwk-leipzig.de

Abstract. There has been a long debate on the "most important" operator when applying genetic algorithms. This is very closely related to the favorite binary encoding, namely standard binary or Gray. Rather than confronting both approaches, this article is motivated by the search for an encoding that supports both mutation and recombination. For this purpose an encoding scheme is proposed and evaluated both using metrics and experiments.

1 Motivation

There exists a long rather dogmatic debate on how genetic algorithms really work and whether mutation or crossover is the most important operator [9,12]. As a consequence the idea is appealing to "re-design" the standard genetic algorithm by integrating the different contradicting views. In this contribution we re-consider the different coding schemes in use and propose a new one.

On the one hand, there is the traditional view: Crossover is the primary operator and the schema processing enables a distributed knowledge within the population combined with a pseudo-parallel search [10]. The standard binary code supports the schema processing ideally, cf. [14] and others.

On the other hand, there is a lot of evidence that the mutation can be more than a mere background operator [12,8]. Following this philosophy, the Gray code becomes interesting since it removes Hamming cliffs [1] and induces fewer local minima [17]. The Gray code embeds the phenotypical neighborhood into the bit flipping neighborhood which means that neighboring values have a Hamming distance of 1. However, the Gray code interferes with the crossover operator [2].

In this contribution we try to find an encoding scheme that supports both mutation and crossover. For comparison and evaluation, we restrict ourselves to 6-bit encodings and a small set of rather simple test functions.

2 State of the Art Concerning Binary Encodings

Rothlauf [14] has examined all possible 3-bit encodings for selecto-recombinative GAs, i.e. without mutation. Using an integer one-max problem he showed the standard binary code to be superior to the Gray code. However he also demonstrates that there are even better encodings.

Another important fundamental comparison was undertaken by Chakraborty and Janikow [3] who used a Markov chain analysis for all possible 3-bit problems. They classified the problems according to the number of local optima (within the phenotype as well as induced by the encoding). Although the standard binary code proves to be better on a small number of problems, there is a class of problems where Gray code is superior. However, the induced number of optima appears to be no reliable criterion.

There are several approaches and ideas to boost the coding. First, Caruana et. al. [2] argued that multiple representations for the different operators might be favorable. As a consequence, when mutating, the individual could be transfered into the Gray code, be bit-flipped, and be transfered back. Multiple representations are also used by Rana and Whitley [13]. Here the algorithm is able to switch between different Gray codes. This concept should reduce premature convergence in local optima and does not aim at better schema processing.

A completely new coding scheme, named E-code, was proposed by Eaton and Collins [7,4]. They assigned the phenotypic values in increasing order to the genotypes with increasing number of ones. This is shown in Figure 1. The authors argued that the E-code minimizes the disruptive effects of the mutation operator on schemata with high fitness and circumstantiate this with an analysis using Toeplitz matrices and an empirical investigation. However, a visualization of the neighborhood relationship for a 1-bit mutation shows that E-code (in Figure 2) does not embed the phenotypic neighbors into the genotype as can be seen in the missing marks next to the principal diagonal. For comparison the neighborhood of the other coding schemes are shown: Gray code (Figure 3) embeds the phenotypic neighborhood ideally and Figure 4 reveals the Hamming cliffs induced by the standard binary code clearly.

In a preliminary work [16] all 4-bit codes have been investigated whether they embed the phenotype structure and show good schema behavior. It is shown that there are Gray-like codings that combine good performance for mutation and crossover—without giving insight in how the codes work structurally.

3 Requirements for a "Good" Binary Encoding

Informally, the requirements are fairly clear: both mutation and crossover should be able to produce good individuals. However, we need precise metrics to assess the coding schemes. Since there are $2^{\ell!}$ codings for ℓ-bit numbers, we do not want to rate the codings empirically. Rather we propose a set of analytical metrics that are used for the selection of interesting codings in the first instance.

Concerning the support for the mutation, we use the relative Toeplitz matrix

$$T_{ij} = \|d_{euklid}(i,j) - d_{ham}(i,j)\|$$

like in [11,4]. However, for assessing the embedment of phenotypic neighbors we put in our metric more emphasis on the closely related values:

$$PhenoEmbed = \sum_{i<j} \frac{T_{ij}}{d_{euklid}(i,j)}.$$

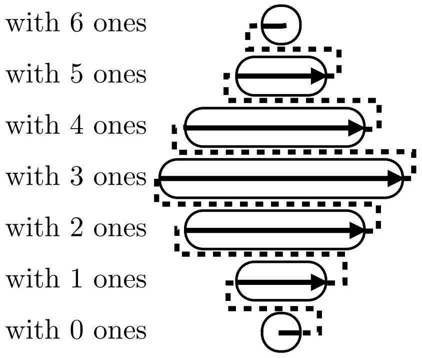


Fig. 1. Idea for the generation of the E-code

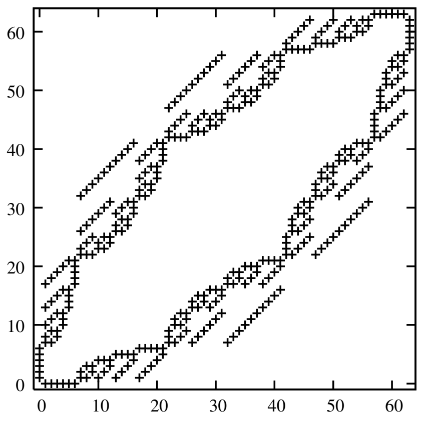


Fig. 2. Neighboring phenotypes for E-code and 1-bit mutation

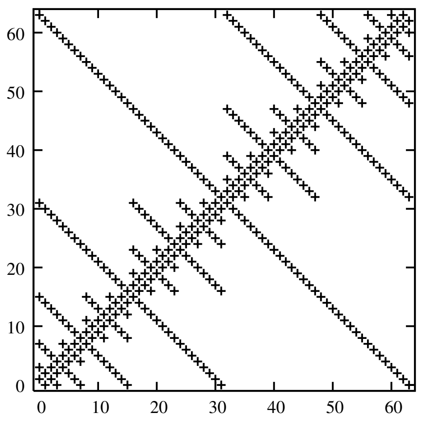


Fig. 3. Neighboring phenotypes for Gray code and 1-bit mutation

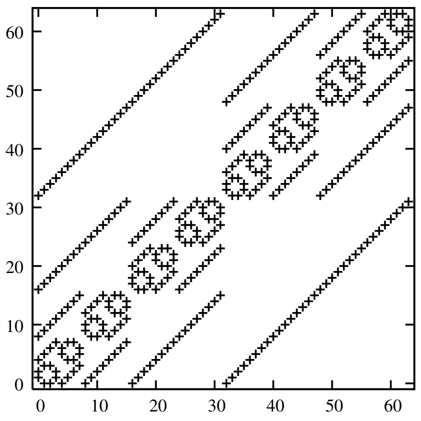


Fig. 4. Neighboring phenotypes for standard binary code and 1-bit mutation

The smaller the value is, the better the neighborhood is embedded. Additionally, we will also look at the L1, L2, and L3 norms defined as

$$L(x)Norm = \sqrt[x]{\sum_{i<j} (T_{ij})^x}$$

which delivers insight into how many points are remapped by the coding [11,4].

Concerning the support for the recombination, we are interested in the property of high quality schema to lead the search towards the optimum. Let $\tilde{x}$ be the optimum and H a schema with the optimum in the class of described individuals ($\tilde{x} \in \mathcal{I}(H)$). Then $\mathcal{H}_H$ is the set of all schemata with the same defining positions as H. This set contains all competing schemata of H, too. We require that for certain schemata the average fitness of H is better than the fitness of each schema $H' \in \mathcal{H}_H \setminus \{H\}$. In our investigation the interesting schemata are

$$Schemata = \bigcup_{k \in \{1,2,3\}} Schemata_k$$

where $Schemata_k$ denotes all optimum schemata with order k and defining length $k-1$. Then, the support of an encoding for schemata is measured by

$$SchemaSupp_f = \|\{H' \mid H \in Schemata \wedge H' \in \mathcal{H}_H \wedge \bar{f}_{H'} \geq \bar{F}_H\}\|$$

in case of a maximization problem f. The smaller the value $SchemaSupp_f$ is, the better the coding is supposed to support the crossover operator. Similar inspections have been made in many articles on deceptiveness [5,6] and related work [14].

We consider the following three fitness functions within the schema analysis ($x \in \{0, \ldots, 63\}$):

$$f_1(x) = x \qquad f_2(x) = 63 - 2 \cdot |x - 32| \qquad f_3(x) = \begin{cases} 63 - |x - 48|, & \text{if } x \leq 48 \\ 59 - 4 \cdot (x - 49), & \text{if } x > 49 \end{cases}$$

4 Constructing a New Coding

Complete examinations of all codings have been executed for three [14] and four [16] bits, so far. However, the encoded range and granularity appear to be too small even in the case of 4-bit encodings—certain effects do not show, like the benefit of mutations jumping a far distance in the phenotypic space. Therefore, in this contribution 6-bit encodings are examined—with the drawback that we cannot consider all codings.

First, we develop a concept how the genotypes could be assigned to the phenotypic values in a sensible way. The presented general idea leaves certain degrees of freedom on the details of construction. Therefore, second, a subset of the possible codings are examined.

The code is based on the following cornerstones: A good portion of the neighboring pairs of values in the phenotype should be neighbors by mutating one bit too. Moreover, we want to base the code on the E-code with increasing numbers of ones. Both cornerstones contradict each other because the distance of neighboring values is usually two within the E-code.

As a consequence, we switch between the genotypes containing k and $(k+1)$ ones in such a way that always one-bit changes, e.g. if 001100 and 101000 are assigned to phenotypes m and $(m+2)$ then phenotype $(m+1)$ has genotype 101100. Figure 5 shows how the numbers are assigned to the genotypes of the different partitions (concerning the number of ones). However, in the middle of the value range, few two-bit changes are necessary (in the center of the 3-ones partition). Because of the alternating number of ones we refer to this class of codings as *zigzag codes*.

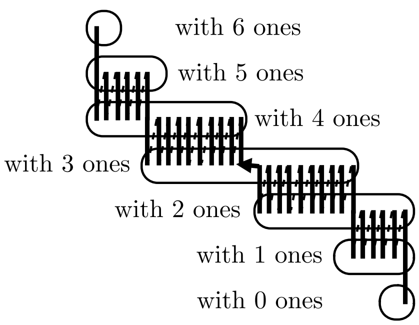


Fig. 5. Idea for the generation of the zigzag code

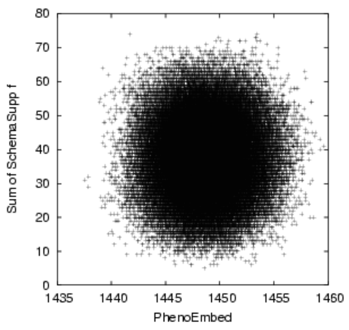


Fig. 6. All zigzag codes concerning the two analytical metrics

In the E-code the genotypes are assigned in increasing order within the partition of equal ones. We restrict our examined codes to an increasing order within the same partition too. We have computed all remaining possible 94614 zigzag codes that are evaluated analytically in the next section using the given metrics.

5 Analytical Results

For 6-bit integers, the metrics have been computed for the standard binary encoding, Gray code, E-code, and all zigzag codes introduced in the previous section. The latter results are displayed in Figure 6. We have picked three zigzag codes for further evaluation: the one with the best value for schema support (zigzag1), the one with the best value for phenotypic embedment (zigzag2), and a member from the Pareto front that appears to be a good compromise (zigzag3). The exact results for these encodings are displayed in Table 1. As can be seen clearly, all zigzag codes in Figure 6 exhibit a better phenotypic embedment than the other codings. In addition zigzag1 shows also schema support better than the standard binary coding.

The neighborhood of zigzag1 is shown in Figure 7, the neighborhood of zigzag3 in Figure 8 (zigzag2 is very similar to zigzag3). Clearly the embedding of neighboring phenotypes can be seen as well as the two-bit differences around

Table 1. Results for the analyzed coding schemes

	SchemaSupp				Toeplitz matrices			
coding	f_1	f_2	f_3	*PhenoEmbed*	L1	L2	L3	entries $\neq 0$
std.bin.	0	0	6	1570.96	37734.0	1062.07	343.50	1942
Gray	38	0	27	1488.45	37536.0	1062.07	344.97	1861
E-code	0	25	8	1597.77	37858.0	1063.92	343.60	1950
zigzag1	0	0	5	1448.58	37552.0	1065.07	344.62	1792
zigzag2	0	9	6	1439.77	37550.0	1065.52	344.58	1781
zigzag3	0	3	6	1441.45	37550.0	1065.50	344.60	1782

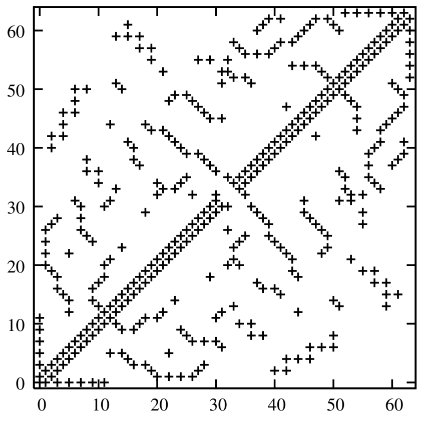


Fig. 7. Neighboring phenotypes for zigzag1 code and 1-bit mutation

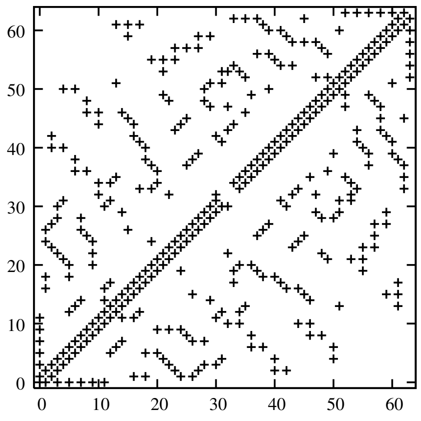


Fig. 8. Neighboring phenotypes for zigzag3 code and 1-bit mutation

phenotypes 31 and 32. The overall structure seems to be more scattered but show similarities to both gray code and standard binary code. Certain peculiarities like the appendices at the diagonal for phenotypes 12 and 51 are common to all zigzag codes – caused by codes with 1, 2, and 3 ones codes in a row (resp. 4/5/6 ones).

If we follow the arguments in [11], we can derive from the L(k) norms that the remapping of points has similar properties compared to the Gray code. However, the number of remapped points (entries $\neq 0$) is even smaller. As a consequence the changes introduced by the zigzag codes are bigger.

6 Experimental Comparison of the Encoding Schemes

In this section, we investigate empirically whether the zigzag codes combine the properties of standard binary and Gray code during the optimization.

We use a genetic algorithm with bit-flipping mutation (with per-bit probability $p_M = \frac{1}{\ell}$ where ℓ is the length of the genome), uniform crossover (with

Table 2. Experiments for selecto-recombinative GAs (crossover only): solved dimensions of best solution of each experiment (averaged over 50 experiments). For F_4 and F_5, 10 dimensions instead of 20 dimensions are used.

	F_1	F_2	F_3	F_4	F_5	F_6
stdbin	13.3±2.4	4.7±2.3	4.4±1.7	2.6±1.4	3.3±1.5	13.2±2.1
Gray	7.9±2.0	9.5±2.8	2.7±1.5	4.6±1.5	4.6±1.6	8.2±2.3
e-code	17.9±1.4	2.1±1.5	1.9±1.5	0.7±0.8	1.3±1.2	17.7±1.6
zigzag1	15.4±2.4	5.2±2.2	3.6±1.9	2.7±1.4	1.9±1.3	15.7±2.0
zigzag2	16.1±2.5	2.6±1.5	4.1±2.0	1.3±1.0	2.1±1.4	16.7±1.8
zigzag3	16.0±1.9	2.5±1.3	2.9±1.5	1.3±1.0	2.2±1.3	16.9±2.0

crossover rate $p_X = 0.7$) because it emphasises short schemata, populations size 100, and tournament selection (best of 3). To investigate the role of the different operators, we run additional experiments with $p_M = 0$ respective $p_X = 0$. We iterate each experiment 50 times and average the results. Optimization stops after 20000 generations or when reaching the optimum.

In this examination we use separable functions only since we want to evaluate the quality of codings. Each search space dimension encodes the values $\{0, \dots, 63\}$. The functions have the form $F_k(x_1, \dots, x_n) = \sum_{i=1}^{n} f_k(x_i)$. Besides the already introduced functions f_1, f_2, and f_3 we consider a sphere-like maximization problem with optimum at $(32, \dots, 32)$ (called f_4) and with a randomly assigned optimum $rnd_i \in \{0, \dots, 63\}$ for each dimension i and each optimization run (called f_5). Furthermore, we use a multi-modal function f_6 by disturbing f_1.

$$f_4(x) = 64^2 - (x - 32)^2 \qquad f_5(x) = 64^2 - (x - rnd)^2 \qquad f_6(x) = \lfloor x + 2 \cdot \sin(x) \rfloor$$

F_1, F_2, F_3, and F_6 are used with $n = 20$; F_4 and F_5 with $n = 10$.

Table 2 shows the results for the GAs using crossover only. The standard binary encoding outperforms Gray code clearly for F_1, F_3, and F_6. The Gray

Table 3. Experiments for mutation-only GAs: success rate (succ.), average number of generations needed to solve the problem (gen.), and average number of dimensions solved in the best solution of each experiment (solv.; note: 10 instead of 20 dimensions for F_4 and F_5)

	F_1			F_2		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	1.0	232.1±48.2	20.0±0.0	0.0	—	10.2±1.6
Gray	1.0	308.9±67.4	20.0±0.0	1.0	257.6±37.5	20.0±0.0
e-code	1.0	214.8±61.1	20.0±0.0	0.0	—	8.7±2.1
zigzag1	1.0	282.3±69.3	20.0±0.0	0.56	10278.1±5092.9	17.7±2.9
zigzag2	1.0	273.5±61.0	20.0±0.0	0.02	12756.0±0.0	14.7±1.9
zigzag3	1.0	281.3±81.6	20.0±0.0	0.1	14492.4±3518.0	15.5±2.1

	F_3			F_4		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	0.0	—	6.3±2.4	0.0	—	5.4±1.3
Gray	1.0	342.4±59.0	20.0±0.0	1.0	105.9±19.6	10.0±0.0
e-code	0.0	—	8.0±1.7	0.02	7408.0±0.0	3.4±2.0
zigzag1	1.0	503.9±90.5	20.0±0.0	1.0	4380.7±4476.0	10.0±0.0
zigzag2	1.0	544.1±104.9	20.0±0.0	0.94	8127.7±4704.7	9.8±0.7
zigzag3	1.0	539.8±105.8	20.0±0.0	0.96	8006.7±4319.8	9.9±0.5

	F_5			F_6		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	0.42	6383.3±5160.8	8.9±1.1	1.0	223.8±48.6	20.0±0.0
Gray	1.0	103.4±17.1	10.0±0.0	1.0	175.9±28.8	20.0±0.0
e-code	0.56	9873.5±5483.9	7.9±2.5	1.0	175.9±28.8	20.0±0.0
zigzag1	1.0	340.0±531.6	10.0±0.0	1.0	330.2±133.9	20.0±0.0
zigzag2	1.0	495.7±969.3	10.0±0.0	1.0	336.0±109.1	20.0±0.0
zigzag3	1.0	932.9±2861.1	10.0±0.0	1.0	311.4±107.0	20.0±0.0

Table 4. Experiments for GAs (with crossover and mutation): success rate (succ.), average number of generations needed to solve the problem (gen.), and average number of dimensions solved in the best solution of each experiment (solv.; note: 10 instead of 20 dimensions for F_4 and F_5).

	F_1			F_2		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	1.0	71.3±9.9	20.0±0.0	0.0	—	10.9±2.2
Gray	1.0	124.6±18.2	20.0±0.0	1.0	86.1±11.9	20.0±0.0
e-code	1.0	44.6±8.1	20.0±0.0	0.0	—	10.0±2.6
zigzag1	1.0	65.9±14.6	20.0±0.0	0.02	5029.0±0.0	15.3±2.2
zigzag2	1.0	62.6±14.6	20.0±0.0	0.0	—	10.7±2.1
zigzag3	1.0	62.5±14.5	20.0±0.0	0.0	—	10.7±2.1
stdbin+Gray	1.0	141.6±42.0	20.0±0.0	1.0	1646.7±913.2	20.0±0.0
	F_3			F_4		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	0.0	—	5.5±2.4	0.0	—	5.4±1.5
Gray	1.0	159.4±23.2	20.0±0.0	1.0	61.6±10.3	10.0±0.0
e-code	0.0	—	8.2±1.9	0.0	—	3.0±1.7
zigzag1	1.0	369.0±83.0	20.0±0.0	0.46	7121.9±6601.5	8.3±1.8
zigzag2	1.0	412.3±69.3	20.0±0.0	0.04	13725.0±5760.0	5.1±1.7
zigzag3	1.0	403.9±72.0	20.0±0.0	0.02	1877.0±0.0	4.5±1.7
stdbin+Gray	0.0	—	6.9±1.3	1.0	681.6±594.9	10.0±0.0
	F_5			F_6		
	succ.	gen.	solv.	succ.	gen.	solv.
stdbin	0.18	3196.7±4972.6	7.9±1.6	1.0	61.4±10.0	20.0±0.0
Gray	1.0	56.0±10.1	10.0±0.0	1.0	416.8±203.9	20.0±0.0
e-code	0.0	—	2.6±1.6	1.0	36.0±3.7	20.0±0.0
zigzag1	0.98	1396.6±2899.1	9.9±0.1	1.0	58.3±25.7	20.0±0.0
zigzag2	0.88	1076.0±2024.0	9.8±0.7	1.0	66.9±42.7	20.0±0.0
zigzag3	0.92	1647.1±2721.1	9.9±0.5	1.0	54.8±26.9	20.0±0.0
stdbin+Gray	1.0	98.2±65.4	10.0±0.0	1.0	213.2±76.6	20.0±0.0

code shows a peculiar advantage when the optimum is placed in the middle (F_2 and F_4). The zigzag codes improve the crossover for F_1 and F_6 but are still beaten by the e-code. For F_4 and F_5 the zigzag results are inferior.

Table 3 shows the results for mutation-only GAs. Here, the standard binary code is only able to outperform the Gray code for F_1. As expected, the zigzag codes (esp. zigzag1) are able to solve all problems. However, the zigzag codes are not as good performing as the best other coding scheme. Again, the Gray code is superior on F_2—one reason are the two diagonal lines (in Figure 3) leading towards the optimum in the center which reduces the probability of a fitness loss.

Table 4 shows the results for the experiments with the GA using both crossover and mutation. The standard binary encoding is not able to solve F_2, F_3, and F_4 in any experiment. Altogether, the Gray code appears to be the best choice. However, in case of the multi-modal F_6 Gray code needs substantially longer than the standard binary encoding. Again zigzag1 appears to be a good compromise:

Table 5. For all codings: mapping of the phenotypic values to the genotype (in standard binary coded form)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
std.bin.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Gray	0	1	3	2	6	7	5	4	12	13	15	14	10	11	9	8	24	25	27	26	30	31	29	28	20	21	23	22	18	19	17	16
E-code	0	1	2	4	8	16	32	3	5	6	9	10	12	17	18	20	24	33	34	36	40	48	7	11	13	14	19	21	22	25	26	28
zigzag1	0	1	3	2	6	4	12	8	24	16	48	32	36	52	20	22	18	50	34	35	33	37	5	21	17	25	9	11	10	42	40	44
zigzag2	0	1	3	2	6	4	12	8	24	16	48	32	33	37	5	13	9	41	40	42	10	26	18	50	34	38	36	52	20	21	17	25
zigzag3	0	1	3	2	6	4	12	8	24	16	48	32	36	44	40	41	33	37	5	21	20	22	18	19	17	25	9	11	10	42	34	38
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
std.bin.	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
Gray	48	49	51	50	54	55	53	52	60	61	63	62	58	59	57	56	40	41	43	42	46	47	45	44	36	37	39	38	34	35	33	32
E-code	35	37	38	41	42	44	49	50	52	56	15	23	27	29	30	39	43	45	46	51	53	54	57	58	60	31	47	55	59	61	62	63
zigzag1	41	53	49	57	56	58	26	27	19	23	7	39	38	46	14	15	13	29	28	60	61	45	47	43	59	51	55	54	62	30	31	63
zigzag2	19	43	11	15	14	46	44	60	56	57	49	51	35	39	7	23	22	30	28	29	31	27	59	58	62	54	55	53	61	45	47	63
zigzag3	50	53	52	60	28	29	13	15	7	39	35	51	49	57	56	58	26	30	14	46	62	54	55	23	31	27	59	43	47	45	61	63

yielding better results than standard binary code and, in the case of F_1 and F_6, even better than Gray code. In addition, we have tested a hybrid approach [2] where, for each mutation, the standard binary genotype is transformed to Gray code, mutated, and transformed back—with convincing results (except F_3 where the two encodings interfere with each other).

7 Conclusion

The idea of the zigzag codes leads to coding schemes that are very stable across all problems and used operators, although there was no synergy creating a superior encoding. Nevertheless, the results show that it is possible to combine the strengths of the different operators—especially against the background that this research was not exhaustive but rather concept-driven.

Nevertheless, zigzag codes have severe disadvantages. The codings do not scale with increasing number of bits [15] and there is no decoding algorithm available (except using table entries). Future research could focus on either full scalability or on building an "intelligent" (adaptive or developmental) representation from 6-bit blocks. Also the restriction on smooth fitness functions should be questioned.

The used metrics appear to be a proper means to assess the suitability of a coding scheme a-priori although there are certain effects between neighborhood of the coding and fitness functions (e.g. Gray code and F_2) that cannot be predicted.

References

1. Caruana, R.A., Schaffer, J.D.: Representation and hidden bias: Gray versus binary coding in genetic algorithms. In: Leard, J. (ed.) Proc. of the 5th Int. Conf. on Machine Learning, pp. 153–161. Morgan Kaufmann, San Mateo (1988)
2. Caruana, R.A., Schaffer, J.D., Eshelman, L.J.: Using multiple representations to improve inductive bias: Gray and binary coding for genetic algorithms. In: Segre, A.M. (ed.) Proc. of the Sixth Int. Workshop on Machine Learning, pp. 375–378. Morgan Kaufmann, San Mateo (1989)

3. Chakraborty, U.K., Janikow, C.Z.: An analysis of gray versus binary encoding in genetic search. Information Sciences 156, 253–269 (2003)
4. Collins, J.J., Eaton, M.: Genocodes for genetic algorithms. In: Proc. of Mendel 1997, pp. 23–30. PC-DIR, Brno (1997)
5. Das, R., Whitley, D.L.: The only challenging problems are deceptive: Global search by solving order-1 hyperplanes. In: Belew, R.K., Booker, L.B. (eds.) Proc. of the Fourth Int. Conf. on Genetic Algorithms, pp. 166–173. Morgan Kaufmann, San Mateo (1991)
6. Deb, K., Goldberg, D.E.: Sufficient conditions for arbitrary binary functions. Ann. Math. Artificial Intell. 10, 385–408 (1994)
7. Eaton, M., Collins, J.J.: A comparison of encoding schemes for genetic algorithms. In: Proc. of the World Congress on Neural Networks, pp. 1067–1070. INNS Press, New York (1996)
8. Forrest, S., Mitchell, M.: Relative building-block fitness and the building-block hypothesis. In: Whitley, L.D. (ed.) Foundations of Genetic Algorithms 2, pp. 109–126. Morgan Kaufmann, San Mateo (1993)
9. Goldberg, D.E.: Zen and the art of genetic algorithms. In: Schaffer, J.D. (ed.) Proc. of the Third Int. Conf. on Genetic Algorithms, pp. 80–85. Morgan Kaufmann, San Mateo (1989)
10. Holland, J.H.: Adaptation in Natural and Artifical Systems. University of Michigan Press, Ann Arbor (1975)
11. Mathias, K., Whitley, D.: Transforming the search space with Gray coding. In: IEEE Conf. onf Evolutionary Computation, pp. 513–518. IEEE Service Center, Piscataway (1994)
12. Mühlenbein, H.: How genetic algorithms really work: I. Mutation and hillclimbing. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving from Nature 2, pp. 15–25. Elsevier Science, Amsterdam (1992)
13. Rana, S.B., Whitley, L.D.: Bit representations with a twist. In: Bäck, T. (ed.) Proc. of the Seventh Int. Conf. on Genetic Algorithms, pp. 188–195. Morgan Kaufmann, San Francisco (1997)
14. Rothlauf, F.: Binary representations of integers and the performance of selectorecombinative genetic algorithms. 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. 99–108. Springer, Heidelberg (2002)
15. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Springer, Heidelberg (2006)
16. Weicker, K.: Auf der Suche nach dem Codierunsgral für genetische Algorithmen. In: Diekert, V., Weicker, K., Weicker, N. (eds.) Informatik als Dialog zwischen Theorie und Anwendung, pp. 35–44. Vieweg+Teubner, Wiesbaden (2009)
17. Whitley, D., Rana, S.: Representation, search and genetic algorithms. In: AAAI 1997: 14th National Conf. on Artificial Intelligence, pp. 497–502. AAAI Press/MIT Press (1997)

Towards Analyzing Recombination Operators in Evolutionary Search

Yang Yu, Chao Qian, and Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210093, China
{yuy,qianc,zhouzh}@lamda.nju.edu.cn

Abstract. Recombination (also called *crossover*) operators are widely used in EAs to generate offspring solutions. Although the usefulness of recombination has been well recognized, theoretical analysis on recombination operators remains a hard problem due to the irregularity of the operators and their complicated interactions to mutation operators. In this paper, as a step towards analyzing recombination operators theoretically, we present a general approach which allows to compare the runtime of an EA turning the recombination on and off, and thus helps to understand when a recombination operator works. The key of our approach is the *Markov Chain Switching Theorem* which compares two Markov chains for the first hit of the target. As an illustration, we analyze some recombination operators in evolutionary search on the LeadingOnes problem using the proposed approach. The analysis identifies some insight on the choice of recombination operators, which is then verified in experiments.

1 Introduction

Evolutionary algorithms (EAs) run a circle of reproducing offspring solutions from the current population, and then selecting to weed out bad solutions [1]. The most popular reproduction operators are *mutation* and *recombination* (also called *crossover*). Contrary to mutation operators that are defined on individual solutions, recombination operators are defined on a population of solutions, that is, they generate offspring solutions by mixing up a set of (usually two) solutions. Recombination operators were born together with the first genetic algorithm. They are a special characteristic of EAs, which makes EAs significantly differ from classical optimization techniques such as branch-and-bound strategy [17], as well as other heuristic search methods such as simulated annealing [17].

When EAs are used to tackle optimization problems, the runtime is among the central concerns. Due to recent advances in theoretical analysis of EAs, e.g. [2,9,10,20,3], a landscape of computational complexity of EAs is emerging. However, most of the previous studies focus on EAs using mutation only.

There are some recent studies on recombination operators [14,18,11,12,13,15,4,5]. Lin and Yao [14] assumed that a 'step size' is critical for recombination operators. They compared the step sizes of four recombination operators, and empirically studied an EA with these recombination operators yet without mutation operator on several problems. Spears [18] studied the construction and destruction probabilities and equilibrium of recombination using the schema theory, and concluded that the recombination operators are beneficial when there are few local optima. Several recent studies proved some effects of recombination operators. On the positive side, Jansen and Wegener [11,12] proved that, on the *Real Royal Road* problem, a recombination operator reduces the runtime of an EA from exponential to polynomial; Lehre and Yao [13] proved that, on the *TwoPath* problem of computing unique input-output sequences, a recombination operator reduces the runtime exponentially; Doerr et al. [4,5] proved that, on the *all pairs shortest path problem*, a recombination operator reduces the runtime. On the negative side, Richter et al. [15] constructed a problem called *Ignoble Trails* on which the EA using mutation only requires polynomial runtime, yet the EA using mutation with recombination requires exponential runtime. These studies disclosed some properties of recombination operators from different aspects, however, there is no general approach for theoretically analyzing recombination operators.

Usually, a recombination operator is used together with a mutation operator in EA, and a nontrivial population (more than one solution) is maintained with a selection operator. Thus, the analysis of recombination operators needs to consider the interactions between mutation and recombination operators, the effect of population size, and the effect of the selection pressure. So, it is not surprising that theoretical analysis on recombination operators is more difficult than that on mutation operators.

In this paper, as a step towards theoretical analysis of recombination operators, we try to tackle the interaction between mutation and recombination operators. We present the *Markov Chain Switching* Theorem for the comparison of two Markov chains on their first hit of the target. This theorem allows to compare the one-step behaviors of two EAs, thus can be used to compare the average runtime of an EA turning on and off the recombination operator. This theorem provides a general tool towards the understanding of the behaviors of recombination operators. As an illustration, we theoretically analyze four strategies using two different recombination operators in an EA on the LeadingOnes problem using the tool. The LeadingOnes problem is a well-studied problem for EAs using mutation only [16], yet the effect of recombination operators on the problem remains untouched. Our analysis identifies helpful and unhelpful recombination strategies from the candidates, which are then verified in experiments. It is interesting to find from the analysis that similar recombination operators can have opposite effects.

The rest of this paper is organized as follows. Section 2 introduces preliminaries, Section 3 presents the main theorem, Section 4 shows the case study on the LeadingOnes problem, and Section 5 concludes.

2 Preliminaries

We model EAs as Markov chains [9,20]. A population of an EA can be mapped to a state of a Markov chain. Formally, let $\mathcal{X}$ denote the population space containing all possible populations of an EA. A Markov chain $\{\xi_t\}_{t=0}^{+\infty}$ modeling the EA can be constructed by taking $\mathcal{X}$ as the state space, i.e., $\xi_t \in \mathcal{X}$. Let $\mathcal{X}^* \subseteq \mathcal{X}$ denote the set of all optimal populations containing at least one optimal solution. The process of an EA seeking $\mathcal{X}^*$ can be analyzed by studying the corresponding Markov chain.

Given a Markov chain $\{\xi_t\}_{t=0}^{+\infty}(\xi_t \in \mathcal{X})$ and a target subspace $\mathcal{X}^* \subset \mathcal{X}$, $\{\xi_t\}_{t=0}^{+\infty}$ is said to be an *absorbing Markov chain* if $\forall t \geq 0 : P(\xi_{t+1} \notin \mathcal{X}^* \mid \xi_t \in \mathcal{X}^*) = 0$, and $\{\xi_t\}_{t=0}^{+\infty}$ is said to be a *homogeneous Markov chain* if $\forall t \geq 0 \forall x, y \in \mathcal{X} : P(\xi_{t+1} = x \mid \xi_t = y) = P(\xi_1 = x \mid \xi_0 = y)$. All practical EAs track the best-so-far solutions during the evolution process. These EAs can be modeled as absorbing Markov chains. EAs without time-variant operators can be modeled as homogeneous Markov chains.

Starting from time step $\tilde{t}$ when $\xi_{\tilde{t}} = x$, let $\tau_{\tilde{t}}$ be a random variable denoting the hitting events:

$$\begin{aligned}
&\tau_{\tilde{t}} = 0 : \ \xi_{\tilde{t}} \in \mathcal{X}^*,\\
&\tau_{\tilde{t}} = 1 : \ \xi_{\tilde{t}+1} \in \mathcal{X}^* \wedge \xi_i \notin \mathcal{X}^* \ (i = \tilde{t})\,,\\
&\tau_{\tilde{t}} = 2 : \ \xi_{\tilde{t}+2} \in \mathcal{X}^* \wedge \xi_i \notin \mathcal{X}^* \ (i = \tilde{t}, \tilde{t}+1)\,,\\
&\dots .
\end{aligned}$$

The mathematical expectation of $\tau_{\tilde{t}}$, $\mathbb{E}[\![\tau_{\tilde{t}} \mid \xi_{\tilde{t}} = x]\!] = \sum_{i=0}^{+\infty} i \cdot P(\tau_{\tilde{t}} = i)$, is called the *conditional first hitting time* (CFHT) of the Markov chain from $\tilde{t}$ and $\xi_{\tilde{t}} = x$. If $\xi_{\tilde{t}}$ is drawn from a distribution π of states, the expectation of the CFHT over $\xi_{\tilde{t}}$,

$$\mathbb{E}[\![\tau_{\tilde{t}} \mid \xi_{\tilde{t}} \sim \pi]\!] = \mathbb{E}_{x\sim\pi}[\![\tau_{\tilde{t}} \mid \xi_{\tilde{t}} = x]\!] = \sum\nolimits_{x\in\mathcal{X}} \pi(x)\mathbb{E}[\![\tau_{\tilde{t}} \mid \xi_{\tilde{t}} = x]\!],$$

is called the *distribution-conditional first hitting time* (DCFHT) of the Markov chain from $\tilde{t}$ and $\xi_{\tilde{t}} \sim \pi$. The DCFHT of the chain from $t = 0$ and uniform distribution π_u,

$$\mathbb{E}[\![\tau]\!] = \mathbb{E}[\![\tau_0 \mid \xi_0 \sim \pi_u]\!] = \mathbb{E}_{x\sim\pi_u}[\![\tau_0 \mid \xi_0 = x]\!] = \sum\nolimits_{x\in\mathcal{X}} \frac{1}{|\mathcal{X}|}\mathbb{E}[\![\tau_0 \mid \xi_0 = x]\!],$$

is called the *expected first hitting time* (EFHT) of the Markov chain [9,10,20], which implies the average runtime of EAs.

About the CFHT, we will use the following two lemmas [8].

Lemma 1. *Given an absorbing Markov chain $\{\xi_t\}_{t=0}^{+\infty}(\xi_t \in \mathcal{X})$ and a target subspace $\mathcal{X}^* \subset \mathcal{X}$, we have, for CFHT,*

$$\forall x \notin \mathcal{X}^* : \mathbb{E}[\![\tau_t \mid \xi_t = x]\!] = 1 + \sum_{y\in\mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau_{t+1} \mid \xi_{t+1} = y]\!],$$

and for DCFHT,

$$\mathbb{E}[\![\tau_t \mid \xi_t \sim \pi_t]\!] = \mathbb{E}_{x\sim\pi_t}[\![\tau_t \mid \xi_t = x]\!] = 1 - \pi_t(\mathcal{X}^*) + \mathbb{E}[\![\tau_{t+1} \mid \xi_{t+1} \sim \pi_{t+1}]\!],$$

where $\pi_{t+1}(x) = \sum_{y\in\mathcal{X}} \pi_t(y)P(\xi_{t+1} = x \mid \xi_t = y)$.

Lemma 2. *Given an absorbing homogeneous Markov chain $\{\xi_t\}_{t=0}^{+\infty}(\xi_t \in \mathcal{X})$ and a target subspace $\mathcal{X}^* \subset \mathcal{X}$, it holds $\forall t_1, t_2 : \mathbb{E}[\![\tau_{t_1} \mid \xi_{t_1} = x]\!] = \mathbb{E}[\![\tau_{t_2} \mid \xi_{t_2} = x]\!]$.*

3 Markov Chain Switching Theorem

We focus on tackling the interaction between mutation and recombination operators. Considering that there are many studies devoted to the estimate of EFHT of EAs using mutation only [2,7,9,20], we propose to analyze recombination operators by studying an EA that turns the recombination operator on and off. For this purpose, we present the following Markov Chain Switching Theorem.

Theorem 1 (Markov Chain Switching Theorem). *Given two absorbing homogeneous Markov chains $\{\xi_t\}_{t=0}^{+\infty}$ and $\{\xi'_t\}_{t=0}^{+\infty}$. Let τ and τ' denote the hitting events of ξ_t and ξ'_t, respectively. Let π_t denote the distribution of ξ_t on states. If it satisfies*

$$\forall t: \sum\nolimits_{x,y\in\mathcal{X}} \pi_t(x)P(\xi_{t+1}=y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1}=y]\!] \tag{1}$$
$$\leq (\geq) \sum\nolimits_{x,y\in\mathcal{X}} \pi_t(x)P(\xi'_{t+1}=y \mid \xi'_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1}=y]\!] \,,$$

and both $\mathbb{E}[\![\tau]\!]$ and $\mathbb{E}[\![\tau']\!]$ are finite, it holds that $\mathbb{E}[\![\tau]\!] \leq (\geq)\mathbb{E}[\![\tau']\!]$.

Before the proof, it is helpful to briefly explain the significance of the theorem. Let ξ_t corresponds to an EA that turns its recombination operator on, and ξ'_t be the opposite. In Eq. 1, only the CFHT of the EA turning recombination off, i.e., $\mathbb{E}[\![\tau'_{t+1}|\xi'_{t+1}]\!]$, is required to be calculated, whilst there is no such term as $\mathbb{E}[\![\tau_{t+1}|\xi_{t+1}]\!]$. Therefore, Theorem 1 enables us to analyze the effect of recombination operator by comparing the one-step transition probabilities, i.e., $P(\xi_{t+1}=y \mid \xi_t = x)$ and $P(\xi'_{t+1}=y \mid \xi'_t = x)$, avoiding complicated calculations on the CFHT of the EA turning recombination on. Note that, besides recombination, Theorem 1 can also be applied to analyze many other kinds of operators.

Proof. Here, we prove the "$\leq$" case, while the "$\geq$" case can be proved similarly. Denote the operator used in the EA modeled by ξ as *op*, and that used in the EA modeled by ξ' as *op*$'$, respectively. Let Markov chain $\{\xi^k_t\}_{t=0}^{+\infty}$ corresponds to the EA using *op* at time steps $\{0,1,\cdots,k-1\}$ and using *op*$'$ otherwise. Thus, for any k and any time step $t \geq k$, we have

$$\forall x \in \mathcal{X}: \mathbb{E}[\![\tau^k_t \mid \xi^k_t = x]\!] = \mathbb{E}[\![\tau'_t \mid \xi'_t = x]\!], \tag{2}$$

since the two chains use the same operator after the time step $k-1$. To prove the inequality $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$, we prove its DCFHT version $\mathbb{E}[\![\tau_0 \mid \xi_0 \sim \pi_0]\!] \leq \mathbb{E}[\![\tau'_0 \mid \xi'_0 \sim \pi'_0]\!]$ by induction on the number of time steps, where *op* is applied instead of *op*$'$. Note that $\pi_0 = \pi'_0$ for the random generation of solutions.

(a) Initialization. We prove by Eq. 1 that $\mathbb{E}[\![\tau^1_0 \mid \xi^1_0 \sim \pi_0]\!] \leq \mathbb{E}[\![\tau'_0 \mid \xi'_0 \sim \pi'_0]\!]$ as follows. Since *op* is applied only at $t=0$, we have

$$\begin{aligned}
&\mathbb{E}[\![\tau^1_0 \mid \xi^1_0 \sim \pi_0]\!] = \sum\nolimits_{x\in\mathcal{X}} \pi_0(x)\mathbb{E}[\![\tau^1_0 \mid \xi^1_0 = x]\!]\\
&= 1-\pi_0(\mathcal{X}^*) + \sum\nolimits_{x,y\in\mathcal{X}} \pi_0(x)P(\xi^1_1 = y \mid \xi^1_0 = x)\mathbb{E}[\![\tau^1_1 \mid \xi^1_1 = y]\!]\\
&= 1-\pi_0(\mathcal{X}^*) + \sum\nolimits_{x,y\in\mathcal{X}} \pi_0(x)P(\xi^1_1 = y \mid \xi^1_0 = x)\mathbb{E}[\![\tau'_1 \mid \xi'_1 = y]\!]\\
&\leq 1-\pi'_0(\mathcal{X}^*) + \sum\nolimits_{x,y\in\mathcal{X}} \pi'_0(x)P(\xi'_1 = y \mid \xi'_0 = x)\mathbb{E}[\![\tau'_1 \mid \xi'_1 = y]\!]\\
&= \mathbb{E}[\![\tau'_0 \mid \xi'_0 \sim \pi'_0]\!] \,,
\end{aligned}$$

where the first, second and last equations are obtained by Lemma 1, the third equation is obtained Eq. 2, and the followed inequality is obtained by Eq. 1 and by $\pi_0 = \pi'_0$.

(b) Inductive Hypothesis. Assume that at the induction step $K > 0$, $\forall k \leq K-1$: $\mathbb{E}[\![\tau_0^k \mid \xi_0^k \sim \pi_0]\!] \leq \mathbb{E}[\![\tau_0' \mid \xi_0' \sim \pi_0']\!]$. Then, at the time step $t = K$, we have

$$\begin{aligned}
&\mathbb{E}[\![\tau_{K-1}^K \mid \xi_{K-1}^K \sim \pi_{K-1}]\!] \\
&= 1 - \pi_{K-1}(\mathcal{X}^*) + \sum_{x,y \in \mathcal{X}} \pi_{K-1}(x) P(\xi_K^K = y | \xi_{K-1}^K = x) \mathbb{E}[\![\tau_K^K | \xi_K^K = y]\!] \\
&= 1 - \pi_{K-1}(\mathcal{X}^*) + \sum_{x,y \in \mathcal{X}} \pi_{K-1}(x) P(\xi_K^K = y | \xi_{K-1}^K = x) \mathbb{E}[\![\tau_K' | \xi_K' = y]\!] \\
&\leq 1 - \pi_{K-1}(\mathcal{X}^*) + \sum_{x,y \in \mathcal{X}} \pi_{K-1}(x) P(\xi_K' = y | \xi_{K-1}' = x) \mathbb{E}[\![\tau_K' | \xi_K' = y]\!] \\
&= \sum\nolimits_{x \in \mathcal{X}} \pi_{K-1}(x) \mathbb{E}[\![\tau_{K-1}' \mid \xi_{K-1}' = x]\!] \\
&= \sum\nolimits_{x \in \mathcal{X}} \pi_{K-1}(x) \mathbb{E}[\![\tau_{K-1}^{K-1} \mid \xi_{K-1}^{K-1} = x]\!] \\
&= \mathbb{E}[\![\tau_{K-1}^{K-1} \mid \xi_{K-1}^{K-1} \sim \pi_{K-1}]\!] ,
\end{aligned}$$

where the first and the third equations are obtained by Lemma [1], the second and the fourth equations are obtained by Eq. [2], and the inequality is obtained by Eq. [1]. Thus, by the induction hypothesis, we get

$$\begin{aligned}
&\mathbb{E}[\![\tau_0^K \mid \xi_0^K \sim \pi_0]\!] = K - 1 - \sum\nolimits_{t=0}^{K-2} \pi_t(\mathcal{X}^*) + \mathbb{E}[\![\tau_{K-1}^K \mid \xi_{K-1}^K \sim \pi_{K-1}]\!] \\
&\leq K - 1 - \sum\nolimits_{t=0}^{K-2} \pi_t(\mathcal{X}^*) + \mathbb{E}[\![\tau_{K-1}^{K-1} \mid \xi_{K-1}^{K-1} \sim \pi_{K-1}]\!] \\
&= \mathbb{E}[\![\tau_0^{K-1} \mid \xi_0^{K-1} \sim \pi_0]\!] \leq \mathbb{E}[\![\tau_0' \mid \xi_0' \sim \pi_0']\!] .
\end{aligned}$$

(c) Conclusion. From (a) and (b), it holds that $\mathbb{E}[\![\tau_0^\infty \mid \xi_0^\infty \sim \pi_0]\!] \leq \mathbb{E}[\![\tau_0' \mid \xi_0' \sim \pi_0']\!]$, i.e., $\mathbb{E}[\![\tau^\infty]\!] \leq \mathbb{E}[\![\tau']\!]$. Finally, since $\mathbb{E}[\![\tau]\!]$ is finite, we have $\mathbb{E}[\![\tau]\!] = \mathbb{E}[\![\tau^\infty]\!]$, and thus $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$. □

4 Analysis of LeadingOnes Problem

4.1 The Evolutionary Algorithm

We study the (2:2)-EA as in Definition [1], which is generalized with the minimum difference from the (1+1)-EA [6] for enabling recombination. The (2:2)-EA uses the population size two and a *direct offspring selection* (i.e., selecting between a parent and its direct offspring, so denoted as '2:2' instead of '2+2'). This helps to isolate the influences of population size and selection pressure that are beyond the scope of this paper. Note that as used in [19,15], though the population size of two is small, it is sufficient for showing the effect of recombination operators.

Definition 1 ((2:2)-EA). *Encode each solution by a string with n binary bits, and let every population, denoted by variable ξ, contain 2 solutions. The (2:2)-EA consists of the following steps:*

```
1. (Initialization) Let $t \leftarrow 0$.
   $\xi_0 :=$ randomly generated population.
2. Let $(s_1, s_2) := \xi_t$.
3. (Reproduction) If $UseRecombination$ is true,
       $(s_1^R, s_2^R) :=$ M&R$(s_1, s_2)$
4. Else, $(s_1^R, s_2^R) :=$ Mutation$(s_1, s_2)$
5. (Selection) $\xi_{t+1} := \big( \arg\max_{s \in \{s_1, s_1^R\}} f(s), \ \arg\max_{s' \in \{s_2, s_2^R\}} f(s') \big)$.
6. (Stop Criterion) Terminates if the optima is reached.
7. (Loop) Let $t \leftarrow t+1$, goto step 2.
```

Mutation : $\mathcal{X} \rightarrow \mathcal{X}$ *is a mutation operator, M&R* : $\mathcal{X} \times \mathcal{X} \rightarrow \mathcal{X} \times \mathcal{X}$ *is the strategy of combining mutation and recombination operators, and* $UseRecombination$ *is a switching parameter. Note that, when a parent has the same fitness as its offspring in selection, the parent is selected.*

We describe several operators below. Note that, when using recombination operators, two parents generate two offsprings by exchanging some bits. To apply the *direct offspring selection*, we need identify which of the two generated solutions is the *direct offspring* of a parent. This is realized by considering which of the two generated solutions is closer to the parent, i.e., with smaller Hamming distance (ties are broken randomly).

– **One-bit mutation.** For each solution, randomly choose one of the n bits and flip (0 to 1 or inverse) the chosen bit.
– **First-bit recombination.** For the two current solutions, scan the solutions from left to right, and exchange the first different bits.
– **One-bit recombination.** For the two current solutions, randomly choose one of the n positions and exchange the bits on that position.

4.2 Analysis

The LeadingOnes problem is given in Definition 2. Denote $s(j)$ be the j-th bit of solution s counting from left to right (thus $s(1)$ is the left-most bit). We define $LO(s) = \{\max i; s.t. \forall j \leq i, s(j) = 1\}$ and $\delta(s_1, s_2) = \|s_1\| - \|s_2\|$, where $\|\cdot\|$ denotes the 1-norm, i.e., the number of one bits.

Definition 2 (LeadingOnes Problem). *LeadingOnes Problem of size* n *is to find an* n *bits binary string* s^* *such that* $s^* = \arg\max_{s \in \{0,1\}^n} LO(s)$.

It is easy to find out that the optimal solution of the LeadingOnes problem is $s^* = (1, 1, \cdots, 1)$, which has n leading one bits, i.e., $LO(s^*) = n$; also note that $LO((0, 1, \cdots, 1)) = 0$ since the leading bit is zero. This problem is one of the most widely studied problems for EAs using mutation operators [16], however, previous analysis on this problem did not touch recombination operators.

In the following, we analyze (2:2)-EA on LeadingOnes problem with Mutation implemented by one-bit mutation and M&R implemented by four strategies described below. The function LO is used as the fitness function in (2:2)-EA. For the two solutions (s_1, s_2) in the current population such that $\delta(s_1, s_2) \geq 0$, we simply denote $LO_1 = LO(s_1)$, $LO_2 = LO(s_2)$, and $\delta = \delta(s_1, s_2)$.

- **M&R1a.** Use the first-bit recombination if either $LO_1 < LO_2$ holds or both $\delta = 0$ and $LO_1 \neq LO_2$ hold; otherwise use the one-bit mutation.
- **M&R1b.** Use the first-bit recombination if both $LO_1 > LO_2$ and $0 < \delta \leq 2$ hold; otherwise use the one-bit mutation.
- **M&R1.** Use the first-bit recombination if either M&R1a or M&R1b holds; otherwise use the one-bit mutation.
- **M&R2.** Use the one-bit recombination if both $LO_1 < LO_2$ and $s_1(LO_2 + 1) = 0$ hold, or both $LO_1 > LO_2$ and $s_2(LO_1 + 1) = 0$ hold; otherwise use the one-bit mutation.

Our analysis below shows that M&R1a and M&R1b are both superior to one-bit mutation (Propositions 2 and 3), and the combination of the two strategies, M&R1, is also superior to one-bit mutation (Proposition 4). However, M&R2 is inferior to one-bit mutation (Proposition 5).

In the following, unless stated, we let $\{\xi_t\}_{t=0}^{+\infty}$ correspond to the (2:2)-EA using M&R, and let $\{\xi'_t\}_{t=0}^{+\infty}$ correspond to the (2:2)-EA using Mutation. Thus, $\mathbb{E}[\![\tau]\!]$ and $\mathbb{E}[\![\tau']\!]$ denote the EFHT of (2:2)-EA using M&R and Mutation, respectively. We denote $\mathbb{E}(i,j)$ as the CFHT $\mathbb{E}[\![\tau'_t|\xi'_t = \{s_1, s_2\}]\!]$ of EA using one-bit mutation, given that $\|s_1\| = n - i$ and $\|s_2\| = n - j$. We present proof sketches below due to space limit, and full proof will be provided in a longer version.

Proposition 1. *The CFHT of* $\{\xi'_t\}_{t=0}^{+\infty}$ *satisfies that*
$\forall i \geq 1, \delta \geq 0 : \frac{n}{2} < \mathbb{E}(i, i+\delta) - \mathbb{E}(i-1, i+\delta) \leq n - \frac{3n-1}{2^{\delta+3}}$,
and $\forall i \geq 1, \delta \geq 1 : \frac{n}{2^{\delta+2}} \leq \mathbb{E}(i, i+\delta) - \mathbb{E}(i, i+\delta-1) < \frac{n}{2}$.

Proof. It is easy to prove the proposition by induction on $i + i + \delta$. □

Proposition 2. *Given M&R in (2:2)-EA being implemented by M&R1a, when* $n \geq 2$, *we have* $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$.

Proof. For any population $x = \{s_1, s_2\}$ such that $\|s_1\| = n - i$ and $\|s_2\| = n - i - \delta(\delta \geq 0)$:
a) in the case $\delta = 0 \wedge LO_1 \neq LO_2$ or $LO_1 < LO_2$, the first-bit recombination and the selection reproduce two solutions $\{s'_1, s'_2\}$ such that $\|s'_1\| = n - i + 1$ and $\|s'_2\| = n - i - \delta$. We have, using Proposition 1,

$$\begin{aligned}&\sum\nolimits_{y\in\mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]\\ &= \mathbb{E}(i-1, i+\delta) < \mathbb{E}(i, i+\delta) - \frac{n}{2} \leq \mathbb{E}(i, i+\delta) - 1.\end{aligned}$$

b) Otherwise, it uses the one-bit mutation, which yields

$$\begin{aligned}&\sum\nolimits_{y\in\mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]\\ &= \sum\nolimits_{y\in\mathcal{X}} P(\xi'_{t+1} = y \mid \xi'_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!] = \mathbb{E}(i, i+\delta) - 1.\end{aligned}$$

Thus, for any population x, it holds that

$$\sum\nolimits_{y\in\mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]$$

$$\leq \mathbb{E}(i, i+\delta) - 1 = \sum\nolimits_{y \in \mathcal{X}} P(\xi'_{t+1} = y \mid \xi'_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!].$$

By Theorem 1, we immediately have $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$. □

Remarks. We can observe from the proof of Proposition 2 that how the analysis has be simplified by Theorem 1. By Theorem 1, we need to bound

$$\sum\nolimits_{y \in \mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!] \ ,$$

where $P(\xi_{t+1} = y \mid \xi_t = x)$ is the one-step behavior of the EA using M&R, and $\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]$ is the CFHT of the EA using Mutation from y. In the proposition, the expression is calculated to be $\mathbb{E}(i-1, i+\delta)$. By the analysis of the EA using Mutation in Proposition 1, we then bound that $\mathbb{E}(i-1, i+\delta) \leq \mathbb{E}(i, i+\delta) - 1$, while

$$\mathbb{E}(i, i+\delta) - 1 = \sum\nolimits_{y \in \mathcal{X}} P(\xi'_{t+1} = y \mid \xi'_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!] \ .$$

Thus the condition of Theorem 1 is satisfied.

Proposition 3. *Given M&R in (2:2)-EA being implemented by M&R1b, when $n \geq 16$, we have $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$.*

Proof. For any population $x = \{s_1, s_2\}$ such that $\|s_1\| = n - i$ and $\|s_2\| = n - i - \delta(\delta \geq 0)$. Note that in the case $0 < \delta \leq 2 \wedge LO_1 > LO_2$ it uses the first-bit recombination. We have

$$\begin{aligned}&\sum\nolimits_{y \in \mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]\\ &= \mathbb{E}(i, i+\delta-1) \leq \mathbb{E}(i, i+\delta) - \frac{n}{2^{\delta+2}} \leq \mathbb{E}(i, i+\delta) - 1 \ .\end{aligned}$$

The remaining of the proof is the same as in Proposition 2. □

Since we only need to deal with the one-step transition probability, we get Proposition 4 directly.

Proposition 4. *Given M&R in (2:2)-EA being implemented by M&R1, when $n \geq 16$, we have $\mathbb{E}[\![\tau]\!] \leq \mathbb{E}[\![\tau']\!]$.*

Proposition 5. *Given M&R in (2:2)-EA being implemented by M&R2, when $n \geq 2$, we have $\mathbb{E}[\![\tau]\!] \geq \mathbb{E}[\![\tau']\!]$.*

Proof. For any population $x = \{s_1, s_2\}$ such that $\|s_1\| = n - i$ and $\|s_2\| = n - i - \delta(\delta \geq 0)$. Note that in the case $LO_1 < LO_2 \wedge s_1(LO_2 + 1) = 0$, it uses the one-bit recombination. So we have

$$\begin{aligned}&\sum\nolimits_{y \in \mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]\\ &= \big(\mathbb{E}(i-1, i+\delta) + (n-1)\mathbb{E}(i, i+\delta)\big)/n\\ &\geq \mathbb{E}(i, i+\delta) - \frac{1}{n}(n - \frac{3n-1}{2^{\delta+3}})\\ &= \mathbb{E}(i, i+\delta) - 1 + \frac{3 - 1/n}{2^{\delta+3}} \geq \mathbb{E}(i, i+\delta) - 1 \ .\end{aligned}$$

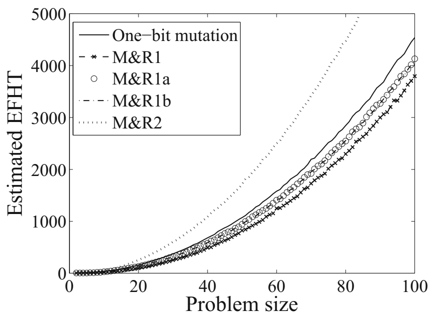


Fig. 1. Estimated EFHT with problem size $[2, 100]$

While in the case $LO_1 > LO_2 \wedge s_2(LO_1 + 1) = 0$, it also uses the one-bit recombination. We have

$$\begin{aligned}&\sum\nolimits_{y\in\mathcal{X}} P(\xi_{t+1} = y \mid \xi_t = x)\mathbb{E}[\![\tau'_{t+1} \mid \xi'_{t+1} = y]\!]\\&= \big(\mathbb{E}(i, i+\delta-1) + (n-1)\mathbb{E}(i, i+\delta)\big)/n\\&\geq \mathbb{E}(i, i+\delta) - \frac{5+1/n}{8} > \mathbb{E}(i, i+\delta) - 1\ .\end{aligned}$$

The remaining of the proof is similar to that in Proposition 2. □

To verify the theoretical results, we run the (2:2)-EA on the LeadingOnes problem with problem size ranging up to 100. On each size, we repeat independent runs of each implementation of the EA for 1,000 times, and then average the runtimes as an estimate of the EFHT. The results are plotted in Figure 1. It can be observed that the runtime of both M&R1a and M&R1b is smaller than that using one-bit mutation, and the two strategies have similar runtime such that their curves overlap largely. It is also observable that M&R1, which combines M&R1a and M&R1b, is more efficient than one-bit mutation, as well as M&R1a and M&R1b. Meanwhile, M&R2 is worse than one-bit mutation. These observations verify our analysis results in Propositions 2-5.

5 Conclusion

It is difficult to analyze recombination operators theoretically in terms of runtime, since they operate on population and interact with mutation operators. In this paper, we present a general approach which allows to compare the runtime of an EA turning the recombination on and off. The key is the *Markov Chain Switching Theorem* which compares two Markov chains for the first hit of the target.

For the simplicity of analysis, in this paper we only present a case study of a simple EA on the LeadingOnes problem to show how the proposed approach can be helpful. We will present the analysis on the OneMax problem in a longer version. For more realistic EAs that use uniform mutation and uniform recombination, since one-step of operation could generate many different solutions, compact analysis using our approach is an interesting future work.

Acknowledgements

We want to thank the anonymous reviewers for their helpful comments and suggestions. This research was supported by the Jiangsu Science Foundation (BK2008018) and the Jiangsu 333 High-Level Talent Cultivation Program.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies. In: Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford (1996)
2. Beyer, H.G., Schwefel, H.P., Wegener, I.: How to analyse evolutionary algorithms. Theoretical Computer Science 287(1), 101–130 (2002)
3. Chen, T., Tang, K., Chen, G., Yao, X.: On the analysis of average time complexity of estimation of distribution algorithms. In: Proceedings of CEC 2007, Singapore, pp. 25–28 (2007)
4. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. In: Proceedings of GECCO 2008, Atlanta, GA, pp. 539–546 (2008)
5. Doerr, B., Theile, M.: Improved analysis methods for crossover-based algorithms. In: Proceedings of GECCO 2009, Montreal, Canada, pp. 247–254 (2009)
6. Droste, S., Jansen, T., Wegener, I.: A rigorous complexity analysis of the (1 + 1) evolutionary algorithm for linear functions with boolean inputs. Evolutionary Computation 6(2), 185–196 (1998)
7. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276(1-2), 51–81 (2002)
8. Freĭdlin, M.I.: Markov Processes and Differential Equations: Asymptotic Problems. Birkhäuser Verlag, Basel (1996)
9. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artifical Intelligence 127(1), 57–85 (2001)
10. He, J., Yao, X.: Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artificial Intelligence 145(1-2), 59–97 (2003)
11. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms - a proof that crossover really can help. Algorithmica 34(1), 47–66 (2002)
12. Jansen, T., Wegener, I.: Real royal road functions - where crossover provably is essential. Discrete Applied Mathematics 149, 111–125 (2005)
13. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input output sequences. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 595–604. Springer, Heidelberg (2008)
14. Lin, G., Yao, X.: Analysing crossover operators by search step size. In: Proceedings of CEC 1997, Indianapolis, IN, pp. 107–110 (1997)
15. Richter, J.N., Wright, A., Paxton, J.: Ignoble trails - where crossover is provably harmful. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 92–101. Springer, Heidelberg (2008)
16. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Verlag Dr. Kovač, Hamburg (1997)
17. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education, Englewood Cliffs (2003)
18. Spears, W.: Evolutionary Algorithms: The Role of Mutation and Recombination. Springer, Berlin (2000)
19. Storch, T., Wegener, I.: Real royal road functions for constant population size. Theoretical Computer Science 320(1), 123–134 (2004)
20. Yu, Y., Zhou, Z.-H.: A new approach to estimating the expected first hitting time of evolutionary algorithms. Artificial Intelligence 172(15), 1809–1832 (2008)

Bidirectional Relation between CMA Evolution Strategies and Natural Evolution Strategies

Youhei Akimoto[1,2], Yuichi Nagata[1], Isao Ono[1], and Shigenobu Kobayashi[1]

[1] Tokyo Institute of Technology,
G5-21, 4259 Nagatsuta-cho, Midori-ku, Yokohama, 226-8502, Japan
{akimoto@fe.,nagata@fe.,isao@,kobayasi@}dis.titech.ac.jp
[2] Research Fellow of the Japan Society for the Promotion of Science

Abstract. This paper investigates the relation between the covariance matrix adaptation evolution strategy and the natural evolution strategy, the latter of which is recently proposed and is formulated as a natural gradient based method on the expected fitness under the mutation distribution. To enable to compare these algorithms, we derive the explicit form of the natural gradient of the expected fitness and transform it into the forms corresponding to the mean vector and the covariance matrix of the mutation distribution. We show that the natural evolution strategy can be viewed as a variant of covariance matrix adaptation evolution strategies using Cholesky update and also that the covariance matrix adaptation evolution strategy can be formulated as a variant of natural evolution strategies.

1 Introduction

Recently in the field of continuous function optimization, natural evolution strategies (NESs) have been proposed by Wierstra et al. [1] and developed by Sun et al. [2]. The NES utilizes the Gaussian mutation to generate new search points and adjusts the parameter of the mutation at each generation in order to improve the expected fitness under the mutation distribution. For the adjustment, the NES makes use of the natural gradient [3] of the expected fitness with respect to the parameter of the mutation distribution, which is referred to as a *natural evolution gradient*. Sun et al. [2] reported that the performance of the NES is competitive to that of the covariance matrix adaptation evolution strategy (CMA-ES, e.g. [4,5]) on standard benchmarks.

Now an interesting question arises as to why they perform similarly despite their apparently different update rules for the parameters of the mutation distribution. Investigating the relation between the NES and the CMA-ES is beneficial to understand the algorithms. Comparing the NES with the CMA-ES, which is more intuitively understandable in terms of the update rules of the parameters of the mutation distribution, helps to understand how the NES adjusts the parameter of the mutation distribution. Describing the CMA-ES in the framework of the NES, which seems theoretically more tractable, allows deriving the convergence theory.

We investigate the relation between the NES and the CMA-ES. The rest of the paper is organized as follows. In section 2, we explain the concepts of the *evolution gradient* – the gradient of the expected fitness – and of the natural evolution gradient. In section 3, we elucidate the connection between the NES and the CMA-ES by deriving the explicit form of the natural evolution gradient, which is computed in [2] with an iterative computation of the inversion of the Fisher information matrix (FIM) of the mutation distribution. In section 4, we show that the CMA-ES employing global weighted recombination and rank-μ update without step-size adaptation can be formulated as a variant of natural evolution gradient based methods. Finally in section 5, we discuss the results and conclude this paper.

2 Formulation

The objective of minimization is to find the point $\mathbf{x}$ at which an objective function $f : \mathbb{R}^d \to \mathbb{R}$ has the minimal value. Both the NES and the CMA-ES search the optimal point via Gaussian mutation. Their algorithms repeat two steps at each generation: mutation step and update step. At the mutation step, their algorithms generate new points from a Gaussian distribution with mean $\mathbf{m}$ and covariance matrix $\mathbf{C}$. At the update step, their parameters $\theta = \langle \mathbf{m}, \mathbf{C} \rangle$ are updated to promote promising mutation by using the sample points. The update of θ in the NES, as we mention in the following subsections, is based on the gradient of the expected fitness under the mutation distribution, while that in the CMA-ES is related to the maximum likelihood estimation of the Gaussian distribution.

2.1 Evolution Gradient

The NES adjusts θ to optimize the expected fitness $J(\theta) = \mathbb{E}[f(\mathbf{x}) \mid \theta]$ of the next generation under the mutation distribution $\pi(\mathbf{x} \mid \theta)$ by using the natural evolution gradient – the natural gradient of $J(\theta)$. Preparatory to introducing the notion of the natural evolution gradient, we introduce the concept of the evolution gradient.

One of the most straightforward approaches to adjusting θ relies on the gradient $\nabla_\theta J(\theta)$ of $J(\theta)$. Let

$$\pi(\mathbf{x} \mid \theta) = \frac{1}{(2\pi)^{d/2} \det(\mathbf{C})^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x} - \mathbf{m})^{\mathrm{T}} \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m})\Big) \tag{1}$$

denote the probability density of the Gaussian distribution given parameter $\theta = \langle \mathbf{m}, \mathbf{C} \rangle$. The expected fitness under the mutation distribution $\pi(\mathbf{x} \mid \theta)$ is

$$J(\theta) = \int f(\mathbf{x}) \pi(\mathbf{x} \mid \theta) d\mathbf{x} \ . \tag{2}$$

Using the log-likelihood trick, we can express the gradient $\nabla_\theta J(\theta)$ with respect to θ as

$$\nabla_\theta J(\theta) = \nabla_\theta \int f(\mathbf{x}) \pi(\mathbf{x} \mid \theta) d\mathbf{x} = \int \pi(\mathbf{x} \mid \theta) f(\mathbf{x}) \nabla_\theta \ln \pi(\mathbf{x} \mid \theta) d\mathbf{x} \ . \tag{3}$$

This is referred to as an evolution gradient. If the gradient $\nabla_\theta J(\theta^t)$ at the current location θ^t is given, one can update θ^{t+1} by shifting θ^t in the direction of the negative gradient, $-\nabla_\theta J(\theta^t)$, as $\theta^{t+1} = \theta^t - \eta \cdot \nabla_\theta J(\theta^t)$.

However, since the objective function is unknown, so is the evolution gradient. We alternatively utilize the Monte-Carlo approximation:

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{\lambda} \frac{f(\mathbf{x}_i)}{\lambda} \nabla_\theta \ln \pi(\mathbf{x}_i \mid \theta) \ , \tag{4}$$

where $\mathbf{x}_i$ are samples generated from $\pi(\mathbf{x} \mid \theta)$. To eliminate premature convergence attributed to disturbance of the estimation of evolution gradients and explicit the invariant property against order-preserving transformation, a ranking based fitness shaping is introduced in [1,2]:

$$(f(\mathbf{x}_1)/\lambda, \dots, f(\mathbf{x}_\lambda)/\lambda) \to (-\mathrm{w}_{R_1}, \dots, -\mathrm{w}_{R_\lambda}), \quad \mathrm{w}_1 \geq \cdots \geq \mathrm{w}_\lambda \ . \tag{5}$$

Here the index R_i denotes the rank of $\mathbf{x}_i$ among $\mathbf{x}_1, \dots, \mathbf{x}_\lambda$ with respect to f-values. That is, $f(\mathbf{x}_i)$ is the R_ith smallest among $f(\mathbf{x}_1), \dots, f(\mathbf{x}_\lambda)$.

2.2 Natural Evolution Gradient

The natural gradient [3] has a background in information geometry, which studies the Riemannian geometric structure of the manifold of probability distributions. A result in information geometry states that the Fisher information matrix (FIM) defines a Riemannian metric tensor on the space of probability distributions [6] and that the direction of the steepest descent on a Riemannian manifold is given by the natural gradient, which is given by the conventional gradient premultiplied by the inverse matrix of the Riemannian metric tensor [3]. Thus, the natural gradient can be computed from the gradient and the FIM, and the natural gradient descent tends to converge faster than conventional one. Furthermore, the natural gradient descent is a variable metric method, which provides uniform convergence properties. In the field of machine learning, natural gradient learning has been used as an efficient method that can prevent from being stuck on plateaus [7].

The NES utilizes the natural evolution gradient – the natural gradient of the expected fitness – in lieu of the conventional one. If the FIM is invertible, the natural evolution gradient $\tilde{\nabla}_\theta J(\theta) = \mathbf{F}^{-1}(\theta)\nabla_\theta J(\theta)$ is given by the evolution gradient premultiplied by the inverse matrix of the FIM $\mathbf{F}(\theta)$. It is well-known that the FIM for a Gaussian distribution takes an explicit form. The $(i,\ j)$ element of the FIM $\mathbf{F}(\theta)$ of the Gaussian distribution $\mathcal{N}(\mathbf{m}(\theta), \mathbf{C}(\theta))$ is

$$\mathbf{F}_{i,j} = \frac{\partial \mathbf{m}^{\mathrm{T}}}{\partial \theta_i} \mathbf{C}^{-1} \frac{\partial \mathbf{m}}{\partial \theta_j} + \frac{1}{2} \mathrm{tr}\Big(\mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial \theta_i} \mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial \theta_j}\Big). \tag{6}$$

We approximate the natural evolution gradient by replacing the exact gradient with its estimation, namely,

$$\tilde{\nabla}_\theta J(\theta) \approx \delta\theta = -\sum_{i=1}^{\lambda} \mathrm{w}_{R_i} \mathbf{F}^{-1}(\theta) \nabla_\theta \ln \pi(\mathbf{x}_i \mid \theta) \ . \tag{7}$$

Consequently, the NES framework repeats the estimation of the natural evolution gradient $\delta\theta^t$ by using the samples $\mathbf{x}_i^t \sim \pi(\cdot \mid \theta^t)$ generated at tth generation and the adjustment of the parameter by $\theta^{t+1} = \theta^t - \eta\delta\theta^t$. It can be considered that the NES transforms the minimization of $f(\mathbf{x})$ into the minimization of the expected fitness $J(\theta)$ under the mutation distribution $\pi(x \mid \theta)$ and minimizes $J(\theta)$ by using the estimation of the natural gradient of $J(\theta)$.

3 NES as a Variant of CMA-ESs

Let $\mathbf{A}$ represent the Cholesky decomposition of the covariance matrix $\mathbf{C}$, or more rigorously, let $\mathbf{A}$ be the unique lower triangular matrix such that $\mathbf{C} = \mathbf{A}\mathbf{A}^{\mathrm{T}}$. Let θ be a $[d(d+3)/2]$-dimensional column vector consisting of the elements of $\mathbf{m}$ and the lower left elements of $\mathbf{A}$, more precisely,

$$\theta = \left[\mathbf{m}^{\mathrm{T}} \ \mathrm{vech}(\mathbf{A})^{\mathrm{T}}\right]^{\mathrm{T}} . \tag{8}$$

Here $\mathrm{vech}(\mathbf{A}) = \left[\, (\mathbf{A}_{1:d,1})^{\mathrm{T}} \ (\mathbf{A}_{2:d,2})^{\mathrm{T}} \ \dots \ (\mathbf{A}_{d:d,d})^{\mathrm{T}} \,\right]^{\mathrm{T}}$ is a rearranging operator, where $\mathbf{A}_{k:d,k}$ is the sub-matrix in $\mathbf{A}$ at row k to d and column k (see e.g. [8]). In [1,2], the mutation distribution is parameterized by (8). Sun et al. [2] proved in the case of the parameterization (8) that the exact FIM of $\pi(\mathbf{x} \mid \theta)$ becomes a block-diagonal matrix $\mathrm{diag}(\mathbf{F}_0, \dots, \mathbf{F}_d)$ whose first block $\mathbf{F}_0$ is identical to $\mathbf{C}^{-1}$ and $k+1$th $(1 \leq k \leq d)$ block $\mathbf{F}_k$ is given by

$$\mathbf{F}_k = a_{k,k}^{-2}\mathbf{u}_k\mathbf{u}_k^{\mathrm{T}} + (\mathbf{C}^{-1})_{k:d,k:d} \ , \tag{9}$$

where $a_{i,i}^{-1}$ is the reciprocal of the ith diagonal element of $\mathbf{A}$, or identically, the ith diagonal element of $\mathbf{A}^{-1}$, and $\mathbf{u}_k$ is a $[d-k+1]$-dimensional column vector whose first element is one and all the other elements are zero. The required matrix inversion can be performed efficiently since $\mathbf{F}^{-1} = \mathrm{diag}(\mathbf{F}_0^{-1}, \dots, \mathbf{F}_d^{-1})$. Moreover, the inverse of each block and the natural evolution gradient can be computed efficiently by an iterative method proposed in [2].

Although an efficient procedure is vital in implementing it on a computer, it makes it difficult to capture the mechanism of the update of θ. To analyze how the NES adjusts the parameter θ and to compare the NES with the CMA-ES, we derive the analytical inverse matrix of the FIM and extract the explicit form of the natural evolution gradient update rule.

3.1 Inverse of the Fisher Information Matrix

First, we derive the inverse matrix of each diagonal block of the FIM. Obviously, $\mathbf{F}_0^{-1} = \mathbf{C}$. Let $\mathbf{v}_k$ denote a d-dimensional column vector whose kth element is one and all the other elements are zero, and $\mathbf{I}$ be the $[d-k+1]$-dimensional identity matrix. Then, the inverse matrix of $k+1$th diagonal block $\mathbf{F}_k$ of the FIM can be written as

$$\mathbf{E}_k = \left[\mathbf{0}\ \mathbf{I}\right] \mathbf{A}\left(\begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix} - \frac{1}{2}\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}\right)\mathbf{A}^{\mathrm{T}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} \ . \tag{10}$$

To see this, it suffices to show the product of $\mathbf{E}_k$ and $\mathbf{F}_k$ becomes $\mathbf{I}_{d+1-k}$. Now, rewriting $\mathbf{F}_k$ in the form

$$\mathbf{F}_k = \begin{bmatrix}\mathbf{0} \ \mathbf{I}\end{bmatrix} \left(\mathbf{A}^{-\mathrm{T}}\mathbf{A}^{-1} + a_{k,k}^{-2}\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}\right) \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} \tag{11}$$

and postmultiplying $\mathbf{F}_k$ by $\mathbf{E}_k$ we have $\mathbf{F}_k\mathbf{E}_k = \mathbf{I}$[1]. Therefore, $\mathbf{F}_k^{-1} = \mathbf{E}_k$.

3.2 Explicit Form of the Natural Evolution Gradient

The estimation $\delta\theta$ of the natural evolution gradient is given as a linear combination of the natural gradient of the log-likelihood for all samples. The partial derivative of the log-likelihood is

$$\frac{\partial}{\partial\theta_k}\ln\pi(\mathbf{x} \mid \theta) = \begin{cases}\mathbf{v}_k^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}) & \text{if } 1 \le k \le d,\\ \mathbf{v}_{m_k}^{\mathrm{T}}\mathbf{R}\mathbf{v}_{n_k} & \text{otherwise,}\end{cases} \tag{12}$$

where m_k and n_k are the row and column indices of $\mathbf{A}$ corresponding to the kth element of θ, such that $1 \le n_k \le m_k \le d$ and $m_k + \sum_{i=1}^{n_k-1} d+1-i = k-d$, and

$$\mathbf{R} = \mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}} - \operatorname{diag}(a_{1,1}^{-1}, \ldots, a_{d,d}^{-1}) \ . \tag{13}$$

The natural gradient of the log-likelihood is obtained by premultiplying the gradient by the inverse of the FIM, which results in

$$\left[(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{F}_0^{-\mathrm{T}} \ (\mathbf{R}_{1:d,1})^{\mathrm{T}}\mathbf{F}_1^{-\mathrm{T}} \ (\mathbf{R}_{2:d,2})^{\mathrm{T}}\mathbf{F}_2^{-\mathrm{T}} \ \ldots \ (\mathbf{R}_{d:d,d})^{\mathrm{T}}\mathbf{F}_d^{-\mathrm{T}}\right]^{\mathrm{T}} \ . \tag{14}$$

Since $\mathbf{F}_0^{-1} = \mathbf{C}$, $\mathbf{F}_0^{-1}\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}) = (\mathbf{x}-\mathbf{m})$. For $1 \le k \le d$, $\mathbf{R}_{k:d,k} = \begin{bmatrix}\mathbf{0} \ \mathbf{I}\end{bmatrix}\mathbf{R}\mathbf{v}_k$ and

$$\mathbf{F}_k^{-1}(\mathbf{R}_{k:d,k}) = \begin{bmatrix}\mathbf{0} \ \mathbf{I}\end{bmatrix}\mathbf{A}\Big(\operatorname{tril}(\mathbf{S}) - \frac{1}{2}\operatorname{diag}(s_1, \ldots, s_d) - \frac{1}{2}\mathbf{I}\Big)\mathbf{v}_k \ , \tag{15}$$

where $\mathbf{S} = \mathbf{A}^{-1}(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}}$, s_k is the kth diagonal element of $\mathbf{S}$, and $\operatorname{tril}(\mathbf{S})$ denotes the lower triangular matrix whose (i, j) element is identical to the (i, j) element of $\mathbf{S}$ if $i \ge j$, zero otherwise[2].

Letting the estimated natural evolution gradient $\delta\theta$ be expressed in the block form $\delta\theta = -\left[\delta_0^{\mathrm{T}} \ \delta_1^{\mathrm{T}} \ \ldots \ \delta_d^{\mathrm{T}}\right]^{\mathrm{T}}$, then $\delta_0 = \sum_{i=1}^{\lambda} \mathrm{w}_{R_i}(\mathbf{x}_i - \mathbf{m})$ and

$$\delta_k = \begin{bmatrix}\mathbf{0} \ \mathbf{I}\end{bmatrix}\mathbf{A}\Big(\operatorname{tril}(\mathbf{Y}) - \frac{1}{2}\operatorname{diag}(y_1, \ldots, y_d) - \frac{\sum_{i=1}^{\lambda}\mathrm{w}_i}{2}\mathbf{I}\Big)\mathbf{v}_k \tag{16}$$

for $1 \le k \le d$. Here $\mathbf{Y} = \sum_{i=1}^{\lambda}\mathrm{w}_{R_i}\mathbf{A}^{-1}(\mathbf{x}_i-\mathbf{m})(\mathbf{x}_i-\mathbf{m})^{\mathrm{T}}\mathbf{A}^{-\mathrm{T}}$ and y_i is the ith diagonal element of $\mathbf{Y}$.

---

[1] Since $\mathbf{A}$ is lower triangular, $\mathbf{A}^{-1}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{A}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{A}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{A}^{-1}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right]$ and $\mathbf{v}_k^{\mathrm{T}}\mathbf{A}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right] = a_{k,k}\mathbf{v}_k^{\mathrm{T}}$, and then the product $\mathbf{F}_k\mathbf{E}_k$ reduces to $\mathbf{F}_k\mathbf{E}_k = \mathbf{I} + a_{k,k}^{-1}\mathbf{u}_k\mathbf{v}_k^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] - \frac{1}{2}a_{k,k}^{-1}\mathbf{u}_k\mathbf{v}_k^{-\mathrm{T}}\mathbf{A}^{\mathrm{T}}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] - \frac{1}{2}a_{k,k}^{-1}\mathbf{u}_k\mathbf{v}_k^{-\mathrm{T}}\mathbf{A}^{\mathrm{T}}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{I}$.

[2] According to $\mathbf{A}^{-1}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{A}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{A}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{A}^{-1}\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}\end{smallmatrix}\right]$, $\mathbf{v}_k^{\mathrm{T}}\mathbf{A}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right] = a_{k,k}\mathbf{v}_k^{\mathrm{T}}$, $\operatorname{diag}(a_{1,1}^{-1}, \ldots, a_{d,d}^{-1})\mathbf{v}_k = a_{k,k}^{-1}\mathbf{v}_k$ and $\mathbf{v}_k^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{A}^{-\mathrm{T}} = \mathbf{v}_k^{\mathrm{T}}$, the product of $\mathbf{F}_k^{-1}$ and $\mathbf{R}_{k:d,k}$ is reduces to $\mathbf{F}_k^{-1}(\mathbf{R}_{k:d,k}) = [\mathbf{0} \ \mathbf{I}]\mathbf{A}(\left[\begin{smallmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{smallmatrix}\right]\mathbf{S} - \frac{1}{2}\mathbf{v}_k^{\mathbf{T}}\mathbf{S} - \mathbf{I} + \frac{1}{2}\mathbf{I})\mathbf{v}_k = [\mathbf{0} \ \mathbf{I}]\mathbf{A}(\operatorname{tril}(\mathbf{S}) - \frac{1}{2}\operatorname{diag}(s_1, \ldots, s_d) - \frac{1}{2}\mathbf{I})\mathbf{v}_k$.

3.3 Parameter Update Rules

We consider the update rules for $\mathbf{m}$ and $\mathbf{A}$ corresponding to $\theta^{t+1} = \theta^t - \eta \cdot \delta\theta^t$. Let $\mathbf{Y}^t = \sum_{i=1}^{\lambda} \mathrm{w}_{R_i} \mathbf{A}^{-1}(\mathbf{x}_i^t - \mathbf{m})(\mathbf{x}_i^t - \mathbf{m})^{\mathrm{T}} \mathbf{A}^{-\mathrm{T}}$, $\delta\mathbf{m}^t = \delta_0$ and $\delta\mathbf{A}^t$ be a $[d \times d]$-dimensional lower triangular matrix whose $(i,\, j)$ element is identical to the $i+1-j$th element of δ_j for $i \leq j$, zero for $i > j$. Then analogous update rules for $\mathbf{m}^{t+1} = \mathbf{m}(\theta^{t+1})$ and $\mathbf{A}^{t+1} = \mathbf{A}(\theta^{t+1})$ can be written as $\mathbf{m}^{t+1} = \mathbf{m}^t + \eta \cdot \delta\mathbf{m}^t$ and $\mathbf{A}^{t+1} = \mathbf{A}^t + \eta \cdot \delta\mathbf{A}^t$, namely,

$$\mathbf{m}^{t+1} = \mathbf{m}^t + \eta \sum_{i=1}^{\lambda} \mathrm{w}_{R_i} (\mathbf{x}_j^t - \mathbf{m}^t) \tag{17}$$

$$\mathbf{A}^{t+1} = \mathbf{A}^t + \eta \mathbf{A}^t \Big(\mathrm{tril}(\mathbf{Y}^t) - \frac{1}{2} \mathrm{diag}(y_1^t, \ldots, y_d^t) - \frac{\sum_{i=1}^{\lambda} \mathrm{w}_{R_i}}{2} \mathbf{I} \Big) \ . \tag{18}$$

Suppose that w_i sum to one. The covariance matrix $\mathbf{C}^{t+1} = \mathbf{A}^{t+1}(\mathbf{A}^{t+1})^{\mathrm{T}}$ is

$$\mathbf{C}^{t+1} = \mathbf{A}^t \Big(\eta \cdot \mathrm{tril}(\mathbf{Y}^t) - \frac{\eta}{2} \mathrm{diag}(y_1^t, \ldots, y_d^t) + \frac{2-\eta}{2} \mathbf{I} \Big) \cdot \Big(\eta \cdot \mathrm{tril}(\mathbf{Y}^t) - \frac{\eta}{2} \mathrm{diag}(y_1^t, \ldots, y_d^t) + \frac{2-\eta}{2} \mathbf{I} \Big)^{\mathrm{T}} (\mathbf{A}^t)^{\mathrm{T}} \ . \tag{19}$$

Here, since $\mathbf{Y}^t$ is symmetric, $\mathrm{tril}(\mathbf{Y}^t) + \mathrm{tril}(\mathbf{Y}^t)^{\mathrm{T}} - \mathrm{diag}(y_1^t, \ldots, y_d^t) = \mathbf{Y}^t$, which reduces the last equality to

$$\mathbf{C}^{t+1} = (\frac{2-\eta}{2})^2 \mathbf{C}^t + \eta \frac{2-\eta}{2} \mathbf{A}^t \mathbf{Y}^t (\mathbf{A}^t)^{\mathrm{T}} + \eta^2 \mathbf{A}^t \cdot \Big(\mathrm{tril}(\mathbf{Y}^t) - \frac{1}{2} \mathrm{diag}(y_1^t, \ldots, y_d^t) \Big) \Big(\mathrm{tril}(\mathbf{Y}^t) - \frac{1}{2} \mathrm{diag}(y_1^t, \ldots, y_d^t) \Big)^{\mathrm{T}} (\mathbf{A}^t)^{\mathrm{T}}. \tag{20}$$

Notice $\mathbf{A}^t \mathbf{Y} (\mathbf{A}^t)^{\mathrm{T}} = \sum_{i=1}^{\lambda} \mathrm{w}_{R_i} (\mathbf{x}_i^t - \mathbf{m})(\mathbf{x}_i^t - \mathbf{m})^{\mathrm{T}}$. This equality (20) and the update rule (17) together are similar to the update rules for the mean vector and the covariance matrix used in the CMA-ES combining global weighted recombination and rank-μ update except for the third summand of (20) and the learning rate. Since the NES directly updates the Cholesky decomposition $\mathbf{A}$ of the covariance matrix $\mathbf{C}$ and equality (20) is related to rank-μ update, update rule (18) can be considered as a variant of the Cholesky update in [10].

From this explicit form of the NES, we can clearly see the difference between the NES and the CMA-ES. One different point is the third term of equality (20) due to the Cholesky update rule (18). Another point is in the learning late. The third point is the existence of step-size adaptation. We leave it to future work to study how these differences affect the performance of their optimization process.

In addition to enable us to compare the NES with the CMA-ES, the explicit forms (17) and (18) of the parameter update of the NES can reduce the time complexity of a single iteration of the NES from $\mathcal{O}(\lambda d^3)$ to $\mathcal{O}(\lambda d^2 + d^3)$[3]. This is because we can compute the estimated natural evolution gradient without computing the FIM and its inverse, as well as natural actor-critic reinforcement learning [11].

[3] This is only a reduction if λ and d increase at the same time.

4 CMA-ES as a Variant of NESs

The result in the previous section says that the NES can be viewed as a variant of CMA-ESs using Cholesky update when using the parameterization (8). The original NES parameterizes the mutation distribution as (8) to ensure the positivity and symmetry of the covariance matrix. An interesting question is whether there is a parameterization such that more standard CMA-ES which update the covariance matrix rather than the Cholesky factor of it is formulated as a variant of NESs. In this section, we answer the question in the affirmative.

Let

$$\theta = \left[\mathbf{m}^{\mathrm{T}} \ \mathrm{vec}(\mathbf{C})^{\mathrm{T}}\right]^{\mathrm{T}} \tag{21}$$

be a $d(d+1)$-dimensional column vector consisting of all the elements of the mean vector $\mathbf{m}$ and the covariance matrix $\mathbf{C}$, where $\mathrm{vec}(\cdot)$ denotes a rearranging operator from a matrix to a column vector such that $\mathrm{vec}(\left[\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_d\right]) = \left[\mathbf{a}_1^{\mathrm{T}} \ \mathbf{a}_2^{\mathrm{T}} \ \dots \ \mathbf{a}_d^{\mathrm{T}}\right]^{\mathrm{T}}$ (see e.g. [8]). Let us consider the natural evolution gradient learning when using this parameterization.

Suppose that $\mathbf{C}$ is positive-definite and symmetric. The gradient of the log-likelihood of $\pi(\mathbf{x} \mid \theta)$ is

$$\nabla_\theta \ln \pi(\mathbf{x} \mid \theta) = \begin{bmatrix} \mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}) \\ \frac{1}{2}\mathrm{vec}(\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^{\mathrm{T}}\mathbf{C}^{-1} - \mathbf{C}^{-1}) \end{bmatrix} . \tag{22}$$

From (6), we have that the FIM of $\pi(\mathbf{x} \mid \theta)$ and its inverse matrix are, respectively,

$$\mathbf{F}(\theta) = \begin{bmatrix} \mathbf{C}^{-1} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{C}^{-1} \otimes \mathbf{C}^{-1} \end{bmatrix} \quad \text{and} \quad \mathbf{F}^{-1}(\theta) = \begin{bmatrix} \mathbf{C} & \mathbf{0} \\ \mathbf{0} & 2\mathbf{C} \otimes \mathbf{C} \end{bmatrix} , \tag{23}$$

where $\otimes$ is the Kronecker product. Therefore, the natural gradient of the log-likelihood of $\pi(\mathbf{x} \mid \theta)$ is

$$\mathbf{F}^{-1}(\theta)\nabla_\theta \ln \pi(\mathbf{x} \mid \theta) = \begin{bmatrix} (\mathbf{x}-\mathbf{m}) \\ \mathrm{vec}((\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^{\mathrm{T}} - \mathbf{C}) \end{bmatrix} . \tag{24}$$

Since the natural evolution gradient $\delta\theta$ estimated from the samples $\mathbf{x}_i$ in the same way as (7) is a linear combination of the natural gradient of the log-likelihood, the natural evolution gradient can be estimated by

$$\delta\theta = \begin{bmatrix} -\sum_{i=1}^{\lambda} \mathrm{w}_{R_i}(\mathbf{x}_i - \mathbf{m}) \\ -\mathrm{vec}(\sum_{i=1}^{\lambda} \mathrm{w}_{R_i}(\mathbf{x}_i-\mathbf{m})(\mathbf{x}_i-\mathbf{m})^{\mathrm{T}} - \sum_{i=1}^{\lambda} \mathrm{w}_i \mathbf{C}) \end{bmatrix} . \tag{25}$$

Therefore, in the case of the parameterization (21), if $\mathbf{C}(\theta^t)$ is symmetric and nonsingular, the natural evolution gradient update produces the next parameter $\theta^{t+1} = \theta^t - \eta\delta\theta^t$ by

$$\theta^{t+1} = \begin{bmatrix} (1-\eta\sum_{i=1}^{\lambda} \mathrm{w}_{R_i})\mathbf{m}^t + \eta\sum_{i=1}^{\lambda} \mathrm{w}_{R_i}\mathbf{x}_i^t \\ \mathrm{vec}((1-\eta\sum_{i=1}^{\lambda} \mathrm{w}_{R_i})\mathbf{C}^t + \eta\sum_{i=1}^{\lambda} \mathrm{w}_{R_i}(\mathbf{x}_i^t-\mathbf{m}^t)(\mathbf{x}_i^t-\mathbf{m}^t)^{\mathrm{T}}) \end{bmatrix} . \tag{26}$$

Suppose $\sum_{i=1}^{\lambda} \mathrm{w}_i = 1$, $\mathrm{w}_i \geq 0$ for all i and $0 \leq \eta < 1$. Starting with an initial parameter θ^0 for which $\mathbf{C}(\theta^0)$ is positive definite and symmetric, the covariance matrix $\mathbf{C}(\theta^t)$ with this update rule is positive definite and symmetric for each t and the supposition that $\mathbf{C}$ is nonsingular and symmetric always holds. By letting $\mathbf{m}^{t+1} = \mathbf{m}(\theta^{t+1})$ and $\mathbf{C}^{t+1} = \mathbf{C}(\theta^{t+1})$, we have

$$\mathbf{m}^{t+1} = (1-\eta)\mathbf{m}^t + \eta \sum_{i=1}^{\lambda} \mathrm{w}_{R_i} \mathbf{x}_i^t \quad \text{and} \tag{27}$$

$$\mathbf{C}^{t+1} = (1-\eta)\mathbf{C}^t + \eta \sum_{i=1}^{\lambda} \mathrm{w}_{R_i} (\mathbf{x}_i^t - \mathbf{m}^t)(\mathbf{x}_i^t - \mathbf{m}^t)^{\mathrm{T}} \ . \tag{28}$$

These update rules are the same as the CMA-ES combining global weighted recombination and rank-μ update except that $\mathbf{m}^t$ update and $\mathbf{C}^t$ update take a common learning rate η. Consequently, the CMA-ES can be considered as a variant of NESs using the parameterization (21) instead of (8), i.e., the CMA-ES implicitly utilizes the natural evolution gradient without the calculation of the FIM and its inverse matrix.

There are some remarks on the result:

1. In terms of natural gradient, the result justifies the form of the update rules in the CMA-ES, i.e., using $\mathbf{m}^t$ rather than $\mathbf{m}^{t+1}$ in (28) and using the same weights in the update of the covariance matrix as in that of the mean vector.
2. The differences $(\mathbf{C}^{t+1} - \mathbf{C}^t)/\eta$ in (20) and in (28) agree in the case $\eta \to 0$. This is because they represent the natural gradient and the natural gradient is independent of the choice of the parameterization. However, since a finite step ($\eta > 0$) in the natural gradient direction depends on the parameterization, the update rules do not agree for $\eta > 0$.
3. Since two partial derivatives $\frac{\partial}{\partial \theta_k} \int f(\mathbf{x})\pi(\mathbf{x} \mid \theta) d\mathbf{x}$ and $\frac{\partial}{\partial \theta_k} \int (f(\mathbf{x}) - b_k)\pi(\mathbf{x} \mid \theta) d\mathbf{x}$ agree for any b_k, $\sum_{i=1}^{\lambda} (f(\mathbf{x}_i)/\lambda \mathbf{I} - \mathrm{diag}(b_1, \ldots, b_d)) \nabla_\theta \ln \pi(\mathbf{x}_i \mid \theta)$ is an unbiased estimator of the gradient as well as (4). When $b_i = 0$ for $1 \leq i \leq d$, $b_i = 1/\lambda$ for $d+1 \leq i \leq d+d^2$, and ranking-based fitness shaping (5) is used, the resulting update rule for $\mathbf{C}$ changes from (28) and, if $\mathbf{C}$ is positive-definite, the new one is $\mathbf{C}^{t+1} = \mathbf{C}^t + \eta \sum_{i=1}^{\lambda} (\mathrm{w}_{R_i} - 1/\lambda)(\mathbf{x}_i^t - \mathbf{m}^t)(\mathbf{x}_i^t - \mathbf{m}^t)^{\mathrm{T}}$. This is similar to Active-CMA [12] without rank-one update.
4. Fitness shaping is fundamental. If it is not used and the function values nearly vanish ($|f(\mathbf{x}_i)| \ll 1$), the natural gradient does and the parameter is not updated. An affine type fitness shaping $f(\mathbf{x}) \to a \cdot f(\mathbf{x}) + b$ does not affect the direction but does the length of the natural gradient. For example, a fitness shaping $f(\mathbf{x}_i)/\sum_{j=1}^{\lambda} f(\mathbf{x}_i)$ does not affect the direction of the natural gradient, but it normalizes the length in terms of the sum of the weights and shares the property with (5) that their values sum to one. However, in general, ranking-based fitness shaping (5) influences both the length and the direction of the natural gradient. In addition, so does the different learning rates (step size) for the mean vector and the covariance matrix in the CMA-ES. They might be important future works for further theoretical foundation of the CMA-ES.

5 Conclusion

These results state that the CMA-ES with global weighted recombination and rank-μ update can be formulated as the NES using the parameterization ([21]) and that their update rules corresponding to the mean and the covariance matrix of the mutation distribution are similar. The difference between the NES and the CMA-ES is essentially only in the parametrization of the mutation distribution. This is in agreement with the similar performances of them reported in [2].

The results also say that the CMA-ES can be formulated as a natural gradient learning which adjusts the parameter θ to minimize the average fitness $J(\theta)$ under the mutation distribution $\pi(\cdot \mid \theta)$. The theoretical foundation is important and profitable to justify, to understand the behavior of the CMA-ES, especially to analyse its convergence behavior, because the research dealing with the convergence properties of stochastic gradient learning (e.g. [13,14]) may help to investigate the convergence behavior of the CMA-ES. Besides, this makes it easier to compare the CMA-ES with gradient based methods. We may be able to draw inspiration from such comparison about when and how evolutionary algorithms perform better than gradient based methods on rugged functions.

Future work would focus on studying how the differences between the NES and the CMA-ES affect the performances of their algorithms. In particular, it is interesting how we can treat the concept of step-size adaptation in the framework. This might lead to further understanding of the CMA-ES and could possibly help to construct the convergence theory of the CMA-ES. Another future work could be to incorporate ideas used in gradient based online learning into the covariance matrix adaptation. For example, the concepts of *optimal baseline* and *importance sampling* are integrated in the natural evolution strategies as optimal fitness baseline and importance mixing [2]. Furthermore, it is possibly useful to introduce learning rate adaptation [3] and to combine other gradient methods such as natural conjugate gradient methods used in several learning problems [15,16,17].

Acknowledgement

The authors are grateful to Nikolaus Hansen for his helpful comments.

References

1. Wierstra, D., Schaul, T., Peters, J., Schmidhuber, J.: Natural evolution strategies. In: Proceedings of CEC 2008, pp. 3381–3387 (2008)
2. Sun, Y., Wierstra, D., Schaul, T., Schmidhuber, J.: Efficient natural evolution strategies. In: Proceedings of GECCO 2009, pp. 539–545 (2009)
3. Amari, S.: Natural gradient works efficiently in learning. Neural Computation 10(2), 251–276 (1998)
4. Hansen, N., Ostermeier, A.: Completely deranomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)

5. Hansen, N., Müller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
6. Amari, S., Nagaoka, H.: Methods of Information Geometry. American Mathematical Society (2007)
7. Amari, S., Douglas, S.: Why natural gradient? In: Proceedings of ICASSP, pp. 1213–1216 (1998)
8. Harville, D.A.: Matrix Algebra From a Statistician's Perspective. Springer, Heidelberg (2008)
9. Murray, M., Rice, J.: Differential geometry and statistics. Chapman & Hall/CRC, Boca Raton (1993)
10. Suttorp, T., Hansen, N., Igel, C.: Efficient covariance matrix update for variable metric evolution strategies. Machine Learning 75(2), 167–197 (2009)
11. Peters, J., Schaal, S.: Natural actor-critic. Neurocomputing 71, 1180–1190 (2008)
12. Jastrebski, G.A., Arnold, D.V.: Improving evolution strategies through active covariance matrix adaptation. In: Proceedings of CEC 2006, pp. 9719–9726 (2006)
13. Kuan, C.M., Hornik, K.: Convergence of learning algorithms with constant learning rates. IEEE Transactions on Neural Networks 2(5), 484–489 (1991)
14. Kushner, H.J., Yin, G.G.: Stochastic approximation and recursive algorithms and applications. Springer, Heidelberg (2003)
15. Gonzàlez, A., Dorronsoro, J.R.: Natural conjugate gradient training of multilayer perceptrons. Neurocomputing 71(13-15), 2499–2506 (2008)
16. Honkela, A., Tornio, M., Raiko, T., Karhunen, J.: Natural conjugate gradient in variational inference. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part II. LNCS, vol. 4985, pp. 305–314. Springer, Heidelberg (2008)
17. Nishimori, Y., Akaho, S., Plumbley, M.D.: Natural conjugate gradient on complex flag manifolds for complex independent subspace analysis. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) ICANN 2008, Part I. LNCS, vol. 5163, pp. 165–174. Springer, Heidelberg (2008)

A Fine-Grained View of GP Locality with Binary Decision Diagrams as Ant Phenotypes

James McDermott, Edgar Galván-Lopéz, and Michael O'Neill

Natural Computing Research & Applications group,
University College Dublin, Ireland
jamesmichaelmcdermott@gmail.com, edgar.galvan@ucd.ie, m.oneill@ucd.ie

Abstract. The property that neighbouring genotypes tend to map to neighbouring phenotypes, i.e. locality, is an important criterion in the study of problem difficulty. Locality is problematic in tree-based genetic programming (GP), since typically there is no explicit phenotype. Here, we define multiple phenotypes for the artificial ant problem, and use them to describe a novel fine-grained view of GP locality. This allows us to identify the mapping from an ant's behavioural phenotype to its concrete path as being inherently non-local, and show that therefore alternative genetic encodings and operators cannot make the problem easy. We relate this to the results of evolutionary runs.

Keywords: Genetic programming, phenotype, locality, problem difficulty, artificial ant.

1 Introduction

In evolutionary computation (EC), the *genotype* is the data structure, belonging to an individual, upon which the genetic operators work. It is mapped by some process to a *phenotype*, which in some sense represents the developed individual. Typically it is the phenotype which is evaluated for fitness, for example.

Some genetic programming (GP) variants have distinct genotypes and phenotypes, for example Cartesian GP [1], linear GP [2], grammatical evolution (GE) [3], and others. The case of standard, tree-structured GP is different. Depending on one's viewpoint, one might say that the genetic operators work directly on the phenotype, that the genotype-phenotype mapping is the identity map, or simply that no phenotype exists. Some methods of studying and comparing representations, common and useful in other areas of EC, turn out to be inapplicable to standard GP for this reason. One example is *locality*, an important measure of the behaviour of the genotype-phenotype mapping commonly used as an indicator of problem difficulty [4]. Some authors resort to measuring the locality of the genotype-fitness mapping as a substitute [5,6,7], but this does not tell the whole story. In particular, identifying a badly-behaved genotype-fitness mapping does not give us any clue about whether and how the mapping might be improved. Using phenotypes (e.g. [4]) can help to identify the components of the algorithm (e.g. a particular aspect of the encoding) responsible for the overall bad behaviour, or to conclude that no improvement is possible. In GP, it is possible to see program semantics or behaviour as a phenotype [8,9,10,11], and this is the approach taken here. GP locality has not been previously studied using such a definition of phenotypes.

Although the relatively poor performance of various GP techniques on the artificial ant benchmark cannot be regarded as an open problem [5], the study of locality can still teach us something new. The aims of this paper are thus to propose a fine-grained view of locality using two abstract, operator- and encoding-independent definitions of ant phenotypes; to use this model to study locality; and to relate the results with GP performance. Our methods may be applicable in a general way to other problems also.

The next section analyses previous work on locality and problem difficulty. In Sect. 3 we explain our view of locality, which allows for multiple phenotypes, and we give two distinct definitions for ant phenotypes. One of these is based on binary decision diagrams (BDDs), and an introduction to this topic is presented in Sect. 3.1. We present experiments on locality and evolutionary runs, and results, in Sect. 4; our conclusions are in Sect. 5.

2 Previous Work

Many authors have used landscape-oriented problem difficulty measures to try to predict search performance in EC. Examples include landscape correlation measures [12], epistasis [13], proportion of optima per program size [5], fitness-distance correlation (FDC) [14], FDC extensions including fitness clouds and negative slope coefficient [15,16], and locality and distance distortion [4].

In many of these cases, a proposed measure of difficulty (e.g. [14]) is followed by a counter-example (e.g. [17]): a problem which is easy to solve but is predicted to be difficult, or vice versa. In some cases the measure is then repaired or improved in some way (e.g. [16]), to be followed by new counter-examples (e.g. [18]). In addition to false predictions, there are problems of practicality. Jansen [19] summarised the situation for several such measures and demonstrated the key problems, including the requirement for very large sample sizes, unreliability of measures and counter-examples.

Despite this, research continues into methods of classifying problem difficulty, often with the aim of showing that one encoding or operator should be expected to improve performance relative to another. Here, we take the attitude that although no measure can predict performance perfectly, there are known results which can be explained in terms of problem difficulty, for example the poor performance of mutation-only GE relative to GP [20]. Such explanations contribute to an overall understanding of what makes EC work.

We concentrate on just one type of problem difficulty measure, *locality*, which in the general sense means the preservation of neighbourhood by a mapping. Rothlauf defines locality over the GA genotype-phenotype mapping and uses it to shed new light on several problems [4]. Although Rothlauf gives a numerical definition for the average locality of an encoding/mutation operator combination

[4] (p. 77), we will not use it in this paper since our aim is not to compare the locality of different encodings or operators. Instead, we will examine *the distribution of distances between neighbours after mapping* from one space to another. That is, we will study locality by taking pairs of neighbours, mapping each to a new space (i.e. genotypes to phenotypes, genotypes to fitness, or phenotypes to fitness), and looking at the distance between them in the new space.

3 A Fine-Grained View of Locality

In the abstract sense, locality refers to the preservation of neighbourhood by any mapping, not just the genotype-phenotype mapping common in many forms of EC. Therefore, when researchers (motivated by the absence of explicit phenotypes) study the preservation of neighbourhood by the genotype-fitness mapping in GP [5,6,7], this should be regarded as study of locality also. By defining explicit phenotypes in this paper, we aim to separate the genotype-fitness mapping into its component parts and study them separately.

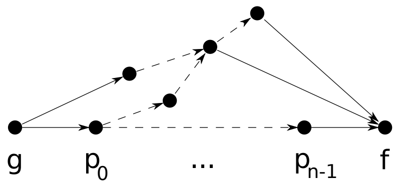


Fig. 1. Genotype g, phenotypes p_i, and fitness f. Arrows represent causality: for example, the calculation of p_0 depends on g, but g cannot be calculated from f.

Although it is common to think of each individual as having a single phenotype, a more general definition is possible: *any data structure which is calculated from the genome and which contributes to the calculation of fitness* may be seen as part of the "extended phenotype" [21]. The most general case is shown in Fig. 1. Each component of the mapping can, ideally, be studied separately.

In this study we focus on the artificial ant problem domain. Here, the problem is to find a program that can navigate a path of food laid out in cells on a grid [22] (pp. 147–155). The terminal set is `{move, right, left}`: these actions move the ant forward one square, and turn 90° to the right or left, respectively. Each consumes one time unit. The function set is `{iffoodahead, prog2, prog3}`. The first is a conditional: it executes its first argument if the ant perceives food directly ahead, and the second otherwise. The two remaining functions execute their two or three arguments in sequence. The most common grid layout is the *Santa Fe ant trail*, consisting of 89 food cells in a 32x32 grid, with the path characterised by twists and gaps. 600 time-steps are allowed.

The remainder of this section presents *two* definitions of ant phenotypes. Section 3.1 describes p_0, a phenotype based on binary decision diagrams, which represents the ant's *behaviour* in an encoding-independent way. Section 3.2 describes p_1, a cell-sequence phenotype, which represents the ant's *path* concretely. This leads to the overall model $g \rightarrow p_0 \rightarrow p_1 \rightarrow f$.

3.1 Binary Decision Diagram Phenotypes

One definition for ant phenotypes is suggested by [8,9]: an ant's behaviour is represented in an abstract form, inspired by the idea of stateful binary decision diagrams (BDDs) [23]. BDDs are a formalism for representing boolean functions. Any Boolean function composed of variables X_0, X_1, etc. and functions AND, OR, and NOT, for example, can be alternatively represented using a BDD.

Initially, a BDD may be thought of as a tree. At the root lives X_0, and it has two children each corresponding to X_1. At layer n live 2^n nodes corresponding to variable X_n. At the very bottom layer live nodes labelled 0 and 1. The essential idea is similar to that of a finite state machine. To evaluate a BDD, one traverses from the root, at each node choosing which of its two outgoing edges (labelled "high" and "low") to follow, depending on the value of the node's corresponding variable. The connectivity (i.e. the edges) ensures that the node one reaches at the end (0 or 1) is the value of the boolean function for the given variable values. In practice, it is common to use *reduced* BDDs, in which redundancies are eliminated: the BDD then no longer has a tree structure, since two divergent paths may re-join at a lower level.

Our definition of BDD-based ant phenotypes is similar but not identical to that of Beadle and Johnson [8,9]. The ant's behaviour is represented as a type of BDD: each node contains a sequence of zero or more action commands (left, right, and move), and each branch represents an if-statement. Branches rejoin after execution of an if-statement. In the ant problem there is only one "variable", the result of the `iffoodahead` predicate. This variable is *stateful*: it varies during an ant's run, so it is necessary to use multiple layers of nodes to represent behaviour. Since the ant's behaviour depends on the order in which it perceives cells, it is not possible to re-order the BDD (as for BDDs in other contexts) without altering behaviour. The mapping from genotype to BDD-phenotype is thus unambiguous. The algorithm for performing the mapping is omitted due to space constraints: code is available at `http://skynet.ie/~jmmcd/representations.html`. BDD-phenotypes are illustrated in Fig. 2.

We can equivalently write our BDD-phenotypes as strings. `L`, `R`, and `M` represent left, right, and move commands. A branch, conditional on the presence of food, is represented by an `<X,Y>` construct, where `X` and `Y` represent the two branches. We invert the standard BDD convention that the negative branch is written first, since in GP trees the positive branch of an if-statement is written first. The convention is arbitrary and the only effect of changing it is to make the BDDs easier to read.

Note that conversion from genotype to phenotype is not a simple matter of altering symbols. The sequencing implied by the `prog2` and `prog3` functions is abstracted away, and this allows some distinct genotypes to map to identical phenotypes. Several types of simplification are also required to obtain an abstract, canonical representation of ant behaviour:

- When one if-statement is nested directly inside another, one or other branch of the inner one will never be executed. That is, we can replace `<<X,Y>W,Z>` with `<XW,Z>`, and we can replace `<X,<Y,Z>W>` with `<X,ZW>` (here `W`, `X`, `Y` and `Z` are arbitrary sequences of actions).
- When the two branches of an if-statement end with the same action, it can be brought outside the branch. That is, we can replace `<XY,ZY>` with `<X,Z>Y`.
- As a result of the above simplifications, it may happen that an if-statement has two empty branches: `<,>` can be removed.

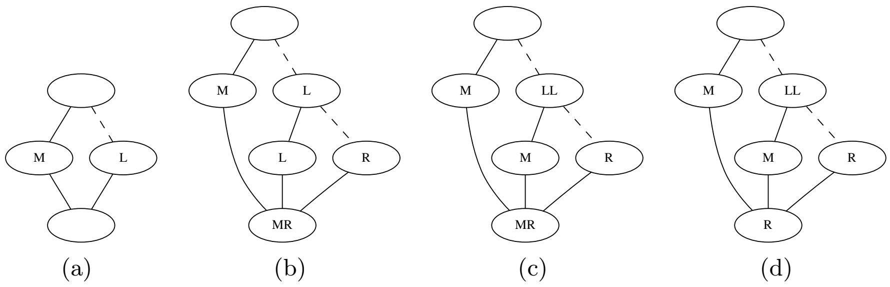


Fig. 2. BDD phenotypes. The positive and negative branches of an if-statement are drawn with solid and dashed lines respectively. Edges from non-branching nodes are drawn with solid lines. In (a) a simple example: the genotype is **`(iffoodahead move left)`** and the phenotype `<M,L>`. In (b) the genotype is **`(prog3 (iffoodahead move (prog2 left (iffoodahead left (iffoodahead move right)))) move right)`**. This translates to the phenotype `<M,L<L,<M,R>>>MR`. After removal of a redundant branch, we get `<M,L<L,R>>MR`, as shown. Phenotypic neighbours can have divergent fitness values: individual (c) has fitness 89, but its neighbour (d), created by a single phenotypic mutation, has fitness 1.

This representation can be run inside a suitable interpreter, and it will give the same ant path and fitness as the original GP genotype. Crucially, this representation is sufficiently abstract that it could also be used as a phenotype for several other types of GP in which the ant problem might be run, including GE, Cartesian GP, evolutionary programming (i.e. a finite state machine encoding), and others. The BDD phenotype representation also admits a (non-unique) backward mapping to genotype, not used in this paper.

We will measure distance between BDD-phenotypes using two string distance measures. *Normalised compression distance* has been previously used in diversity analysis [24]; *string edit distance* is well-known; both are general-purpose measures. This choice is made because no single distance measure of which we are aware is naturally suited for distances between BDD-phenotypes. Each measure gives at best an approximation to the "true" distance between a pair of phenotypes. However, improved measures must be left for future work.

We define BDD-phenotypic neighbourhood via “minimal edits” or mutations on phenotypes, which consist of insertion, deletion, or editing of any of the action commands `L`, `R` and `M`. Fig. 4 shows that phenotypic neighbours thus defined can have divergent fitness values. Our reasoning in considering only insertion, deletion and editing of the action commands is that non-minimal, structure-altering phenotypic edits will also lead *a fortiori* to divergent fitness values.

Examples of phenotypes and canonicalisation are shown in Fig. 2. We can illustrate the non-locality of the phenotype-fitness mapping: Fig. 2(c) shows the phenotype of an individual which solves the Santa Fe problem, i.e. has fitness 89, and Fig. 2(d) an individual created by a single phenotypic mutation which has fitness 1. The large change in fitness occurs because behaviour at each step depends on position and orientation after previous steps, and is repeated multiple times. Small changes in behaviour are thus multiplied.

3.2 Cell-Sequence Phenotypes

Another definition for an ant’s phenotype is *the time-indexed sequence of cells it visits*, as illustrated in Fig. 3. This leads immediately to a natural definition of phenotypic distance: $d(a, b) = \sum_{t=0}^{T} d_c(a_t, b_t)$, where a_t and b_t are the position of ants a and b at time t, T is the maximum time, and d_c is a distance metric between cell positions, such as the toroidal taxi-driver’s distance. The position of the food pellets mediates the behaviour of the ant but is not used in the calculation of these metrics. Note that a small change in cell phenotype will necessarily induce only a small change in fitness. The mapping from cell phenotype to fitness is, in other words, highly local by definition.

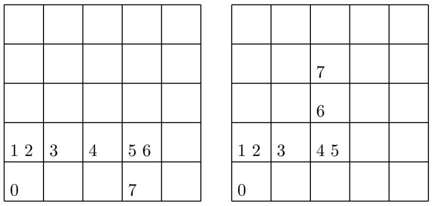


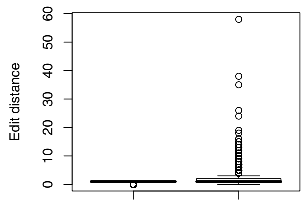
Fig. 3. Two ants’ cell-sequence phenotypes. An integer t in a cell indicates that the ant was in that cell at time t. The food pellets are not shown. The distance d between these two ants is calculated as the sum of toroidal taxi-driver distances between corresponding points in the paths. Where the ants coincide (as for $t < 5$) the distance is 0. For $t = 5$ the distance is 1, for $t = 6$ it is 2, and for $t = 7$ it is 3 (take a shortcut through the bottom, emerging at the top), so the total distance is $0+0+0+0+0+1+2+3 = 6$.

4 Experiments and Results

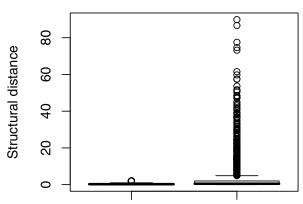
Here we report the results of two experiments. The central hypothesis is that poor results in evolutionary runs can be explained in terms of locality measures.

We consider locality first, sampling individuals, mutating them using several methods, and measuring distances between pairs in several spaces. In every case, 1000 individuals were randomly generated, and a single mutation of the type shown (one-point, subtree, or phenotypic) was performed on each, yielding 1000 pairs of genotypic (respectively phenotypic) neighbours. The distance between pairs in the genotypic (respectively phenotypic, fitness) space was then recorded. Note that one-point mutation changes a single node per individual. Subtree mutation is a standard operator. Phenotypic mutation works as in Sect. 3.1.

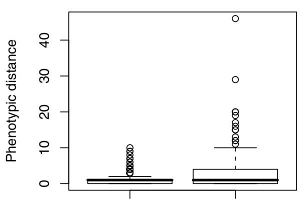
Figs. 4(a) and 4(b) use two measures of genotypic distance (tree-edit distance and structural distance [25]) to show that different operators give different genotypic step-sizes. Fig. 4(c) shows that genotypic neighbours map to similar phenotypes when neighbourhood is defined by one-point mutation, but often do not when it is defined by subtree mutation. Fig. 4(e) (left and centre) shows that genotypic neighbours can have highly divergent fitness values, when genotypic neighbourhood is defined by either mutation operator. Finally, 4(e) (right) shows that BDD-phenotypic neighbours can have highly divergent fitness values also.



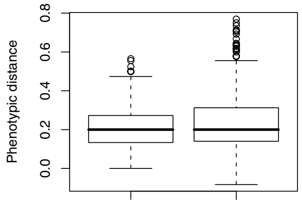
(a) Genotypic step-size



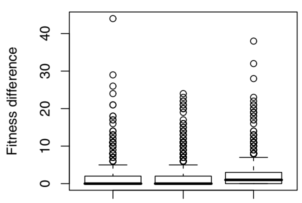
(b) Genotypic step-size



(c) $g \rightarrow p_0$ (string-edit)



(d) $g \rightarrow p_0$ (NCD)



(e) $g \rightarrow f$, $g \rightarrow f$, $p_0 \rightarrow f$

Fig. 4. Different operators have different step-sizes ((a) and (b)). The genotype-to-BDD-phenotype mapping ((c) and (d)) can therefore be well- or badly-behaved, depending on the operator used to define neighbourhood. The genotype-to-fitness mapping ((e), left and centre) is badly-behaved for both. The BDD-phenotype-to-fitness mapping ((e), right) is badly-behaved.

The definition of non-trivial phenotypes allows us a fine-grained view of mapping behaviour. Recall that our model of the mapping is $g \rightarrow p_0 \rightarrow p_1 \rightarrow f$, where p_0 is the BDD-phenotype and p_1 is the cell-sequence. We already know that the overall map $g \rightarrow f$ is badly-behaved (confirmed by Fig. 4(e), left and centre): we can now seek to explain which of its components are responsible. The $p_1 \rightarrow f$ mapping has high locality by definition (see Sect. 3.2) and so is not to blame. However, $p_0 \rightarrow f$ has been shown to be badly-behaved (see Fig. 4(e), right, and recall Fig. 2(d)). Taking these results together shows that it is $p_0 \rightarrow p_1$, the mapping from the ant's abstract behaviour to its cell-sequence, which is badly behaved and at least partly responsible for the behaviour of the overall $g \rightarrow f$ mapping. Note, however, that subtree mutation can also cause bad behaviour in the $g \rightarrow p_0$ mapping (see Fig. 4(c)).

Table 1. Artificial Ant performance measured over 100 runs. Higher is better.

Algorithm/Setup	Mean Best Fitness	Std. Dev.	Hits
GP: no xover; subtree mut	60.54	9.98	6/100
GP: no xover; onepoint mut	51.24	7.22	0/100
GP: 9010 xover; subtree mut	61.27	10.10	5/100
GP: 9010 xover; onepoint mut	50.97	7.92	0/100
Random search:	60.86	13.93	1/100

In Table 1 we show the results of evolutionary runs using standard GP. The aim is to confirm that GP techniques perform poorly on the ant problem. 100 runs were performed with each setup, with population 500 and 50 generations. Typical parameters were used: 90/10 crossover probability 0.7, mutation probability 0.01 (but 1.0 for mutation-only GP), and maximum tree depth 7. A random search (implemented as a GP run of population 25,000 and 1 generation, with ramped half-and-half initialisation) is also reported for comparison. The only "knowledge" input to the random search was to avoid tree depths less than 4.

An ANOVA and pairwise t-tests were performed on the 100 best fitness values from each setup. Three set-ups (subtree mutation, crossover/subtree, and random search) each performed significantly better than the other two (one-point and crossover/one-point) ($p < 0.01$, Bonferroni correction for 10 pairwise t-tests). But the overall result is that GP performs poorly. There is little novelty in this: it reinforces the conclusion (in 1998) of Langdon and Poli [5], that the ant problem is difficult for many representations. Other representations including Cartesian GP and GE have since produced largely similar results [1,3].

It is noteworthy that subtree mutation does well relative to one-point, despite being more "randomising". It tends to take larger jumps in the search space. When a search space is badly-behaved, as here, highly local methods such as minimum-change operators lose any advantage they would have on smooth, well-behaved spaces. For difficult problems, then, random search tends to perform surprisingly well compared to more sophisticated algorithms.

5 Conclusions

We have proposed a fine-grained model of locality in the ant problem, in which the overall genotype-to-fitness map is broken up into three components with two intermediate phenotypes which have not been previously used in the study of locality. This new model allows us to identify the component—the map from the ant's abstract behaviour to its concrete path—responsible for the overall map's bad behaviour. We have performed various evolutionary runs and as expected we have added to Langdon and Poli's list of poor results [5] on this problem.

Our core conclusion is an attempt to explain these poor results: in the ant problem, the mapping from BDD-phenotypes to fitness is inherently badly-behaved. Since these BDD-phenotypes can function as encoding-independent behavioural phenotypes for many GP approaches to the problem, this result goes some way to explaining their universally poor performance. Thus, poor performance is not due to inadequate representations.

Although we have studied only the ant problem, the fine-grained model of locality proposed here may allow new insights into other problems also. Our definitions of phenotypes will not be directly usable in other problems, but possibilities are suggested by Beadle and Johnson's BDDs [8,9] and other semantic approaches. Other methods of characterising mapping behaviour, such as correlation analysis, might also benefit from similar fine-grained models.

Acknowledgments

Special thanks are due to Colin Johnson for an enlightening discussion of BDDs. JMcD is funded by the Irish Research Council for Science, Engineering and Technology under the Empower scheme. EG-L and MO'N thank the support of Science Foundation Ireland under Grant No. 08/IN.1/I1868.

References

1. Miller, J.F., Thomson, P.: Cartesian genetic programming. In: Poli, R., Banzhaf, W., Langdon, W.B., Miller, J., Nordin, P., Fogarty, T.C. (eds.) EuroGP 2000. LNCS, vol. 1802, pp. 121–132. Springer, Heidelberg (2000)
2. Brameier, M., Banzhaf, W.: Linear genetic programming. Springer, New York (2006)
3. O'Neill, M., Ryan, C., Keijzer, M., Cattolico, M.: Crossover in grammatical evolution. Genetic Programming and Evolvable Machines 4(1), 67–93 (2003)
4. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Physica-Verlag (2006)
5. Langdon, W.B., Poli, R.: Why ants are hard. In: Koza, J.R. (ed.) Proceedings of the Third Annual Conference on Genetic Programming, pp. 193–201. Morgan Kaufmann, Madison (1998)
6. Galván-Lopéz, E., McDermott, J., O'Neill, M., Brabazon, A.: Towards an understanding of locality in genetic programming. In: GECCO 2010: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation, Portland, Oregon, USA. ACM Press, New York (2010)
7. Galván-Lopéz, E., McDermott, J., O'Neill, M., Brabazon, A.: Defining locality in genetic programming to predict performance. In: CEC 2010: Proceedings of the 12th Annual Congress on Evolutionary Computation, Barcelona, Spain. IEEE Press, Los Alamitos (2010)

8. Beadle, L., Johnson, C.G.: Semantic analysis of program initialisation in genetic programming. Genetic Programming and Evolvable Machines 10(3), 307–337 (2009)
9. Beadle, L., Johnson, C.G.: Semantically driven mutation in genetic programming. In: Proceedings of IEEE Congress on Evolutionary Computing, pp. 1336–1342 (2009)
10. Nguyen, Q.U., Nguyen, X.H., O'Neill, M.: Semantic aware crossover for genetic programming: The case for real-valued function regression. In: Proceedings of the 12th European Conference on Genetic Programming, pp. 292–302. Springer, Heidelberg (2009)
11. Jackson, D.: Phenotypic diversity in initial genetic programming populations. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 98–109. Springer, Heidelberg (2010)
12. Hordijk, W.: A measure of landscape. Evolutionary Computation 4(4), 335–360 (1996)
13. Altenberg, L.: NK fitness landscapes. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation. IOP Publishing Ltd./Oxford University Press (1997)
14. Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, Albuquerque (1995)
15. Vanneschi, L., Tomassini, M., Collard, P., Verel, S., Pirola, Y., Mauri, G.: A comprehensive view of fitness landscapes with neutrality and fitness clouds. In: Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I. (eds.) EuroGP 2007. LNCS, vol. 4445, pp. 241–250. Springer, Heidelberg (2007)
16. Poli, R., Vanneschi, L.: Fitness-proportional negative slope coefficient as a hardness measure for genetic algorithms. In: Proceedings of GECCO 2007, London, UK, pp. 1335–1342 (2007)
17. Altenberg, L.: Fitness distance correlation analysis: An instructive counterexample. In: Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 57–64 (1997)
18. Vanneschi, L., Valsecchi, A., Poli, R.: Limitations of the fitness-proportional negative slope coefficient as a difficulty measure. In: Proceedings of the 11th Annual conference on genetic and evolutionary computation, pp. 1877–1878. ACM, New York (2009)
19. Jansen, T.: On classifications of fitness functions. In: Theoretical aspects of evolutionary computing, pp. 371–386. Springer, Heidelberg (2001)
20. Rothlauf, F., Oetzel, M.: On the locality of grammatical evolution. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 320–330. Springer, Heidelberg (2006)
21. Dawkins, R.: The Extended Phenotype. Oxford University Press, Oxford (1982)
22. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
23. Bryant, R.: Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys (CSUR) 24(3), 318 (1992)
24. Gomez, F.J.: Sustaining diversity using behavioral information distance. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, Montréal, Canada, pp. 113–120. ACM, New York (2009)
25. Tomassini, M., Vanneschi, L., Collard, P., Clergue, M.: A study of fitness distance correlation as a difficulty measure in genetic programming. Evolutionary Computation 13(2), 213–239 (2005)

Drift Analysis with Tail Bounds

Benjamin Doerr[1] and Leslie Ann Goldberg[2]

[1] Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
[2] Department of Computer Science, University of Liverpool, Ashton Bldg, Liverpool L69 3BX, UK

Abstract. We give a simple and short alternative proof of the multiplicative drift theorem published recently (Doerr, Johannsen, Winzen (GECCO 2010)). It completely avoids the use of drift theorems previously used in the theory of evolutionary computation. By this, its proof is fully self-contained.

The new theorem yields exactly the same bounds for expected run-times as the previous theorem. In addition, it also gives good bounds on the deviations from the mean. This shows, for the first time, that the classical $O(n \log n)$ run-time bound for the (1+1) evolutionary algorithm for optimizing linear functions holds with high probability (and not just in expectation). Similar improvements are obtained for other classical problems in the evolutionary algorithms literature, for example computing minimum spanning trees, finding single-source shortest paths, and finding Eulerian cycles.

1 Introduction

Drift analysis was introduced to the theory of evolutionary algorithms by He and Yao [18,19]. It soon became one of the strongest tools both for proving run-time guarantees for many evolutionary algorithms and for giving evidence that some algorithms cannot solve certain problems. See, e.g., [13,14,17,24,22] for some notable subsequent uses of this method.

While it has many successful applications, drift analysis is nevertheless a tool that is not liked by many researchers in evolutionary algorithms. We see three main reasons for this. (i) The drift theorem employed is considered a deep mathematical tool, and its proof [16] is not easy to understand. (ii) When used to prove upper bounds on the run-time of evolutionary algorithms, drift analysis only yields bounds for the expected run-time. Hence, even for simple test problems like the linear functions one, we do not even know if the run-time bound of $O(n \log n)$ also holds with probability $1 - o(1)$. (iii) To successfully employ drift analysis, one has to find a suitable potential function. This is often difficult. After having found one, proving that it satisfies the drift condition often involves tedious calculations.

In [8,7], some progress concerning the third point was made. By giving a multiplicative drift theorem, more natural proofs for many problems were obtained. Since the multiplicative drift theorem was derived from the classical one, naturally no progress concerning the difficulties (i) and (ii) was made.

In this work, we solve the remaining two problems in an elegant way. We give a simple and short proof for the multiplicative drift theorem. It is fully self-contained up to using Markov's inequality. It yields the same bounds for expected run-times as the work of [8,7], which again yielded at least as good bounds for many classical problems as did earlier work.

However, by not implicitly relying on classical additive drift theorems, our proof also solves the second issue of obtaining bounds that hold with high probability. This in particular shows that the classical (1+1) evolutionary algorithm optimizes an arbitrary linear function in time $O(n \log n)$ with high probability (that is, with probability $1 - n^{-c}$, where c can be any constant). If we care for the implicit constants, then building on Jägersküpper's analysis we obtain that an upper bound of $1.39en\ln(n)(1 + o(1))$ is attained with probability $1 - o(1)$.

Similarly, we obtain that the known bounds on the expected run-time hold with high probability also for the problems of computing minimum spanning trees and minimum weight bases in matroids, single-source shortest paths (with a single-criterion fitness function), and Eulerian cycles.

This paper is organized as follows. In the following section, after introducing some elementary notation, we state and prove a simple drift theorem that is suited to prove bounds for optimization times of evolutionary algorithms that hold with high probability. In the subsequent four sections, we apply this theorem to different classical problems.

2 Drift Analysis

In this section, we briefly describe drift analysis to the extent needed for our purposes. For a more general background, we refer to the papers cited above.

2.1 The (1+1) Evolutionary Algorithm

Let $n \in \mathbb{N}$. Let Ω_n be a search space and $f_n : \Omega_n \to \mathbb{R}$ be an objective function defined on Ω_n. We refer to n as the problem size. Without loss of generality, we may assume (and shall always in this paper) that f is to be *minimized.*

If there is little risk of confusion, we usually omit the subscript n and simply write Ω and f. Let $\Omega_{\text{opt}} \subseteq \Omega$ denote the set of optimal search points, that is, those which have a minimal f-value.

The randomized search heuristic we regard in this work is the well-known (1+1) EA. It starts with an initial solution x chosen uniformly at random from the search space Ω_n. In each iteration, from its existing solution x it generates a new solution x' by *mutation.* If $\Omega_n = \{0,1\}^n$ is the set of bit strings of length n, then mutation typically consists of flipping each bit of x with some probability p, often $p = 1/n$, independently. In other words, for each $i \in \{1, \ldots, n\}$ independently, we have $\Pr(x'_i = 1 - x_i) = p$ and $\Pr(x'_i = x_i) = 1 - p$.

In the subsequent *selection* step, if $f(x') \leq f(x)$, the EA *accepts* x' as solution, meaning that the next iteration starts with $x_{\text{new}} := x'$. Otherwise, the next iteration starts with $x_{\text{new}} := x$ unchanged. Since we are interested in analyzing how many iterations are necessary until an optimal solution is found, we do not specify a termination criterion here.

Algorithm 1. (1+1) EA with mutation probability p

```
1: Initialization: Choose $x \in \{0,1\}^n$ uniformly at random.
2: repeat forever
3:     Create $x' \in \{0,1\}^n$ by copying $x$.
4:     Mutation: Flip each bit in $x'$ independently with probability $p$.
5:     Selection: if $f(x') \leq f(x)$ then $x := x'$.
```

We should stress that the (1+1) EA typically is not used to actually solve difficult optimization problems. Here, one would rather choose more complex search heuristics. However, understanding the optimization behavior of the (1+1) EA often helps in predicting the one of more complicated EAs (which mostly are too complex to admit a rigorous theoretical investigation).

2.2 A Simple Drift Theorem with Tail Bounds

Definition 1. *Let $\nu : \mathbb{N} \to \mathbb{R}$ be monotonically increasing. We call $\Phi : \Omega_n \to \mathbb{R}$ a feasible ν-drift function for f_n and a given (1+1) EA, if the following conditions are satisfied.*

1. *$\Phi(x) = 0$ for all $x \in \Omega_{\mathrm{opt}}$;*
2. *$\Phi(x) \geq 1$ for all $x \in \Omega_n \setminus \Omega_{\mathrm{opt}}$;*
3. *there is a constant $\delta > 0$ (independent of n) such that for all $x \in \Omega_n \setminus \Omega_{\mathrm{opt}}$,*

$$E(\Phi(x_{\mathrm{new}})) \leq \left(1 - \frac{\delta}{\nu(n)}\right) \Phi(x),$$

where as above we denote by x_{new} the solution resulting from executing a single iteration (consisting of mutation and selection) with initial solution x.

As a semi-trivial example, note that if f is a linear function with coefficients at least one, then f itself is a feasible n-drift function for f and all mutation probabilities $p = c/n$, c constant. However, this often is not a very useful drift function.

When feasible drift functions exist, they allow an elegant analysis yielding upper bounds for the optimization time of EAs. The *optimization time* of a randomized search heuristic is usually defined to be the number of evaluations of the objective function f performed until the optimum is found. In case of the (1+1) EA, this is (apart from an additive deviation of one) equal to the number of mutation-selection iterations.

The following well-known theorem shows how the optimization time can be bounded using a drift function. Similar arguments appear in the context of coupling proofs. See, for example, [11, Section 5]. Much more is known about drift analysis. See, for example [16]. For a first use of drift analysis in the analysis of evolutionary algorithms, see [18]. For the first explicit use of a multiplicative version of drift in evolutionary computation, see [8].

Note that Theorem 1 gives a probability tail bound in addition to an upper bound on the expected optimization time. Also the tail bound is not new, but it seems to be unknown in the evolutionary algorithms literature. It can be applied to improve several previous results (cf. the following sections).

Theorem 1. *Let $\Phi : \Omega_n \to \mathbb{R}$. Denote by $\Phi_{\max} := \max\{\Phi(x) \mid x \in \Omega_n\}$ the maximum value of Φ. If Φ is a feasible ν-drift function (with implicit constant δ) for f_n and a given (1+1) EA, then the expected optimization time is at most*

$$\frac{\nu(n)}{\delta}(1 + \ln \Phi_{\max}).$$

Also, for any $c > 0$ (possibly depending on n), we have that the optimization time exceeds $\frac{\nu(n)}{\delta}(\ln \Phi_{\max} + c \ln n)$ with probability at most n^{-c}.

For the proof, we need the following well-known and elementary fact, which can, e.g., be found in [15, Problem 13(a) Section 3.11]. For completeness, we shall repeat the elementary proof, which is a simple re-ordering argument.

Lemma 1. *Let X be a random variable taking values in the non-negative integers. Then $E(X) = \sum_{i=1}^{\infty} \Pr(X \geq i)$.*

Proof. $E(X) = \sum_{i=1}^{\infty} i \Pr(X = i) = \sum_{i=1}^{\infty} \sum_{j=1}^{i} \Pr(X = i) = \sum_{j=1}^{\infty} \sum_{i=j}^{\infty} \Pr(X = i) = \sum_{j=1}^{\infty} \Pr(X \geq j)$. □

Proof (of Theorem 1). Fix an arbitrary initial solution $x_0 \in \Omega$ for the (1+1) EA and consider a run started with this initial solution. Denote by Φ_t the value of $\Phi(x)$ after t selection-mutation steps. Denote by T_{opt,x_0} the first time when the current solution x is optimal.

From the fact that Φ is a feasible ν-drift function, we have $E(\Phi_t) \leq (1 - \delta/\nu(n))^t \Phi_0 \leq (1 - \delta/\nu(n))^t \Phi_{\max} \leq \exp(-t\delta/\nu(n))\Phi_{\max}$, where in the last estimate we used the well-known inequality $1 + z \leq e^z$ valid for all $z \in \mathbb{R}$. By Lemma 1, the expected optimization time $E(T_{\text{opt},x_0})$ can be written as $E(T_{\text{opt},x_0}) = \sum_{t \geq 0} \Pr(\Phi_t > 0)$, which is at most $T + \sum_{t \geq T} \Pr(\Phi_t > 0) \leq T + \sum_{t \geq T} E(\Phi_t)$ for any T. Here, Markov's inequality was used to show that $\Pr(\Phi_t > 0) = \Pr(\Phi_t \geq 1) \leq E(\Phi_t)$ holds for all t. Let $T = \lceil \ln(\Phi_{\max})\nu(n)/\delta \rceil = \ln(\Phi_{\max})\nu(n)/\delta + \varepsilon$ for some $0 \leq \varepsilon < 1$. By our estimate above, we obtain

$$\begin{aligned} E(T_{\text{opt},x_0}) &\leq T + (1 - \delta/\nu(n))^T \Phi_{\max} \sum_{i=0}^{\infty} (1 - \delta/\nu(n))^i \\ &\leq \ln(\Phi_{\max})\nu(n)/\delta + \varepsilon \\ &\quad + (1 - \delta/\nu(n))^{\varepsilon} \exp(-(\delta/\nu(n)) \ln(\Phi_{\max})\nu(n)/\delta)\, \Phi_{\max}\, \nu(n)/\delta \\ &= \ln(\Phi_{\max})\nu(n)/\delta + \varepsilon + (1 - \delta/\nu(n))^{\varepsilon} \nu(n)/\delta \\ &\leq \ln(\Phi_{\max})\nu(n)/\delta + \nu(n)/\delta. \end{aligned}$$

The last estimate is valid for all $\varepsilon \in [0,1]$. This is easiest seen by checking it for $\varepsilon = 0$ and $\varepsilon = 1$ and noting that the term is convex in ε.

Similarly, we compute for any c and $T_c := \lceil (\nu(n)/\delta)(\ln(\Phi_{\max}) + c\ln(n)) \rceil$ that

$$\Pr(T_{\mathrm{opt},x_0} > T_c) = \Pr(\Phi_{T_c} > 0) \le E(\Phi_{T_c}) \le \exp(-T_c\delta/\nu(n))\Phi_{\max} \le n^{-c}.$$

□

The proof above uses the argument $E(\Phi_t) \le (1-\delta/\nu(n))^t\Phi_{\max}$ as in the so-called methods of expected weight decrease [23]. Then, however, a general argument is employed to derive from the information on $E(\Phi_t)$, $t \ge 0$, a useful bound on the random variable $\min\{t \mid \Phi_t < 1\}$. Note that a statement like $E(\min\{t \mid \Phi_t < 1\}) = \min\{t \mid E(\Phi_t) < 1\}$ usually is not true.

3 Linear Functions

As laid out in the introduction, determining the run-time of the (1+1) EA on linear functions is a classical test problem in the theory of evolutionary computation. It led to the introduction of drift analysis to this field. In this section, we show that all previous results on this problem, which all only give a bound on the expected run-time, also hold with high probability.

Let us sketch the state of the art for this problem. Recall that we regard the minimization problem (which is, of course, equivalent to the maximization version). For a unified presentation, let for each constant $c \in [1,2]$ and $x \in \{0,1\}^n$, $d_c(x) := \sum_{i=1}^{\lfloor n/2 \rfloor} x_i + c\sum_{i=\lfloor n/2 \rfloor+1}^{n} x_i$.

The first proof of the $O(n\log n)$ bound for the expected run-time was given in [10]. It shows what we would now phrase as follows. Let $\Phi(x) := d_2(x)$. If $\Phi(x) > 0$, then $E(\Phi(x_{\mathrm{new}})) \le (1-\varepsilon)\Phi(x)/n$ for some constant $\varepsilon > 0$. From this, without drift analysis present in the field, the authors still manage to derive the $O(n\log n)$ bound for the expected optimization time.

To use an additive drift theorem, He and Yao [19] showed that for $V(x) := n\ln(1+d_c(x))$, where c can be any constant in $[1,2] \setminus \{1\}$, the following is true. If x is not yet the optimum, then $E(V(x_{\mathrm{new}})) \le V(x) - \varepsilon$ for some constant ε. This leads to the same $O(n\log n)$ upper bound for the expected optimization time, but with a more insightful proof. An alternative proof of this result via multiplicative drift was given in [8]. There, it was shown that for $\Phi(x) = d_{5/4}(x)$, $E(\Phi(x_{\mathrm{new}})) \le ((1-\varepsilon')/n)\Phi(x)$ holds (for a suitable constant ε').

Using a clever averaging argument, Jägersküpper showed the following. Let $x_0, x_1, \ldots,$ denote the sequence of search points forming the one-point population after each iteration. Then $E(d_1(x_{t+1})) \le 0.736\, d_1(x_t)/n$. Via an additive drift theorem, this was used to prove an upper bound on the expected optimization time of $2.02en\ln(n)(1+o(1))$. Via the multiplicative drift theorem, Jägersküpper's estimate immediately yields an upper bound for the expected optimization time of $1.39en\ln(n)(1+o(1))$, cf. [7].

All results above hold for the standard (1+1) EA with mutation probability $p = 1/n$. Recently, the authors of this paper [3] showed that also for all other mutation probabilities $p = c/n$, c a constant, for each linear function f there is a drift function Φ and a constant $\varepsilon > 0$ such that for all x, $E(\Phi(x_{\text{new}})) \leq (1 - \varepsilon)\Phi(x)$. Hence again, the expected optimization time is bounded by $O(n \log n)$.

Note that for all these $O(n \log n)$ results, we may as well apply Theorem 1 and thus have the corresponding bound with high probability. This was not known before, even for the classical $O(n \log n)$ bound for mutation probability $1/n$. For the result with leading constant made explicit, we obtain the bound of $1.39en\ln(n)(1 + o(1))$ with probability $1 - o(1)$.

4 Minimum Spanning Trees and Minimum Weight Bases in Matroids

Neumann and Wegener [23] show that the (1+1) EA finds a minimum spanning tree in an integer-weighted undirected graph $G = (V, E)$ in expected time $O(|E|^2 \log |E|)$.

Their key argument can be phrased in the language of drift analysis. A solution of this problem is described by a bit string of length $m := |E|$, stating which edges form the tree. For such a bit string, let $f(x)$ denote the sum of the edge weights of those edges that are in the solutions. Let f_{opt} denote the weight of an optimal solution. Let $\Phi(x) := f(x) - f_{\text{opt}}$. What Neumann and Wegener show in their analysis is that for all solutions x that already form a tree, we have $E(f(x_{\text{new}})) - f_{\text{opt}} \leq (1 - 1/m^2)(f(x) - f_{\text{opt}})$. Here, x_{new} as above denotes the outcome of one mutation-selection step performed with the indiviual x. By construction of the algorithm, this is always a spanning tree provided x was.

This shows that Φ is a feasible m^2-drift function (when restricted to solutions forming a spanning tree). Since it is easy to see from [23] that the EA finds some spanning tree (not necessarily a minimum one) with high probability in time $O(m \log m)$, Theorem 1 now asserts that, with probability $1 - m^{-c}$, the EA finds a minimum spanning tree in $O(m^2 \log(nw_{\max}))$ iterations, where as usual $n := |V|$ denotes the number of vertices of the graph.

Theorem 2. *The (1+1) EA for the minimum spanning tree problem investigated in [23] has an optimization time of $O(m^2 \log(nw_{\max}))$ with probability $1 - m^{-c}$, where c can be any constant.*

Very similar, and therefore with no further details given, one can extend the bound [25] of $O(m^2 \log(mw_{\max}))$ for the expected optimization time of the (1+1) EA searching for a minimum weight basis in a weighted matroid of cardinality m to a bound holding with probability $1 - m^{-c}$.

Theorem 3. *The (1+1) EA for the minimum weight basis problem in weighted matroids investigated in [25] has an optimization time of $O(m^2 \log(mw_{\max}))$ with probability $1 - m^{-c}$, where m is the size of the matroid, $w_{\max}$ the maximum weight and c can be any constant.*

5 Shortest Paths

Let $G = (V, E)$ be a directed graph, $w : E \to \mathbb{Z}_{>0}$ a positive integral weight function and $s \in V$ be a distinguished vertex called *source*. The single-source shortest path problem is to compute a shortest path tree for s, that is, a directed tree rooted as s such that the unique path from s to any other vertex of G is a shortest path from s to that vertex in G. Note that this is the classical shortest path problem because for general graphs there is no algorithm known for only computing a shortest path from s to one given vertex that is better than solving this single-source all destinations problem.

This problem is particularly interesting for the theory of evolutionary computation community because two different fitness functions are a natural choice. The *single-criterion* one is to simply add the distances of all vertices from s in the current solution. The multi-criteria one is to regard the vector formed from these distances, and accept a new solution only if it is better in all components of this vector, that is, only if all vertices are at least as close to s in the new solutions as in the old one.

The multi-criteria formulation is easier to analyze. Scharnow, Tinnefeld and Wegener [26] show that the optimization time of the (1+1) EA (with a natural representation, which we do not describe here) is of order n^3 in expectation. If we denote by ℓ the minimum height of a shortest path tree, then this was improved to a bound of $O(n^2 \max\{\log(n), \ell\})$ in [4], which in addition holds with high probability.

For the single-criterion formulation, no reasonable results can be obtained if non-connected vertices contribute ∞ to the fitness. If, what makes more sense from the view-point of implementation, non-connected vertices 'only' contribute a large penalty term, say $nw_{\max}$, then Baswana et al. [1] showed an expected runtime of $O(n^3 \log(nw_{\max}))$. By using a slightly different mutation operator, this was improved to $O(nm \log(nw_{\max}))$ by Doerr and Johannsen [6]. Both proofs rely on a drift argument. In [1], a gap function was defined, which in our language simply is a feasible n^3-drift function. In [6], it was shown that the difference to the optimum fitness is a feasible nm-drift function. Since $n^2 w_{\max}$ in all cases in an upper bound for the value of either of these drift functions, Theorem 1 yields the following.

Theorem 4. *With probability $1 - n^{-c}$, the EA given in [1] finds a shortest path tree in time $O(n^3 \log(nw_{\max}))$. With probability $1 - n^{-c}$, the EA given in [6] finds a shortest path tree in time $O(nm \log(nw_{\max}))$.*

6 Eulerian Cycles

Let $G = (V, E)$ be an undirected graph. A *Eulerian cycle* in G is a cyclic walk (described via the sequence of edges it traverses) that contains each edge exactly once. It is well-known that G contains a Eulerian cycle if and only if each vertex has even degree, that is, it is incident with an even number of edges [12]. It is less trivial to actually compute a Eulerian cycle in such a graph. Hierholzer [20] describes an algorithm with run-time $O(mn)$.

For the theory of evolutionary computation, the Eulerian cycle problem is interesting because it can serve as a test problem suitable to analyse what are good representations for the individuals (which are permutations of the edges set) and what are good mutation operators to build upon them. This resulted in a series of papers [21,2,9,5] on this problem. All of them show bounds on the expected run-time of different implementations of the (1+1) EA for the Eulerian cycle problem.

The currently fastest solution was given in [5]. It represents the sought-after Eulerian cycle via a perfect matching in the adjacency lists of each vertex. If neighbors x, y of a vertex v form such a matching edge, this indicates that the edges $\{x, v\}$ and $\{v, y\}$ are adjacent in the Eulerian cycle. Clearly, an arbitrary matching in each adjacency list does not necessarily encode a Eulerian cycle, but only a partition of the edges of G into edge-disjoint cycles (cycle cover).

An elementary mutation for this representation chooses a vertex-edge incidence uniformly at random, e.g., by choosing uniformly at random an edge $e \in E$ and then a vertex v from e). Denote by x the other vertex of e. The elementary mutation now chooses a second vertex y from the adjacency list of v, makes $\{x, y\}$ a matching edge, and (if needed), makes the two former partners of x and y a matching edge as well. In other words, it adds an edge uniformly at random to the matching of v's adjacency list and repairs this by forming a perfect matching again in the obvious way.

A full mutation step consists of performing $S + 1$ such elementary mutations, where S is chosen according to a Poisson distribution with parameter $\lambda = 1$.

For this (1+1) EA, an expected optimization time of $O(m \log m)$ was shown in [5]. A key argument in the proof is that if the current solution is a cycle cover consisting of k cycles, then a single application of the mutation operator with probability at least $(k - 1)/(2m)$ unites two cycles to one.

From this, we easily deduce that the bound of [5] also holds with high probability. To this aim, let $\Phi(x)$ denote the number of cycles in the cycle cover represented by the individual x, minus one. Hence $\Phi(x) = 0$ is equivalent to having only one cycle, which then is a Eulerian cycle. Clearly, $\Phi(x) \leq m$ for all x. The key arguments described in the previous paragraph shows that Φ is an m-drift function. Hence the EA has an optimization time of $O(m \log m)$ with high probability.

7 Conclusion

By reproving a classical drift theorem, we obtain that the classical $O(n \log n)$ bound for expected optimization time of the (1+1) EA on linear functions also holds with high probability. The same argument allows us to extend a number of other classical bounds stemming from drift or "expected multiplicative weight decrease" arguments to also hold with high probability, instead of only in expectation. We expect that this version of the drift theorem will see more applications in the theory of evolutionary algorithms.

Acknowledgements

The authors would like to thank Daniel Johannsen for several useful comments.

References

1. Baswana, S., Biswas, S., Doerr, B., Friedrich, T., Kurur, P.P., Neumann, F.: Computing single source shortest paths using single-objective fitness. In: Proceedings of FOGA 2009, pp. 59–66. ACM, New York (2009)
2. Doerr, B., Hebbinghaus, N., Neumann, F.: Speeding up evolutionary algorithms through restricted mutation operators. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 978–987. Springer, Heidelberg (2006)
3. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. In: Proceedings of PPSN XI. Springer, Heidelberg (to appear, 2010)
4. Doerr, B., Happ, E., Klein, C.: A tight bound for the (1+1)-EA on the single-source shortest path problem. In: Proceedings of CEC 2007, pp. 1890–1895. IEEE, Los Alamitos (2007)
5. Doerr, B., Johannsen, D.: Adjacency list matchings — an ideal genotype for cycle covers. In: Proceedings of GECCO 2007, pp. 1203–1210. ACM, New York (2007)
6. Doerr, B., Johannsen, D.: Edge-based representation beats vertex-based representation in shortest path problems. In: Proceedings of GECCO 2010. ACM, New York (to appear, 2010)
7. Doerr, B., Johannsen, D., Winzen, C.: Drift analysis and linear functions revisited. In: Proceedings of CEC 2010. IEEE, Los Alamitos (to appear, 2010)
8. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. In: Proceedings of GECCO 2010. ACM, New York (to appear, 2010)
9. Doerr, B., Klein, C., Storch, T.: Faster evolutionary algorithms by superior graph representation. In: Proceedings of FOCI 2007, pp. 245–250. IEEE, Los Alamitos (2007)
10. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
11. Dyer, M., Greenhill, C.: Random walks on combinatorial objects. In: Surveys in Combinatorics 1999, pp. 101–136. University Press (1999)
12. Euler, L.: Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae 8, 128–140 (1741)
13. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
14. Giel, O., Lehre, P.K.: On the effect of populations in evolutionary multi-objective optimization. In: Proceedings of GECCO 2006, pp. 651–658. ACM, New York (2006)
15. Grimmett, G.R., Stirzaker, D.R.: Probability and random processes, 2nd edn. The Clarendon Press Oxford University Press, New York (1992)
16. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability 13, 502–525 (1982)
17. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Proceedings of GECCO 2008, pp. 953–960. ACM, New York (2008)

18. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence 127, 57–85 (2001)
19. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3, 21–35 (2004)
20. Hierholzer, C.: Über die Möglichkeit, einen Linenzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen 6, 30–32 (1873)
21. Neumann, F.: Expected runtimes of evolutionary algorithms for the Eulerian cycle problem. In: Proceedings of CEC 2004, pp. 904–910. IEEE, Los Alamitos (2004)
22. Neumann, F., Oliveto, P.S., Witt, C.: Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In: Proceedings of GECCO 2009, pp. 835–842. ACM, New York (2009)
23. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theoretical Compututer Science 378, 32–40 (2007)
24. Oliveto, P.S., Witt, C.: Simplified drift analysis for proving lower bounds in evolutionary computation. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 82–91. Springer, Heidelberg (2008)
25. Reichel, J., Skutella, M.: Evolutionary algorithms and matroid optimization problems. In: Proceedings of GECCO 2007, pp. 947–954. ACM, New York (2007)
26. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms 4, 349–366 (2004)

More Effective Crossover Operators for the All-Pairs Shortest Path Problem

Benjamin Doerr[1], Daniel Johannsen[1], Timo Kötzing[1], Frank Neumann[1], and Madeleine Theile[2]

[1] Max-Planck-Institut für Informatik, Campus E 1 4, 66123 Saarbrücken, Germany
[2] Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany

Abstract. The all-pairs shortest path problem is the first non-artificial problem for which it was shown that adding crossover can significantly speed up a mutation-only evolutionary algorithm. Recently, the analysis of this algorithm was refined and it was shown to have an expected optimization time of $\Theta(n^{3.25}(\log n)^{0.25})$.

In this work, we study two variants of the algorithm. These are based on two central concepts in recombination, *repair mechanisms* and *parent selection*. We show that repairing infeasible offspring leads to an improved expected optimization time of $O(n^{3.2}(\log n)^{0.2})$. Furthermore, we prove that choosing parents that guarantee feasible offspring results in an optimization time of $O(n^3 \log n)$.

1 Introduction

One of the important issues when designing successful evolutionary algorithms is to choose a suitable representation of possible solutions together with good variation operators. Different representations and variation operators have been discussed for a wide range of combinatorial optimization problems (see e. g. [18,11,19]). Often, variation operators (such as crossover or mutation) are designed to produce feasible offsprings. For mutation this is easy to achieve, as a mutation operator usually only applies a small number of local changes to a given feasible solution.

However, the design of crossover operators, producing from two feasible solutions a new feasible one, is usually more complicated (see e. g. [14] for different crossover operators for the traveling salesman problem). Whenever a crossover operator produces an infeasible solution, one option is to discard it. However, this typically does not lead to efficient methods, as time is wasted on producing infeasible solutions and evaluating them. To deal with this situation, one can use repair mechanisms, which produce from an infeasible solution a feasible one based on properties of both parents [22]. Another way of dealing with the problem of infeasible solutions is to use specific selection methods and/or more problem specific crossover operators that are likely to produce promising solutions [2,15].

The goal of this paper is to point out the effect of repair mechanisms and parent selection for crossover on the runtime of evolutionary algorithms in combinatorial optimization. Analyzing the runtime behavior of evolutionary algorithms has become a major part in their theoretical analysis. Based on results for different kinds of pseudo-Boolean functions [6,10], results have been obtained for different kinds of combinatorial optimization problems. Starting with some results for classical combinatorial optimization problems that are solvable in polynomial time such as the computation of minimum spanning trees [17] or maximum matchings [8], different results have been obtained for NP-hard problems [16,7,12,23]. One cannot expect to beat the best known algorithms if the problem under consideration can be solved in polynomial time. With such studies we want to gain new insights on how evolutionary algorithms behave on natural optimization problems and give insights into the important modules that make such algorithms successful.

We carry out theoretical studies on evolutionary algorithms for the computation of shortest paths. Computing shortest paths is one of the basic problems in computer science and has already been considered in various theoretical studies of evolutionary algorithms. There are different results for the single-source shortest path (SSSP) problem [1,21,3].

We investigate the all-pairs shortest path (APSP) problem which is a generalization of the SSSP problem. Given a strongly connected directed graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ and a weight function $w : E \to \mathbb{R}$ that assigns weights to the edges. We distinguish between the *weight* of a path (the sum of the weight of all its edges) and its *length* (the *number* of edges in the path). The task is to compute from each vertex $v \in V$ a weight-shortest path to every other vertex $u \in V \setminus \{v\}$. Throughout this paper, we assume that G does not contain cycles of negative weight. The APSP problem can be solved by the Floyd-Warshall algorithm; using appropriate data structures, APSP can be computed in time $O(nm + n^2 \log n)$ (see, e. g. [13]). Our aim is to study how general purpose algorithms can deal with the APSP problem. In particular, we want to examine the usefulness of crossover operators in evolutionary computation.

We take the APSP problem as a prominent example to show in a rigorous way how different crossover operators influence the runtime of evolutionary algorithms. Recently, it has been shown that the use of crossover operators provably leads to better evolutionary algorithms than evolutionary algorithms that are just based on mutation [4,5]: The runtime for the mutation-and-crossover approach is $\Theta(n^{3.25}(\log n)^{0.25})$, which is better than the expected optimization time of $\Theta(n^4)$ of the algorithm just using mutation. In addition, [9] studied the runtime behavior of ant colony optimization for this problem and proved an upper bound of $O(n^3(\log n)^3)$. However, we will see that the evolutionary approach examined in this paper solves the APSP problem in expected optimization time $O(n^3 \log n)$.

In the next section, Section 2, we introduce the algorithms that are subject to our analyses. In Section 3, we show how repair mechanisms can speed up the

```
1  $\mathcal{P} = \{P_{u,v} = (u,v) \mid (u,v) \in E\}$;
2  while true do
3      Choose $r \in [0,1]$ uniformly at random;
4      if $r \leq p_c$ then
5          choose two individuals $P_{x,y}$ and $P_{x',y'}$ from $\mathcal{P}$ u. a. r.; perform crossover
           on $P_{x,y}$ and $P_{x',y'}$ to obtain an individual $P'_{s,t}$ ;
6      else
7          choose one individual $P_{x,y}$ uniformly at random from $\mathcal{P}$ and mutate
           $P_{x,y}$ to obtain an individual $P'_{s,t}$;
8      if $P'_{s,t}$ is a path from $s$ to $t$ then
9          if there is no individual $P_{s,t} \in \mathcal{P}$ then $\mathcal{P} = \mathcal{P} \cup \{P'_{s,t}\}$;
10         else if $w(P'_{s,t}) \leq w(P_{s,t})$ then $\mathcal{P} = (\mathcal{P} \cup \{P'_{s,t}\}) \setminus \{P_{s,t}\}$;
```

Algorithm 1: Steady State GA$_{\text{APSP}}$

optimization process to $O(n^{3.2}(\log n)^{0.2})$. In Section 4, we analyze a crossover selecting two matching individuals, and show that this leads to an optimization time of $O(n^3 \log n)$.

In order to meet space constraints, some proofs had to be left out.

2 Algorithms

For the APSP problem we examine the population-based approach introduced in [4], where each individual in the population is a path. Our goal is to evolve an initial population consisting of a set of paths into a population which contains, for each pair of vertices (u, v) with $u \neq v$, a shortest path from u to v.

We investigate two evolutionary algorithms for the APSP problem that differ on how they apply crossover. The algorithms start with a population $\mathcal{P} := \{P_{u,v} = (u,v)|(u,v) \in E\}$ of size $|E|$, containing all paths corresponding to the edges of the given graph G. The variation operators produce in each iteration one single offspring.

Our algorithm, called Steady State GA$_{\text{APSP}}$ (see Algorithm 1), decides in each iteration whether the offspring is produced by crossover or mutation. With probability p_c a crossover operator is applied to two randomly chosen individuals of $\mathcal{P}$ or otherwise (with probability $1 - p_c$) mutation is used to produce the offspring. To make sure that both operators, mutation and crossover, are used we require $p_c \notin \{0, 1\}$. For all investigations in this paper, we assume that p_c is chosen as an arbitrary constant, i. e. $p_c \in]0, 1[$.

The mutation operator takes an individual $P_{x,y}$ from the population and applies sequentially $S + 1$ local operations. Here, S is a parameter that is chosen according to the Poisson distribution with parameter $\lambda = 1$. In a local operation, the current path is either lengthened or shortened by a single edge. Assume that the current individual represents a path $P_{x,y} = (x = v_0, v_1, \ldots v_{\ell-1}, y = v_\ell)$ from x to y consisting of ℓ edges, and denote by $E^-(v)$ and $E^+(v)$ the set of incoming and outgoing edges of a vertex v in G, respectively. Then an edge

$e = (u, v) \in E^-(x) \cup E^+(y) \cup \{(x, v_1), (v_{\ell-1}, y)\}$ is chosen uniformly at random. If $e \in \{(x, v_1), (v_{\ell-1}, y)\}$, the edge is removed. This means that either the first edge or the last edge in the path is removed leading to an individual $P'_{v_1,y}$ or $P'_{x,v_{\ell-1}}$ consisting of $\ell - 1$ edges. If $e \in (E^-(x) \cup E^+(y)) \setminus \{(x, v_1), (v_{\ell-1}, y)\}$, the edge is added and the path is lengthened. Here, a new individual $P'_{u,y}$ or $P'_{x,v}$ is produced that contains $\ell + 1$ edges. Note that a local operation applied to a valid path always leads to a new valid solution which implies that the mutation operator only constructs solutions which are paths.

Crossover takes two individuals and combines them into a valid path if the end vertex of $P_{x,y}$ and the start vertex of $P_{x',y'}$ match. Choosing both individuals uniformly at random from $\mathcal{P}$, as it was done in [4,5], often does not lead to a recombined offspring that represents a path in the given graph. In the next section, we discuss how repair mechanisms can lead to more efficient evolutionary algorithms. Later on, we discuss how selection methods that select promising pairs of individuals for crossover lead to evolutionary algorithms that are almost as fast as classical algorithms for the APSP problem.

The selection operator only accepts individuals that are paths in the graph. In addition, it ensures diversity with respect to the different pairs of vertices. For this reason, each individual $P_{u,v}$ is indexed by the start vertex u and the end vertex v. In the selection step an offspring is only compared to an individual of the current population that has the same start and end vertex. It is ensured that, for each pair of vertices (u, v) with $u \neq v$, at most one individual $P_{u,v}$ is contained in the population. This implies that the population size of our algorithms is always at most $n(n - 1)$.

For our theoretical investigations, we measure the optimization time of the algorithm by the number of fitness evaluations until an optimal population has been reached for the first time. A population is optimal if it represents, for each pair of vertices, a shortest path.

Finally, the term w. h. p. (with high probability) denotes a result that holds with probability at least $(1 - O(n^{-c}))$ for some $c > 0$ independent of n.

3 Crossover with Repair

In this section, we present a simple way to increase the success probability of the crossover operator used in previous work. This result, as we shall prove rigorously, is an improved optimization time of $O\big(n^{3.2}(\log n)^{0.2}\big)$.

The main reason why previous crossover operators for the APSP problem have a relatively small success probability is the fact that very often the two parent individuals simply do not fit together. That is, the end-point of the first is not equal to the starting point of the second path. Since this is a rather obvious way of failing, one might think of simple solutions.

One natural way is the following. If end-point of first and starting point of second path are different, we try to bridge this gap by the (if existent, unique) path from one point to the other which is contained in our population. If the population does not contain such a bridging path, then the crossover operator still fails. This is what we shall call *crossover with repair*.

Definition 1 (Crossover with repair). *Let $\otimes_r$ denote the crossover operator with repair as follows (compare Figure 1).*

```
  Input: $P_{x,y} = (x, \dots, y)$ and $P_{x',y'} = (x', \dots, y')$ taken u. a. r. from $\mathcal{P}$
1 if $y = x'$ then
2     $P'_{s,t} = (s = x, \dots, y = x', \dots, t = y')$ merging $P_{x,y}$ and $P_{x',y'}$ at vertex $y$ ;
3 else
4     if there is a path $P_{y,x'}$ from $y$ to $x'$ in $\mathcal{P}$ then
5         $P'_{s,t} = (s = x, \dots, y, \dots, x', \dots, t = y')$ merging $P_{x,y}$, $P_{y,x'}$ and $P_{x',y'}$ at their common endpoints;
6     else
7         $\otimes_r$ fails and returns a dummy individual with fitness worse than all other possible individuals;
```

The individual $P_{y,x'}$ from Line 4 is called repair-path.

Note that this operation is inserted in Line 5 of Algorithm 1.

Assuming that all individuals used in the operation have a length of at most k, the application of the $\otimes_r$-operator produces a new individual of size at most $3k$. If all individuals considered for the operation are shortest paths (w. r. t. to their weight) then it is possible to produce shortest paths (w. r. t. to their weight) of length up to $3k$ due to the optimal substructure property of shortest paths. However, later on we will merely consider the case that if all optimal individuals of length k are present in the population the new individual will have a length of $\frac{3}{2}k$.

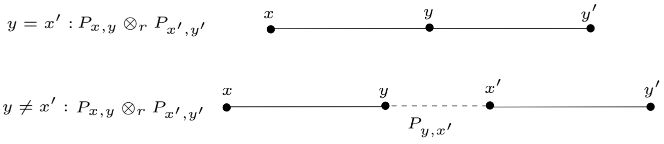

$y = x' : P_{x,y} \otimes_r P_{x',y'}$
x
y
y'
$y \neq x' : P_{x,y} \otimes_r P_{x',y'}$
x
y
x'
y'
$P_{y,x'}$

Fig. 1. Effect of the repair crossover applied to two paths $P_{x,y}$ and $P_{x',y'}$.

To analyze our crossover operator, we use the gap concept introduced in [5]. The key observation is that it suffices that crossover finds a path that sufficiently well approximates a sought-after path, because mutation is fast enough to fill the gaps.

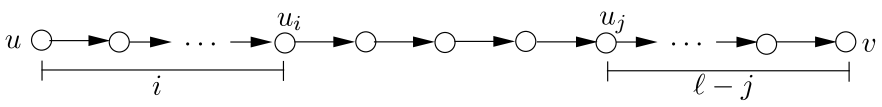


Fig. 2. Approximating path P_{u_i,u_j} with a gap of $g := i + \ell - j$

Definition 2 (Gap). *Consider a path* $P_{u,v} = (u = u_0, u_1, \ldots, u_\ell = v)$ *and an arbitrary sub-path* $P_{u_i,u_j} = (u_i, u_{i+1}, \ldots, u_j)$ *with* $0 \leq i \leq j \leq \ell$*, compare Figure 2. We call the integral value* $g := i + \ell - j$ *the* gap *of the path* P_{u_i,u_j} *(w. r. t. P). We also call* P_{u_i,u_j} *an approximating path of* $P_{u,v}$*. If* $P_{u,v}$ *is a shortest path between vertex u and v, we call* P_{u_i,u_j} *an approximating shortest path.*

For a simplification of the proofs we make use of the following definition which takes into account all pairs of vertices for which there is a shortest path containing at most k edges.

Definition 3. *Let* $G = (V, E)$ *be a graph and let* $k \in \mathbb{R}$*. We let* V_k^2 *be the set of all* $(u, v) \in V^2$ *with* $u \neq v$ *such that there exists a shortest path* $P_{u,v}$ *from u to v consisting of at most k edges.*

Note that we allow for a fractional k in order get rid of some delicate case distinctions later on.

The main statement of this section is captured by the following theorem. It shows that the use of the introduced repair mechanism leads provably to a better optimization time.

Theorem 1. *The Steady State* GA_{APSP} *with any constant rate* $0 < p_c < 1$ *using crossover with repair (Definition 1) has an optimization time of* $O\big(n^{3.2}(\log n)^{0.2}\big)$ *with high probability.*

For the proof of the theorem we need to analyze the success probability of the $\otimes_r$-operator, analyze the success probability of the mutation operator and describe the interplay between mutation and crossover.

We start by investigating the success probability for the crossover operator with repair. Using the crossover operator with repair gives us an additional factor of k for the success probability compared to the corresponding results in [5]. This is made precise in the following lemma.

Lemma 1 (Analysis of Crossover). *Let* $k > 1$ *and let* $\mathcal{P}$ *be an arbitrary but fixed population. Assume that* $\mathcal{P}$ *contains for each pair* $(u, v) \in V_k^2$ *a shortest path connecting them. Let* $\ell := \frac{3}{2}k$ *and* $g \leq \frac{k}{4}$*. Then the following holds.*

(1) A single step of the $\otimes_r$*-operator generates a shortest path from u to v with* $(u, v) \in V_\ell^2 \backslash V_k^2$ *with probability* $\Omega\Big(\frac{k^2}{n^4}\Big)$*.*
(2) Consider a gap g, then a single step of the $\otimes_r$*-operator generates an approximating shortest path from u to v with* $(u, v) \in V_\ell^2 \backslash V_k^2$ *with probability* $\Omega\Big(\frac{k^2 g^2}{n^4}\Big)$*.*

For the progress made by mutation, we use the following result given in [5].

Lemma 2 (Analysis of Mutation). *Let* $\ell > 0$ *and* $\mathcal{P}$ *be an arbitrary but fixed population. Assume that there is a shortest path for every* $(u, v) \in V_\ell^2$ *in* $\mathcal{P}$ *and the Steady State* GA_{APSP} *is allowed to use only mutations an no crossover-operations. Then the following holds.*

(1) The success probability to get an arbitrary but fixed shortest path in $V^2_{\ell+1} \setminus V^2_\ell$ is $\Omega\big(n^{-3}\big)$. Hence the expected waiting time for generating such a path is $O\big(n^3\big)$.

(2) Let $\lambda > 0$ and $c > 0$ with $c\lambda \geq 24 \ln n$. The Steady State GA$_{APSP}$ finds all shortest paths $(u, v) \in V^2_{\ell+c}$ with probability at least $n^{2-\frac{c\lambda}{8 \ln n}}$ in $O\big(c\lambda n^3\big)$ iterations.

Combining the analysis of the mutation operator and the crossover operator with repair, we obtain the following key lemma. It nicely shows the interplay between the two operators from the point on when we have shortest paths connecting all pairs in V^2_k for $k = \Omega\big((n \log n)^{0.2}\big)$.

Lemma 3. *Assume that the Steady State GA$_{APSP}$ uses mutation as well as the $\otimes_r$-operator with constant probability. Let $k \geq (n \log n)^{0.2}$ and $\Delta := \frac{(n \log n)^{0.2}}{k}$. Assume that there is a shortest path in the population for each pair $(u, v) \in V^2_k$. Let $\ell := \frac{3}{2}k$.*

(1) For all pairs $(u, v) \in V^2_\ell \setminus V^2_k$ an approximating shortest path (u_i, u_j) with gap at most $g \leq (n \log n)^{0.2}\Delta$ is found in $t = O\big(n^3 (n \log n)^{0.2}\Delta\big)$ iterations with high probability.

(2) Assume that for each pair $(u, v) \in V^2_\ell \setminus V^2_k$ there is an approximating shortest path (u_i, u_j) with a gap of at most $g \leq (n \log n)^{0.2}\Delta$. Then the algorithm finds all shortest paths with end-vertices in V^2_ℓ in $t = O\big(n^3 (n \log n)^{0.2}\Delta\big)$ iterations with high probability.

(3) Let $k = (n \log n)^{0.2}(1.5)^i$ for some $i \in \mathbb{N}_0$. Then with high probability, $O\big((1.5)^{-i} n^{3.2} (\log n)^{0.2}\big)$ iterations suffice to have all shortest paths of up to $1.5k$ edges in the population (where the hidden constant in the time does not depend on i).

Now we are in the position to prove our main theorem.

Proof (of Theorem 1). Both the crossover and the mutation operator have constant probability to be applied in an iteration, and neither can decrease the fitness of an individual. Hence we may occasionally only regard the effect of one of the two. Applying Lemma 2 with $c\lambda := (n \log n)^{0.2}$, we see that after $O\big(n^3 \cdot (n \log n)^{0.2}\big)$ iterations, with high probability all shortest paths having up to $\ell = (n \log n)^{0.2}$ edges are in the population.

We now repeatedly apply Lemma 3(3). In time $O\big(n^{3.2} \log_{1.5} n (1.5)^{-i}\big)$, with high probability we construct all shortest paths connecting vertices in $V^2_{(1.5)^{i+1}\ell}$ out of a population containing all shortest paths for vertices in $V^2_{(1.5)^i\ell}$. Hence the run-times form a geometric series and the less than $\log_{1.5} n$ such stages needed to find all shortest paths still take time $O\big(n^{3.2} (\log n)^{0.2}\big)$. Since each stage works fine with high probability, our algorithm finds all shortest paths with high probability as well.

The previous proof shows that the proposed repair mechanism leads provably to a better optimization time. In the next section, we will examine how selection for reproduction can influence the runtime of crossover-based evolutionary algorithms. We will see that this even leads to bounds on the optimization time that are close to the ones of problem-specific algorithms.

4 Feasible Parent Selection

The previous section has shown that a simple repair mechanism leads to an optimization time of $O\big(n^{3.2}(\log n)^{0.2}\big)$, which is already an improvement over the optimization time of $O\big(n^{3.25}(\log n)^{0.25}\big)$ for the Steady State GA$_{\text{APSP}}$ in [5]. Nevertheless, the crossover operator may still produce solutions that do not constitute paths. This is the case if the start vertex of the second individual does not match the end vertex of the first individual and there is no individual in $\mathcal{P}$ for repair.

In the following, we want to make sure that the crossover operator constructs feasible solutions, i. e. individuals that represent paths. This is done by restricting the parent selection for crossover to individuals that match with respect to their endpoints. We choose the two individuals for crossover in Line 5 of the Steady State GA$_{\text{APSP}}$ (Algorithm 1) using the feasible parent selection procedure given in Algorithm 2.

```
1 Choose $P_{x,y} \in \mathcal{P}$ uniformly at random.
2 Choose $P_{x',y'} \in \{P_{u,v} \mid P_{u,v} \in \mathcal{P} \wedge u = y \wedge v \neq x\}$ uniformly at random.
```

Algorithm 2: Feasible Parent Selection

It chooses the first individual $P_{x,y}$ uniformly at random from the population $\mathcal{P}$ and the second individual $P_{x',y'}$ uniformly at random among all individuals in $\mathcal{P}$ whose start vertex equals the end vertex y of $P_{x,y}$ but whose end vertex does not equal the start vertex of $P_{x,y}$. Afterwards, in Line 5, crossover is performed by concatenation. Note that, due to the selection of the two individuals, a path from x to y' is constructed, which implies that the crossover operator only constructs feasible solutions.

This selection operator for the two parents reduces the optimization time even further. The following theorem shows, that the optimization time of Steady State GA$_{\text{APSP}}$ comes close to the best known upper bound on the runtime of problem specific algorithms if Algorithm 3 is used to select the individuals for crossover.

Theorem 2. *The Steady State GA$_{APSP}$ with any constant rate $0 < p_c < 1$ using feasible parent selection (Algorithm 3) has an optimization time of $O\big(n^3 \log n\big)$ with high probability.*

Our proof of this theorem uses an analysis similar to that of Theorem 1, but without considering gaps.

5 Conclusions

We have shown how the use of repair mechanism or appropriate selection strategies can speed up crossover-based evolutionary algorithms in the special case of the all-pairs shortest path problem. Understanding the usefulness of crossover in evolutionary computation in a rigorous way for other problems remains a challenging task for future research.

References

1. Baswana, S., Biswas, S., Doerr, B., Friedrich, T., Kurur, P.P., Neumann, F.: Computing single source shortest paths using single-objective fitness functions. In: Proceedings of the 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009), Orlando, USA, pp. 59–66. ACM Press, New York (2009)
2. Cherba, D.M., Punch, W.F.: Crossover gene selection by spatial location. In: Cattolico, M. (ed.) GECCO, pp. 1111–1116. ACM, New York (2006)
3. Doerr, B., Happ, E., Klein, C.: A tight analysis of the (1 + 1)-EA for the single source shortest path problem. In: IEEE Congress on Evolutionary Computation, pp. 1890–1895. IEEE, Los Alamitos (2007)
4. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. In: Ryan, C., Keijzer, M. (eds.) GECCO, pp. 539–546. ACM, New York (2008)
5. Doerr, B., Theile, M.: Improved analysis methods for crossover-based algorithms. In: Rothlauf [20], pp. 247–254
6. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. 276, 51–81 (2002)
7. Friedrich, T., Hebbinghaus, N., Neumann, F., He, J., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Lipson, H. (ed.) GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 797–804. ACM, New York (2007); Journal version appears in Evolutionary Computation
8. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
9. Horoba, C., Sudholt, D.: Running time analysis of ACO systems for shortest path problems. In: Stützle, T., Birattari, M., Hoos, H.H. (eds.) SLS 2009. LNCS, vol. 5752, pp. 76–91. Springer, Heidelberg (2009)
10. Jansen, T., Wegener, I.: Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans. Evolutionary Computation 5(6), 589–599 (2001)
11. Knowles, J.D., Corne, D.: A comparison of encodings and algorithms for multiobjective spanning tree problems. In: Proceedings of the Congress on Evolutionary Computation 2001, pp. 544–551. IEEE Press, Los Alamitos (2001)
12. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. In: Rothlauf [20], pp. 293–300
13. Mehlhorn, K., Sanders, P.: Algorithms and Data Structures: The Basic Toolbox. Springer, Heidelberg (2008)
14. Michalewicz, Z., Fogel, D.B.: How to solve it: Modern heuristics. Springer, Berlin (2004)

15. Michalewicz, Z., Nazhiyath, G., Michalewicz, M.: A note on usefulness of geometrical crossover for numerical optimization problems. In: Evolutionary Programming, pp. 305–312 (1996)
16. Neumann, F., Reichel, J.: Approximating minimum multicuts by evolutionary multi-objective algorithms. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 72–81. Springer, Heidelberg (2008)
17. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theoretical Computer Science 378(1), 32–40 (2007)
18. Raidl, G.R., Julstrom, B.A.: Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans. Evolutionary Computation 7(3), 225–239 (2003)
19. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Springer, Heidelberg (2003)
20. Rothlauf, F. (ed.): Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12. ACM, New York (2009)
21. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms, 349–366 (2004)
22. Walters, T.: Repair and brood selection in the traveling salesman problem. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 813–822. Springer, Heidelberg (1998)
23. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)

Comparison-Based Adaptive Strategy Selection with Bandits in Differential Evolution

Álvaro Fialho[1], Raymond Ros[2], Marc Schoenauer[1,2], and Michèle Sebag[1,2]

[1] Microsoft Research - INRIA Joint Centre, Orsay, France
[2] TAO team, INRIA Saclay - Île-de-France & LRI (UMR CNRS 8623), Orsay, France
FirstName.LastName@inria.fr

Abstract. Differential Evolution is a popular powerful optimization algorithm for continuous problems. Part of its efficiency comes from the availability of several mutation strategies that can (and must) be chosen in a problem-dependent way. However, such flexibility also makes DE difficult to be automatically used in a new context. *F-AUC-Bandit* is a comparison-based Adaptive Operator Selection method that has been proposed in the GA framework. It is used here for the on-line control of DE mutation strategy, thus preserving DE invariance w.r.t. monotonous transformations of the objective function. The approach is comparatively assessed on the BBOB test suite, demonstrating significant improvement on baseline and other *Adaptive Strategy Selection* approaches, while presenting a very low sensitivity to hyper-parameter setting.

1 Introduction

Differential Evolution (DE) uses the weighted difference between two or more parent solutions to generate offspring [19]. DE has been successfully applied to many real-world applications[17] thanks to its simplicity and high flexibility. This flexibility is mostly provided by the number of different mutation strategies that can be used for the offspring generation. However, this flexibility is also a limitation to its wide dissemination, as the user needs to choose the mutation strategy for every new problem — and the efficiency of the algorithm is highly sensitive to this choice.

Such choice is usually done by following the user's intuition, or by using an off-line tuning procedure aimed at identifying the best strategy for the problem at hand. Besides being computationally expensive, off-line tuning however generally delivers sub-optimal performances, as the appropriate strategy depends on the stage of the optimization process: exploration-like strategies should be more frequently used in the early stages while priority should be given to exploitation when approaching the optimum.

For this reason, the present paper focuses on on-line tuning, aimed at selecting the strategy for the next offspring generation on the basis of the current results of the strategies (*i.e.*, the quality of the recent offspring they generated). Such on-line selection, referred to as *Adaptive Strategy Selection* (*AdapSS*), is similar in spirit to *Adaptive Operator Selection* (*AOS*) in the GA framework. A brief review of some *AdapSS*/*AOS* techniques is presented in Section 2, focusing on the ones used in the experiments and compared to the proposed approach.

Another important feature of DE (and many other bio-inspired algorithms) is that it is comparison-based, and thus invariant to monotonous transformations of the fitness function. Such invariance property significantly increases the robustness of the approach; in particular it implies that no fitness scaling is ever required when dealing with a new application. Unfortunately, this invariance is generally lost when adding mechanisms like *AOS*. *Fitness-based Area-Under-Curve - Bandit* (*F-AUC-Bandit*) is a recently introduced *AOS* that only uses the ranks of the most recent offspring to assess the strategy credit, combining an *Area Under the ROC Curve* (*AUC*) measure [3] and a Multi-Armed Bandit algorithm [9]. It is here ported to the *AdapSS* context, preserving the invariance properties of DE, while implementing on-line operator choice. For the sake of completeness, *F-AUC-Bandit* is briefly described in Section 3.

The *F-AUC-Bandit AdapSS* approach is experimentally validated on the BBOB-2010 noiseless benchmarking suite [11]. Extensive comparative results are reported in Section 4, considering baseline strategies (uni-strategies and random selection) as well as three adaptive schemes: *Adaptive Pursuit* (*AP*) [20] and *Dynamic Multi-Armed Bandit* (*DMAB*) [4], both being here assessed for the first time in the continuous domain, and *PM-AdapSS-DE*, another technique recently proposed and analyzed in the context of *Adaptive Strategy Selection* within DE [10]. Finally, Section 5 concludes the paper, summarizing the presented results and pointing out possible directions for future work.

2 Adaptive Strategy Selection

Adaptive Strategy Selection performs on-line selection of the mutation strategy for the generation of each new offspring, based on the recent known performance of each of the available strategies. The *AdapSS* paradigm requires two ingredients: the *Credit Assignment* scheme assesses the performance of a given strategy, translating the fitness of the newly generated offspring into a numerical credit; the *Strategy Selection* method rules how to choose a given strategy among all available ones, based on their (continuously updated) credit. This paradigm closely parallels that of *Adaptive Operator Selection* in the Genetic Algorithms community [20,4]. A brief review of some approaches previously proposed on both communities is presented in this section, without aiming at exhaustivity.

2.1 Credit Assignment

Different approaches for *Credit Assignment* have been proposed, differing mainly on three aspects: (i) how the impact of the strategy application should be measured; (ii) how to assign credit based on these impact assessments; and finally, (iii) to which strategy the credit should be assigned to.

The most common impact measure of a strategy application is the fitness improvement brought by the newly generated offspring, when compared to its parent [7], to the current median [14] or to the best individual [5] in the population. In [10], a relative fitness improvement is used, taking into account the difference of the fitness of the offspring with that of its parent, and normalizing it by the ratio between its fitness and the best one in the current population.

These rewards are then transformed into a credit to be assigned to the strategy, thus updating its empirical quality estimate. This quality estimate is in turn used by the *Strategy Selection* for the selection of the next strategy. Such credit might be the instantaneous reward, *i.e.*, received after the last application; the average of the rewards received over a few recent applications; or the extreme (or maximum) reward recently received by the strategy [6]. The number of recent applications considered for the latter two is usually a user-defined parameter, referred to as W (size of the sliding window) in the following.

Finally, some authors [5,14] have proposed to assign credit to the strategies that were used to generate the ancestors of the current individual, by means of a bucket brigade scheme. However, most works, including the present one, only assign the reward to the strategy used to generate the newborn offspring.

2.2 Strategy Selection

The *Strategy Selection* schemes select the next strategy between the available ones based on their known empirical quality, which is updated by the *Credit Assignment* mechanism after each application. The main difference between the proposed methods lies in how they use such empirical estimates to select the strategy to be applied. Two types of schemes are distinguished.

The probability-based methods *Probability Matching* (*PM*) and *Adaptive Pursuit* (*AP*) [20] calculate an application probability for each strategy, and use roulette wheel to select the next strategy. Both methods set a lower bound on the probabilities to preserve some exploration. *PM* sets each probability proportionally to the empirical quality of the strategy; *AP* implements a winner-takes-all scheme, quickly increasing the probability of the current best strategy.

The bandit-based methods *Multi-Armed Bandit* (*MAB*) and *Dynamic MAB* (*DMAB*) [4,7] deterministically choose the strategy to be applied based on (a variant of) the *Upper Confidence Bound* (*UCB*) algorithm [1]:

$$\text{Select } \arg\max_i \left(\hat{q}_{i,t} + C\sqrt{\frac{2\log\sum_k n_{k,t}}{n_{i,t}}} \right) \tag{1}$$

where $\hat{q}_{i,t}$ denotes the empirical quality of the i-th option (exploitation term), $n_{i,t}$ the number of times it has been selected so far (the right term corresponding to the exploration term), and C is a user-defined constant (hyper-parameter) controlling the balance between Exploration and Exploitation.

Bandit algorithms have been proven to optimally solve the Exploration vs. Exploitation (EvE) dilemma, albeit in a stationary context. While an *AdapSS* method indeed faces an EvE dilemma (the algorithm should exploit as much as possible the current best mutation strategy, while maintaining some exploration of the other strategies in case one of them becomes more efficient at a later stage of the optimization), it is a dynamic one: the performances of the strategies vary as the search advances. In order to better cope with these dynamics, the *DMAB* proposes the restart of the *MAB* process whenever a change in the reward distribution is detected by means of the Page-Hinkley statistical test [16].

3 Comparison-Based Adaptive Strategy Selection: Fitness-Based AUC Bandit

Although alleviating the user from the need of selecting which strategies should be applied to the problem at hand, and doing so in an on-line manner, each of the presented *Strategy Selection* methods involves some hyper-parameters that need to be tuned as well. Furthermore, the common use of fitness improvements as reward makes these hyper-parameters highly problem-dependent, as the range of fitness values varies widely from one problem to another – as well as in the course of an optimization run. A natural way to improve the *Strategy Selection* robustness w.r.t. fitness scaling is to preserve the comparison-based invariance property, that DE shares with many other bio-inspired optimization algorithms (ES, tournament-based GAs, PSO). For this reason, the paper focuses on the *Fitness-based Area-Under-Curve - Bandit* (*F-AUC-Bandit*), a fully comparison-based *AdapSS* recently proposed in the context of GAs [9].

The *Area Under the ROC Curve* (*AUC*) is a criterion originally used in Signal Processing and later adopted in Machine Learning to compare binary classifiers, with the property of being robust with respect to class imbalance [3]. The *Receiving Operator Curve* (*ROC*) depicts how the true positive rate varies with the false positive rate. This indicator is adapted to the comparison-based assessment of strategies/operators as follows. Let us consider the list of the offspring generated in a given time window, and let the list be ranked after the offspring fitness. The *Receiving Operator Curve* associated to a given strategy s is drawn by scanning the ordered list, starting from the origin: a vertical segment is drawn when the current offspring has been generated by s, a horizontal segment is drawn otherwise, and a diagonal one is drawn in case of ties (Fig. 1, reproduced from [9]). The credit associated to strategy s finally is the area under this curve.

While all rank positions have same weight in the above *AUC* calculation (as in Fig. 1 for the sake of clarity), *i.e.*, all horizontal and vertical segments have same length, it makes sense to give more weight to the top ranked offspring. Algorithmically, a decay factor D is used as follows. Let W denote the size of the time window storing the list of recently generated offspring, let r be a rank position, the length of the current fragment (its weight in the *AUC* calculation) is set to $D^r(W - r)$, with $D \in]0, 1]$. The smaller D, the faster the decay, *i.e.*, the more skewed the credit assignment is.

This *Credit Assignment* scheme is coupled with a bandit-based *Strategy Selection* using Equation (1), where $\hat{q}_{i,t}$ is the *AUC* credit and $n_{i,t}$ is the number of times the i-th strategy has been applied in the current window. Note that, in the original *MAB* algorithm for *AdapSS* [4], $\hat{q}$ is defined as the average over all recent credits assigned to the given strategy. However, the *AUC* indicator also

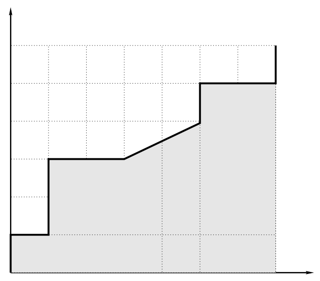


Fig. 1. Computing the AUC reward (reproduced from [9]) associated to strategy 1. Only two operators are considered; the list of the generated offspring is sorted by fitness value; replacing each offspring by the index of the generating strategy gives (1 2 1 1 2 2 [2 2 1] 1 2 2 1), where [2 2 1] stands for three offspring with same fitness values, resulting in the diagonal line between points (3 3) and (5 4).

provides an empirical statistics over the last W offspring, reflecting the up-to-date performance of the given strategy w.r.t. the others.; it is thus directly used as the exploitation term in the MAB formula.

4 Experimental Results

This section reports on the evaluation of the *F-AUC-Bandit* approach coupled with standard DE, comparatively to single-strategies and other *AdapSS* schemes.

4.1 Experimental Setting

The goal of the experiments is to assess the comparative performances of the *AdapSS* schemes when coupled with standard *Differential Evolution* [19], the only difference regarding the strategy selection. DE is governed by three parameters NP, F and CR, respectively denoting the population size, the mutation scaling factor and the crossover rate. It must be emphasized that our goal is not to compete with state-of-the-art continuous optimizers; for this reason, no specific effort was put on tuning the DE parameters depending on the problem at hand. Population size NP is set to $10 \times d$, where d denotes the dimension of the search space; mutation scaling factor F is set to .5; crossover rate CR is set to 1, enforcing DE invariance w.r.t. rotation and stressing the impact of the mutation strategy. Along the same lines, four mutation strategies were chosen, retaining the same as in [10] for the sake of comparative evaluation:

1. "rand/1": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \big(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\big)$
2. "rand/2": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \big(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\big) + F \cdot \big(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\big)$
3. "rand-to-best/2": $\mathbf{v}_i = \mathbf{x}_{r_1} + F \cdot \big(\mathbf{x}_{best} - \mathbf{x}_{r_1}\big) + F \cdot \big(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\big) + F \cdot \big(\mathbf{x}_{r_4} - \mathbf{x}_{r_5}\big)$
4. "current-to-rand/1": $\mathbf{v}_i = \mathbf{x}_i + F \cdot \big(\mathbf{x}_{r_1} - \mathbf{x}_i\big) + F \cdot \big(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}\big)$

where $\mathbf{x}_i$ is the current (or target) individual, $\mathbf{x}_{best}$ is the current best one, and $\mathbf{x}_{r_1}, \mathbf{x}_{r_2}, \mathbf{x}_{r_3}, \mathbf{x}_{r_4}$ and $\mathbf{x}_{r_5}$ are individuals uniformly drawn in the population.

A first range of experiments considers DE using a single mutation strategy, and a uniform selection of the mutation strategy. A second range of experiments considers the adaptive schemes introduced in section 2. The *PM-AdapSS-DE* [10] uses the *Probability Matching* (*PM*) method coupled with the average relative fitness improvement gathered during the current generation. *Adaptive Pursuit* (*AP*) [20] and *Dynamic MAB* (*DMAB*) [4] use as credit assignment the extreme fitness value over a window of size W, as it was found to be the best one in earlier extensive experiments (albeit in a different context) [6,7].

For the sake of a fair empirical comparison, the parameters of the adaptive schemes, referred to as *hyper-parameters*, have been tuned using a racing technique; due to space limitations, the reader is referred to [10,20,4] for a detailed description of the hyper-parameters mentioned in the following.

The comparative validation thus shows the peak performance of each scheme, where the best hyper-parameter configuration has been determined by means of[1] *F-Race* [2]. The hyper-parameter tuning considers the performance of each hyper-parameter over all functions for a given dimension within the benchmark suite; the first elimination round happens after one run over all functions, and it goes on until achieving 10 runs or pruning all configurations but one.

Following this methodology, the hyper-parameters of *F-AUC-Bandit* are varied as follows. The scaling factor C is varied in $\{\{1, 5\}.10^{\{-2\leq i\leq 1\}}, 100\}$; the window size W in $\{50, 100, 500\}$; the decay factor D is set to .5, giving much more weight to the top-ranked rewards (although in a smoother way than the extreme value based reward mechanism [6]). The best configuration determined by the racing procedure over the benchmark suite, which will be used in all reported results, is: $C = .5$, $D = .5$, $W = 50$.

Along the same lines, the hyper-parameters of *PM-AdapSS-DE* are varied as follows: minimal probability $p_{min} \in \{0, .05, .1, .2\}$, adaptation rate $\alpha \in \{.1, .3, .6, .9\}$. Same values are tried for *AP*, with the additional learning rate β varied in $\{.1, .3, .6, .9\}$. The *DMAB* hyper-parameters are varied as follows: scaling factor $C \in \{\{1, 5\}.10^{\{-2\leq i\leq 1\}}, 100\}$, and change-detection threshold $\gamma \in \{Range(\mathcal{C}), 1000\}$. The window size W, involved in *AP* and *DMAB* is varied in $\{50, 100, 500\}$. Ultimately, the best *PM-AdapSS-DE* configuration is $p_{min} = 0$ and $\alpha = .6$; the best *AP* configuration is $p_{min} = .2$, $\alpha = .3$, $\beta = .3$, $W = 100$; the best *DMAB* configuration is $C = 100$, $\gamma = .1$, and $W = 50$.

Experiments are conducted using the BBOB-2010 noiseless testbed [12], including 24 single-objective functions from 5 different classes. They are performed following the default guidelines, 15 trials per function [11], with the maximum number of function evaluations being fixed at $10^5 \times d$. The BBOB-2010 experimental set-up uses as performance measurement the *Expected Running Time* (ERT), defined as follows: given a target function value, ERT is the empirical

[1] F-Race is an off-line tuning method, running all candidate configurations and stopping them as soon as it is shown to be statistically worse than the current best one at a given confidence level (95% in this case).

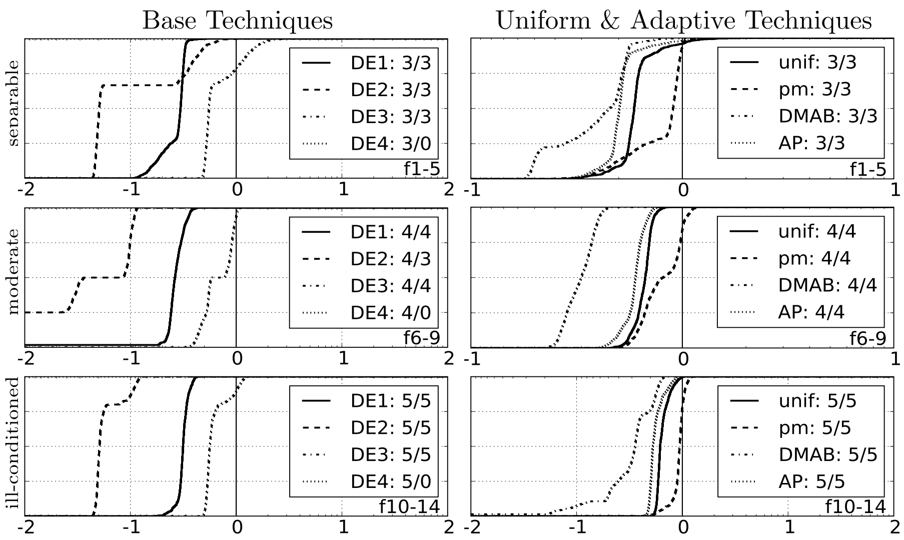


Fig. 2. Empirical cumulative distribution function of the speed-up ratios in dimension $d = 20$ for the *F-AUC-Bandit* compared with the base techniques (left) and with the uniform and adaptive ones (right). The speed-up ratios are the pairwise ratios of the number of function evaluations for *F-AUC-Bandit* to surpass the target function value 10^{-8} over the one of the baseline techniques over all trials of one functions. Pairs where both trials failed are disregarded, pairs where one trial failed are visible in the limits being > 0 or < 1 (for this reason, the lines for DE4 are not visible, as they coincide with the axes). The legends also indicate the number of functions that were solved in at least one trial (*F-AUC-Bandit* first).

expected number of function evaluations for attaining a fitness value below the target, *i.e.*, the ratio of the number of function evaluations for reaching the target value over successful trials, plus the maximum number of evaluations for unsuccessful trials, divided by the number of successful trials. Only the results over the *separable*, *moderate* and *ill-conditioned* function classes are reported here, for none of the considered schemes was able to perform well on the *multi-modal* and *weak-structure* ones with the given budget. Due to space constraints, the presented results are restricted to dimension $d = 20$, referring the reader to [8] for a comprehensive presentation including dimension 5, all function classes, and exhaustive pair-wise statistical comparisons between *F-AUC-Bandit* and the other schemes. The results are summarized in Fig. 2, being complemented by Table 1.

4.2 Comparative Results

F-AUC-Bandit is firstly compared with non-adaptive schemes referred to as DE1..DE4, and Uniform-DE, respectively using the single mutation strategy 1..4, and a uniformly selected mutation strategy. DE4 shows unable to solve any of the functions in dimension $d = 20$, and it is thus discarded in the following (although being occasionally used by the adaptive schemes). All other non-adaptive

Table 1. Median ERT speed-up in dimension $d = 20$ (inter-quartile range in brackets) for a given budget of FEvals. For a given test function, the ERT speed-up is computed as the ratio of the ERT of the algorithm considered (row) over the ERT of *F-AUC-Bandit* (median and inter-quartile range given in first row) for the smallest function value attained by it after a budget of 10, 10^3, 10^5 times the dimension function evaluations or 10^{-8} if it was smaller. The best three values are in bold. The probability of success for reaching the precision 10^{-8} is given in the rightmost column.

(a) Base Techniques

F	budg.	200	20,000	2M	p_s
separable	FAUC	**43**(71)	**16e3**(14e3)	**51e3**(42e4)	.6
	DE1	1 (.84)	**3.3** (.97)	**3.2** (2.1)	.6
	DE2	**1** (4.3)	20 (6.2)	21 (∞)	.6
	DE3	**.86** (.14)	**1.7** (.56)	**1.9** (2.6)	.6
	DE4	1 (.39)	∞	∞	0
moderate	FAUC	**110** (16)	**16e3** (6e3)	**12e4** (5e4)	1
	DE1	1.7 (1.1)	**3.5** (.58)	**4** (1.1)	.98
	DE2	3.9 (3.8)	28 (3.4)	22 (∞)	.75
	DE3	**.92** (.48)	**1.9** (.31)	**1.5** (.92)	1
	DE4	**1.1** (.79)	1442 (1e3)	∞	0
ill-condit.	FAUC	67 (125)	**19e3** (314)	**52e3** (52e3)	1
	DE1	**.87** (.55)	**3.2** (.2)	**3.2** (.23)	1
	DE2	1.2 (2.7)	20 (1.9)	20 (3.4)	1
	DE3	**.89** (.61)	**1.9** (.15)	**1.8** (.35)	1
	DE4	**.69** (.53)	∞	∞	0

(b) Uniform & Adaptive Techniques

budg.	200	20,000	2M	p_s
FAUC	**43**(71)	**16e3**(14e3)	**51e3**(42e4)	.6
UNIF	1 (.28)	**1.6** (.16)	**1.6** (.63)	.6
PM	**1** (.42)	**1.1** (.25)	**1.1** (.88)	.6
AP	**.82** (.53)	1.9 (.22)	1.9 (.5)	.6
DMAB	1 (.42)	7.7 (6 .8)	3.5 (∞)	.6
FAUC	**110** (16)	**16e3** (6e3)	**12e4** (5e4)	1
UNIF	**.89** (.35)	**1.7** (.23)	**1.5** (.12)	1
PM	**1.1** (1.1)	**1.1** (.06)	**1.3** (.49)	1
AP	1.1 (.45)	1.9 (.21)	1.6 (.22)	1
DMAB	1.1 (5.6)	10 (7)	3.1 (.76)	1
FAUC	67 (125)	**19e3** (314)	**52e3** (52e3)	1
UNIF	**.86** (.54)	**1.7** (.08)	**1.7** (.24)	1
PM	1 (.44)	**1.1** (.03)	**1.1** (.11)	1
AP	**9** (64)	1.9 (.14)	1.9 (.32)	1
DMAB	1.3 (2)	6.1 (8)	3.9 (2.6)	.99

schemes achieve the target value on all trials for the *ill-conditioned* functions, and on 60% of the trials for the *separable* ones, failing on the multi-modal ones. For the *moderate* functions, both *F-AUC-Bandit* and *DE*3 are able to achieve 100% of success, while *DE*1 and *DE*2 respectively get 98% and 75% success.

Compared with *DE*1, *F-AUC-Bandit* shows to be around 3 times faster on the 3 analyzed function classes. *DE*2 is around 20 times slower than *F-AUC-Bandit* on around 65%, 50% and 80% of the trials, respectively, for the *separable*, *moderate* and *ill-conditioned* function classes. *DE*3 is the best one out of the single strategies, performing 10 times faster than *DE*2; overall, it is around 2 times slower than *F-AUC-Bandit.*

F-AUC-Bandit shows to be around 1.5 times faster than *Uniform-DE* in around 80% of the trials; the difference is statistically significant for most functions (referring the reader to the pair-wise statistical comparisons presented in [8]). This difference seems to be moderate, relatively to the price to pay for an adaptive scheme. It might thus be observed that the uniform strategy considers here a small number of strategies, most of which perform well: *DE*1 and *DE*3 perform quite well; although much slower, *DE*2 still reaches the target; the only inefficient strategy is *DE*4. In the general case however, the performance of the strategies is unknown; the performance of the above strategies was assessed through extensive experiments. The use of an *Adaptive Strategy Selection* scheme is thus relevant in the general case.

The last series of experiments deals with the previously proposed *AdapSS* schemes, *PM-AdapSS-DE* [10], *AP* [20] and *DMAB* [4], the two latter schemes being fed by extreme rewards [6]. Compared with *AP*, *F-AUC-Bandit* is around 1.5 times faster on around 90% of the trials on the three function classes. It also shows to be around 3 times faster than *DMAB* on half of the trials, being at least around 1.5 times faster on all trials. *PM-AdapSS-DE* shows to be the best out of the three other adaptive schemes, which is attributed to the use of a relative instead of a raw reward. *F-AUC-Bandit* is around 1.5 times faster than *PM-AdapSS-DE* on around 25% of the trials on the *separable*, and 40% for the *moderate* functions, with an even smaller performance gain on the *ill-conditioned* ones, although still being faster on around 75% of the functions.

This performance improvement of the *F-AUC-Bandit* w.r.t. the others is attributed mostly to: (i) the use of a comparison-based *Credit Assignment*, which is robust to all the very different situations tackled within this benchmark suite, while efficiently following the changes in the qualities of the strategies (the reduction of the AUC for one operator, by definition, results in the augmentation of the AUC for one of the others); and to (ii) the use of a bandit-based *Strategy Selection*, which has already shown to be very efficient in the GA context [6,7].

5 Conclusion and Perspectives

F-AUC-Bandit, an *Adaptive Operator Selection* scheme that has been recently proposed within the GA framework [9], is fully comparison-based. Thus, when used within a comparison-based algorithm, the invariance w.r.t monotonous transformations of the fitness is preserved. *F-AUC-Bandit* has been used here to select between mutation strategies within a *Differential Evolution* (DE) algorithm. Such combination has been assessed on a set of single-objective continuous problems, defined in the BBOB-2010 [11] noiseless benchmark: *F-AUC-Bandit*+DE was empirically compared with naive DE schemes, as well as with 3 other adaptive schemes from the literature, *PM-AdapSS-DE* [10], *AP* [20], and *DMAB* [4]. In terms of *expected running time* to achieve a given function target value, *F-AUC-Bandit*+DE was found to outperform the other techniques.

Furthermore, *F-AUC-Bandit* was found to be very robust w.r.t. the tuning of its hyper-parameter – the same setting was found by the racing procedure for all dimensions and all function classes. However, the robustness of such tuning needs to be further assessed, as was done in the GA framework [9].

Nevertheless, the main goal of this work has been reached – validate the *F-AUC-Bandit* approach in a different context than the one it had been designed for originally. However, though much improved over the results of all naive strategies used within DE, the best results of the *F-AUC-Bandit* +DE algorithm remains below those of state-of-the-art optimizers [13]. But the DE algorithms to which *F-AUC-Bandit* has been applied here only use the basic DE techniques, and several improvements have been recently proposed, *e.g.*, adding adaptive parameter control for F and CR [18] . The applicability of *F-AUC-Bandit* in DE framework opens the path for fruitful research using the numerous recent DE variants.

Another further work is to address the multi-modality issue: all tested algorithms fail on multi-modal functions (40% of the separable class – see Table 1). On-going work is concerned with preserving the comparison-based property in the framework of the rewards proposed in [15] to tackle multi-modality.

References

1. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multi-armed bandit problem. Machine Learning 47(2-3), 235–256 (2002)
2. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B., et al. (eds.) Proc. GECCO, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
3. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30, 1145–1159 (1997)
4. Da Costa, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive operator selection with dynamic multi-armed bandits. In: Keijzer, M., et al. (eds.) Proc. GECCO, pp. 913–920. ACM, New York (2008)
5. Davis, L.: Adapting operator probabilities in genetic algorithms. In: Schaffer, J.D. (ed.) Proc. ICGA, pp. 61–69. Morgan Kaufmann, San Francisco (1989)
6. Fialho, A., Da Costa, L., Schoenauer, M., Sebag, M.: Extreme value based adaptive operator selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 175–184. Springer, Heidelberg (2008)
7. Fialho, A., Schoenauer, M., Sebag, M.: Analysis of adaptive operator selection techniques on the royal road and long k-path problems. In: Raidl, G., et al. (eds.) Proc. GECCO, pp. 779–786. ACM, New York (2009)
8. Fialho, A., Ros, R.: Adaptive strategy selection within differential evolution on the BBOB-2010 noiseless benchmark. Research Report RR-7259, INRIA (2010)
9. Fialho, A., Schoenauer, M., Sebag, M.: Toward comparison-based adaptive operator selection. In: Branke, J., et al. (eds.) Proc. GECCO. ACM Press, New York (2010)
10. Gong, W., Fialho, A., Cai, Z.: Adaptive strategy selection in differential evolution. In: Branke, J., et al. (eds.) Proc. GECCO. ACM Press, New York (2010)
11. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2010: Experimental setup. Tech. Rep. RR-7215, INRIA (2010)
12. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA (2009)
13. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In: GECCO Workshop Proceedings. ACM Press, New York (2010)
14. Julstrom, B.: What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm. In: Eshelman, L.J., et al. (eds.) Proc. ICGA, pp. 81–87. Morgan Kaufmann, San Francisco (1995)
15. Maturana, J., Lardeux, F., Saubion, F.: Autonomous operator management for evolutionary algorithms. Journal of Heuristics (2010)
16. Page, E.: Continuous inspection schemes. Biometrika 41, 100–115 (1954)
17. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)
18. 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)
19. Storn, R., Price, K.: Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4) (1997)
20. Thierens, D.: Adaptive strategies for operator allocation. In: Lobo, F.G., et al. (eds.) Parameter Setting in Evolutionary Algorithms, pp. 77–90. Springer, Heidelberg (2007)

Fixed Parameter Evolutionary Algorithms and Maximum Leaf Spanning Trees: A Matter of Mutation

Stefan Kratsch[1], Per Kristian Lehre[2,3], Frank Neumann[1], and Pietro Simone Oliveto[3]

[1] Algorithms and Complexity, Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] DTU Informatics, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark
[3] School of Computer Science, University of Birmingham, Birmingham, United Kingdom

Abstract. Evolutionary algorithms have been shown to be very successful for a wide range of NP-hard combinatorial optimization problems. We investigate the NP-hard problem of computing a spanning tree that has a maximal number of leaves by evolutionary algorithms in the context of fixed parameter tractability (FPT) where the maximum number of leaves is the parameter under consideration. Our results show that simple evolutionary algorithms working with an edge-set encoding are confronted with local optima whose size of the inferior neighborhood grows with the value of an optimal solution. Investigating two common mutation operators, we show that an operator related to spanning tree problems leads to an FPT running time in contrast to a general mutation operator that does not have this property.

1 Introduction

Evolutionary Algorithms (EAs) are a large class of stochastic search algorithms that are widely used to solve combinatorial optimization problems. They have found many applications for different kinds of NP-hard spanning tree problems (see e. g. [10,5]). Our aim is to contribute to the theoretical understanding of evolutionary algorithms for such kind of problems. Rigorous runtime analyses have been widely used to provide theoretical insights into the optimization process of evolutionary algorithms and we follow this line of research throughout this paper. The first runtime analyses of EAs were performed on artificially created pseudo-Boolean functions to understand what characteristics of a problem make its optimisation easy or hard for an EA (see e. g. [2]). These first efforts led to the development of a range of mathematical techniques used for the analyses. Building up on these results it has been possible to analyse the performance of EAs on classical combinatorial optimisation problems (see [9] for an overview).

Recently, the notion of fixed parameter tractability has been introduced into the theoretical analysis of evolutionary algorithms [6]. A parameterized analysis allows a more detailed inspection on which instances of an NP-hard combinatorial optimization problem are hard to solve. Such an analysis depends on a parameter k which measures the difficulty of the problem under consideration. In parameterized complexity, a problem with parameter k is *fixed parameter tractable* (FPT) if there exists an algorithm that decides it in time $O(f(k) \cdot n^c)$ [1]. Hence, the runtime of the *FPT algorithm* is $O(n^c)$ for every fixed value of k. For many real-world instances of NP-hard problems, the parameter k in question is bounded, and not too large. Hence, these problem instances can solved by FPT-algorithms in polynomial time, despite the general problem being NP-hard. We point out that problems considered in parameterized complexity often have straightforward $O(n^{f'(k)})$ time algorithms; however, while also polynomial for every fixed k, the degree of the polynomial does depend on k. Fixed-parameter evolutionary algorithms are evolutionary algorithms that compute an optimal solution in expected time $O(f(k) \cdot n^c)$. In [6], it has been shown that there are fixed parameter evolutionary algorithms for the vertex cover problem.

We put forward the parameterized analysis of evolutionary algorithms and investigate this kind of algorithms for the computation of a maximum leaf spanning tree. There are different approximation algorithms based on local search for this problem that give a constant approximation ratio [8,7]. On the other hand, it is known that the problem is APX-complete [4]. We consider exact optimization and investigate two evolutionary algorithms working with an edge-set encoding [10] which is very popular when solving spanning tree problems. Our algorithms differ from each other by the chosen mutation operator. The more general mutation operator is motivated by standard bit-mutation and does not necessarily create a tree whereas the second (more problem-specific) operator makes sure that each created offspring is a tree. We present instances containing a local optima that is hard to leave if the value of an optimal solution is large. Based on the structure of these instances, we prove lower bounds on the expected optimization time for both algorithms which grow with the value of an optimal solution. Later on, we show that the more problem specific mutation operator leads to fixed parameter evolutionary algorithms for the maximum leaf spanning tree problem, while the more general mutation operator does not have this property according to our proven lower bounds.

After having motivated our work, we introduce the problem and algorithms in Section 2. We present an instance with a local optimum and a large inferior neighborhood in Section 3. In Section 4, we show that a suitable mutation operator leads to fixed parameter evolutionary algorithms for the maximum leaf spanning tree problem. Finally, we finish with some concluding remarks.

2 Problem and Algorithms

We investigate the following NP-hard spanning tree problem. Given an undirected connected graph $G = (V, E)$, the goal is to find a spanning tree T^* of G such that the number of leaves is maximal.

We consider two simple evolutionary algorithms which differ by the choice of the mutation operator. Both algorithms start with an arbitrary spanning tree T

of G. We denote by m the number of edges in G, and $\ell(T)$ the number of leaves of the spanning tree T. A new solution is only accepted if it is a spanning tree whose number of leaves is at least as high as the number of leaves in the current solution. The first algorithm can be described as follows.

Algorithm 1 (Generic (1+1) EA)

```
1. Choose a spanning tree of $T$ uniformly at random.
2. Produce $T'$ by swapping each edge of $T$ independently with probability $1/m$.
3. If $T'$ is a tree and $\ell(T') \geq \ell(T)$, set $T := T'$.
4. Go to 2.
```

Swapping an edge in step 2. of Algorithm 1 means that if an edge is present in T then it is not contained in T' with probability $1/m$. On the other hand, if an edge is not present in T then it is contained in T' with probability $1/m$. An edge does not change from T to T' with probability $1 - 1/m$ in each mutation step independently of the other edges. Note, that the mutation operator of Algorithm 1 does not necessarily create an offspring that is a tree. If the offspring is not a tree then this individual is discarded as it represents an infeasible solution.

Often it is assumed that choosing a mutation operator that is more tailored to the problem gives a significant speed up. The second algorithm uses a problem-specific mutation operator that ensures valid solutions, i. e. spanning trees.

Algorithm 2 (Tree-Based (1+1) EA)

```
1. Choose an arbitrary spanning tree $T$ of $G$.
2. Choose $S$ according to a Poisson distribution with parameter $\lambda = 1$ and perform sequentially $S$ random edge-exchange operations to obtain a spanning tree $T'$. A random exchange operation applied to a spanning tree $\tilde{T}$ chooses an edge $e \in E \setminus \tilde{T}$ uniformly at random. The edge $e$ is inserted and one randomly chosen edge of the cycle in $\tilde{T} \cup \{e\}$ is deleted.
3. If $\ell(T') \geq \ell(T)$, set $T := T'$.
4. Go to 2.
```

Our goal is to point out the differences between the two algorithms. To do this, we compare the expected number of iterations that our algorithms need to compute an optimal solution. The expected number of iterations needed to obtain an optimal solution is called the *expected optimization time*, and is the commonly used performance measure in the rigorous runtime analysis of evolutionary algorithms. We will show that choosing the more problem-specific mutation operator of Algorithm 2 makes the difference between a fixed-parameter evolutionary algorithm and an evolutionary algorithm that does not compute an optimal solution within expected FPT-time.

3 Local Optima and Lower Bounds

The aim of this section is to point out structures of the problem that make it hard for our algorithms to achieve an improvement. We discuss the presence of local

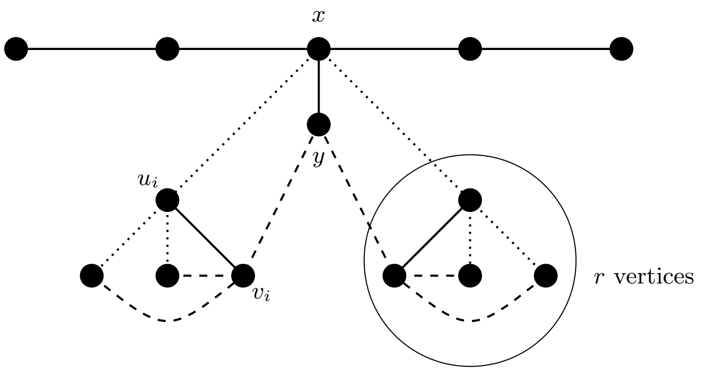


Fig. 1. Local optimum shown with dashed edges, global optimum with dotted edges, shared edges are drawn solid

optima and present a graph that consists of a local optimum which has a large distance (in terms of the number of edge exchanges) from the global optimum. Using this observation, we show lower bounds on the expected optimization time for the two algorithms under consideration.

Our graph called G_{loc} (see Figure 1) contains two components consisting of r vertices each. In component i, $1 \leq i \leq 2$, two vertices u_i and v_i are connected to all the other vertices in that component. The vertex u_i is connected to vertex x which lies outside the component. Similarly vertex v_i is connected to vertex y. In addition, x and y share an edge. The graph is completed by attaching a path of $n-2r-2$ vertices to the vertex x. A tree has to contain all the edges of the path attached to x. For a given component, the maximal number of possible leaves is at most $r-1$. This can be obtained by attaching all nodes of the component either to u_i or v_i.

The graph contains a local optimum T_{lopt} which consists of all edges attached to the vertices v_i, $1 \leq i \leq 2$, the edge $\{x, y\}$ and all path edges. The global optimum T_{opt} consists of all edges attached to the vertices u_i, $1 \leq i \leq 2$, the edge $\{x, y\}$ and all path edges. Compared to T_{lopt}, T_{opt} has an extra leaf, namely the vertex y. However, T_{lopt} and T_{opt} differ by $4(r-1)$ edges which make it hard for the algorithms under consideration to obtain T_{opt} if T_{lopt} has been produced before.

Our goal is to study the expected optimization time of the algorithms introduced in the previous section in dependence of the number of leaves which, in turn, depends on r. To do this, we first consider the number of different spanning trees of G_{loc} in dependence of r.

Lemma 1. *The number of spanning trees of G_{loc} is at most 2^{4r}.*

Proof. A spanning tree has to contain all edges of the path attached to x. The path attached to x consists of $n-2r-2$ edges. A spanning tree contains exactly $n' = n - 1 - (n - 2r - 2) = 2r + 1$ non-path edges.

We count the total number of non-path edges in G_{loc}. Consider a component consisting of r edges. The number of edges within such a component is $2r-3$ as u_i and v_i are connected to all other vertices and share an edge. In addition there are two edges connecting each component to the outer part. Hence, the total number of edges connected to vertices of a single component is $2r-1$. In addition, there is the edge connecting x and y.

Summing up, the graph consists of $m' = 2(2r-1)+1 = 4r-1$ non-path edges. The number of different spanning trees is therefore at most

$$\binom{m'}{n'} = \binom{4r-1}{2r+1} \leq 2^{4r}.$$

□

Using the previous lemma, we show the following lower bound on the expected optimization time of Generic (1+1) EA on G_{loc}.

Theorem 1. *The expected optimization time of Generic (1+1) EA on G_{loc} is lower bounded by $\left(\frac{m}{c}\right)^{2(r-2)}$ where c is an appropriate constant.*

Proof. The number of spanning trees of G_{loc} is at most 2^{4r}. Therefore, the initial spanning tree is T_{lopt} with probability at least 2^{-4r}. This spanning tree is a local optimum with $2(r-1)+2$ leaves. In order to obtain a different spanning tree with at least as many leaves, $r-1$ leaves have to be achieved in each component, or at least $r-1$ leaves have to be obtained in one component and y has to become a leaf. Hence, in order to achieve an accepted solution that is different from T_{lopt} all $(r-2)$ nodes of at least one component i have to be assigned to u_i instead of v_i. This implies that at least $2(r-2)$ edges for a fixed component have to be swapped to escape from the local optimum. There are two components where this can happen which implies that the probability for such a step is at most $2\left(\frac{1}{m}\right)^{2(r-2)}$. The expected waiting time for such a step is at least $\frac{1}{2} \cdot m^{2(r-2)}$. Altogether the expected optimization time is lower bounded by

$$2^{-4r} \cdot \frac{1}{2} \cdot m^{2(r-2)} \geq \left(\frac{m}{c}\right)^{2(r-2)},$$

where c is an appropriate constant. □

Using the previous ideas, we can also lower bound the expected optimization time of Tree-Based (1+1) EA on G_{loc}.

Theorem 2. *The expected optimization time of Tree-Based (1+1) EA on G_{loc} is lower bounded by $(\frac{r-2}{c})^{r-2}$ where c is an appropriate constant.*

Proof. We follow the ideas of the previous theorem. With probability at least 2^{-4r}, T_{lopt} is chosen as the initial spanning tree. In order to produce from T_{lopt} the optimal solution T_{opt}, $(r-2)$ exchange operations have to be carried out in a single mutation step. According to the Poisson distribution with $\lambda = 1$, the probability that this happens in the next step is

$$\frac{1}{e(r-2)!} \leq \frac{1}{\sqrt{2\pi(r-2)}} e^{r-3}(r-2)^{-(r-2)} \leq e^{r-3}(r-2)^{-(r-2)}.$$

Altogether the expected optimization time is lower bounded by

$$2^{-4r} \cdot e^{-r+3}(r-2)^{(r-2)} \geq \left(\frac{r-2}{c}\right)^{r-2},$$

where c is an appropriate constant. □

To show that both algorithms need not only in expectation that many steps, but also with a high probability the graph can be modified such that it consists of more than two components attached to x and y. Then a typical run can be investigated to show that at least two components end up in the local optimum.

4 FPT of Edge Exchanges

In this section we prove that Algorithm 2 is an FPT algorithm for the maximum leaf spanning tree problem with respect to the maximal number of leaves k. Given that the maximal-leaf spanning tree has k leaves, in the following lemma we derive upper bounds in dependence of k on the number of edges and on the number of nodes of degree at least three that the graph may contain. These bounds will allow us to prove the main result of this section presented in Theorem 3.

The lemma is proven using an approach similar (but greatly simplified) to the one used in [3]; our focus here is on giving a self-contained presentation sufficient for obtaining the claimed expected runtime. Note also, that kernelization results, such as [3], almost always require a modification of the problem instance while we are interested in bounding the original instance.

Lemma 2. *Any connected graph G on n nodes and with a maximum number of k leaves in any spanning tree has at most $n+5k^2-7k$ edges and at most $10k-14$ nodes of degree at least three.*

Proof. Let G be a graph on n nodes and let T be a spanning tree of G with (the maximum number of) k leaves. We let P_0 denote the set of all leaves and all nodes of degree at least three in T. (We denote the degree of node x within the tree T by $\deg_T(x)$.) Furthermore, let $P \supseteq P_0$ denote the set of all nodes that are within distance of at most two of any node of P_0 (distance and degree w.r.t. T). We let Q denote the set of remaining nodes.

In the following we show that all nodes of Q have degree two in G (clearly they have degree at least two in G since they have degree two in T). We assume for contradiction that there is a node $v \in Q$ which has degree at least three in G. Therefore, v has a neighbor u in G to which it is not adjacent in T. We distinguish two cases:

If u is at distance two from v (w.r.t. T) then it is neither a leaf nor does it have degree greater than two. Let w be the node that is adjacent to both u and v in T. Observe that adding the edge $\{u, v\}$ to T and removing $\{v, w\}$ creates a spanning tree with an additional leaf, namely w (note that the graph does not disconnect since we remove an edge of a cycle). This contradicts our choice of T.

If u is at distance greater than two from v (w.r.t. T) then we observe the following: Adding the edge $\{u, v\}$ to T creates a cycle $C = \{\ldots, u, v, w, x, \ldots\}$ on at least four nodes (else u would be too close to v). Since $v \in Q$ both nodes w and x (to which v is at distance at most two in T) have degree two in T. Thus adding the edge $\{u, v\}$ to T and removing $\{w, x\}$ would give two additional leaves w and x while possibly losing the leaf u (again, removing an edge from a cycle must give a connected graph). This contradicts our choice of T.

Thus all nodes in Q have degree two in G. We now bound the size of P. To this end, we make the following observations: T has k leaves and thus it has at most $k-2$ nodes of degree at least three. Also, the number of leaves in any tree is equal to 2 plus $deg(v) - 2$ for every node v of degree at least three, so

$$\sum_{v:deg_T(v)\geq 3} (deg_T(v) - 2) = k - 2.$$

The number of elements in P_0 are the k leaves, plus the at most $k-2$ nodes of degree at least 3. Let P_1 be the set of nodes at distance 1 from P_0, and P_2 the set of nodes at distance 2 from P_0. The number of nodes in P_1 that are connected to a leaf node in P_0 is at most k. The number of nodes in P_1 that are connected to a node of degree at least 3 in P_0 is at most

$$\begin{aligned}\sum_{v:deg_T(v)\geq 3} deg_T(v) &= \sum_{v:deg_T(v)\geq 3} (deg_T(v) - 2 + 2)\\ &\leq 2(k-2) + \sum_{v:deg_T(v)\geq 3} (deg_T(v) - 2) \leq 3k - 6.\end{aligned}$$

In total, there are no more than $4k - 6 = k + (3k - 6)$ nodes in P_1. Finally, each node in P_1 has degree two and is adjacent to at least one node of P_0. Furthermore, each node of P_2 is adjacent to at least one node of P_1. Therefore $|P_2| \leq |P_1|$. In total, there are no more than $|P_0| + |P_1| + |P_2| \leq 10k - 14$ nodes in P. Clearly, $10k - 14$ is also an upper bound on the number of nodes of degree at least three in G since they cannot be contained in Q.

To bound the number of edges we observe that no node of G can have degree greater than k: Starting a tree from a node of degree greater than $k+1$ and adding the remaining nodes to this tree would give a spanning tree with more than k leaves. Thus we get the claimed upper bound on the number m of edges:

$$m \leq \frac{1}{2}(k|P| + 2|Q|) = \frac{k}{2}|P| + |Q| \leq 5k^2 - 7k + n.$$

This completes the proof. □

Now we are ready to prove the main result. Since a spanning tree always has $n-1$ edges, from Lemma 2 there are at most $5k^2$ edges to choose from at each step and at most all of them need to be replaced to reach the optimal spanning tree. The proof of the following theorem first shows that the probability of increasing the number of leaves by one in the current (non-optimal) spanning tree only decreases with the fixed parameter k. The proof is concluded by showing that the probability of exchanging all the $5k^2$ edges in one mutation step also depends only on k leading to the claimed runtime.

Theorem 3. *If the maximal number of leaf nodes in any spanning tree of G is k, then Algorithm 2 finds an optimal solution in expected time $O(2^{15k^2 \log k})$.*

Proof. Let $n_{\geq 3}$ be the number of nodes with degree at least three. We call an edge *distinguished* if it is incident on a node of degree at least 3, and non-distinguished otherwise. By applying Lemma 2, the number of distinguished edges on any cycle is at most $2n_{\geq 3} \leq 20k - 28$, since there are at most $n_{\geq 3}$ nodes of degree at least 3 on the cycle, and each node is incident with at most two edges of the cycle.

We first bound the probability of reducing the distance to an optimal spanning tree by 1. Let $E^* \subseteq E$ be the optimal spanning tree that is closest to the current spanning tree, and let e be any edge in E^* that is not yet in the current spanning tree. By Lemma 2, the number of edges in the graph is $m \leq n + 5k^2 - 7k$. So the probability that edge e is introduced in an edge exchange operation is at least $1/(m-(n-1)) \geq 1/5k^2$. Introducing edge e creates a cycle. Consider first the case when the cycle consists only of distinguished edges. The length of such a cycle is no more than $20k - 28$, and the probability of removing one of the edges that is not in the optimal spanning tree is at least $1/20k$. In the case where the cycle contains non-distinguished edges, we claim that it suffices to remove any non-distinguished edge e' from the cycle. The claim obviously holds when the chosen edge e' is not in the optimal spanning tree, so assume that edge e' is in the optimal spanning tree. A *bridge edge* in a connected graph is any edge e such that the subgraph on the edges $E \setminus \{e\}$ is disconnected. Edge e' connects two components T_1 and T_2 in E^*, and cannot be a bridge edge because then the edge could not have been part of a cycle. Since edge e connects T_1 and T_2, the cycle must contain at least one other edge e'' that connects T_1 and T_2, and this edge is not part of the optimal spanning tree E^*. However, the spanning tree $(E^* \setminus \{e'\}) \cup \{e''\}$ must also be optimal, because adding edge e'' decreases the number of leaf nodes by at most 2, and removing edge e' increases the number of leaf nodes by exactly 2. Hence, adding edge e and removing edge e' reduces the distance to an optimal spanning tree by 1. Let ℓ be the number of non-distinguished edges on the cycle. No cycle contains more than $20k - 28$ distinguished edges, so the probability of removing a non-distinguished edge is at least $\ell/(20k - 28 + \ell) \geq 1/20k$. The probability of reducing the distance to a global optimum by 1 is therefore at least $1/(20k \cdot 5k^2)$.

The number of edges r that must be inserted in the spanning tree is no more than $m - (n-1) \leq 5k^2$. The edges can be inserted in any order. The probability that in Step 2 of the algorithm, we choose to do $S = r$ operations is $1/er!$. So, the probability that in one step, we decide to do r edge exchange operations in any of the $r!$ orders, and each of the edge exchanges decreases the Hamming distance to an optimal spanning tree is at least

$$r! \cdot \frac{1}{er!} \cdot \left(\frac{1}{5k^2} \cdot \frac{1}{20k}\right)^r \geq \frac{1}{e}\left(\frac{1}{100k^3}\right)^{5k^2} \geq \frac{1}{e}\left(\frac{1}{100}\right)^{5k^2}\left(\frac{1}{k}\right)^{3\cdot 5k^2},$$

which implies that the expected number of steps to find an optimal spanning tree is at most $O(2^{15k^2 \log k})$. □

5 Conclusions

The parameterized complexity analysis of evolutionary algorithms is a promising research direction that is likely to become an important part in the theoretical analysis of evolutionary computation during the next years. An advantage in comparison to classical worst-case considerations is that this kind of analysis gives characterizations of what difficult instances for a specific algorithm look like in relation to some parameter of the problem. Evolutionary algorithms have produced very good results for different kind of NP-hard spanning tree problems. In this paper, we have studied evolutionary algorithms for the NP-hard maximum leaf spanning tree problem in the context of parameterized complexity. In our case the parameter is the size of the global optimum. Our investigations show that there may be local optima where the size of an inferior neighborhood grows with the number of leaves in optimal solutions. Investigations of two common mutation operators point out that a more problem-specific operator makes the difference between a fixed parameter evolutionary algorithm for the maximum leaf problem and an algorithm that does not have this property.

Acknowledgements. Per Kristian Lehre was supported by EPSRC under grant no. EP/D052785/1, and Deutsche Forschungsgemeinschaft (DFG) under grant no. WI 3552/1-1. Pietro Simone Oliveto was supported by EPSRC under grant no. EP/P502322/1.

References

1. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Monographs in Computer Science. Springer, Heidelberg (1998)
2. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
3. Fellows, M.R., Lokshtanov, D., Misra, N., Mnich, M., Rosamond, F.A., Saurabh, S.: The complexity ecology of parameters: An illustration using bounded max leaf number. Theory Comput. Syst. 45(4), 822–848 (2009)
4. Galbiati, G., Maffioli, F., Morzenti, A.: A short note on the approximability of the maximum leaves spanning tree problem. Inf. Process. Lett. 52(1), 45–49 (1994)
5. Knowles, J.D., Corne, D.: A comparison of encodings and algorithms for multiobjective spanning tree problems. In: Proceedings of the Congress on Evolutionary Computation 2001, pp. 544–551. IEEE Press, Los Alamitos (2001)

6. Kratsch, S., Neumann, F.: Fixed-parameter evolutionary algorithms and the vertex cover problem. In: Rothlauf, F. (ed.) GECCO, pp. 293–300. ACM, New York (2009)
7. Lu, H.-I., Ravi, R.: Approximating maximum leaf spanning trees in almost linear time. J. Algorithms 29(1), 132–141 (1998)
8. Lu, H.-I., Ravi, R., Ravi, R.: The power of local optimization: Approximation algorithms for maximum-leaf spanning tree. In: Proceedings of Thirtieth Annual Allerton Conference on Communication, Control and Computing, pp. 533–542 (1996)
9. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4(3), 281–293 (2007)
10. Raidl, G.R., Julstrom, B.A.: Edge sets: an effective evolutionary coding of spanning trees. IEEE Trans. Evolutionary Computation 7(3), 225–239 (2003)

An Archive Maintenance Scheme for Finding Robust Solutions

Johannes Kruisselbrink, Michael Emmerich, and Thomas Bäck

LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden
{jkruisse,emmerich,baeck}@liacs.nl
http://natcomp.liacs.nl

Abstract. This paper presents an archive maintenance scheme that can be used within an evaluation scheme for finding robust optima when dealing with expensive objective functions. This archive maintenance scheme aims to select the additional sampling points such that locally well-spread distributions of archive points will be generated. By doing so, the archive will contain better predictive information about the robustness of candidate solutions. Experiments on 10D test problems show that this scheme can be used for accurate local search for robust solutions.

1 Introduction

Given an objective function, $f(\mathbf{x}) \to \min$, $\mathbf{x} \in \mathbb{R}^N$ for which it is required to account for fluctuations in the input variables, a frequently used approach is to determine the quality of a design vector $\mathbf{x}$ using the *expected* quality with respect to those fluctuations. Commonly, it is assumed that the underlying distribution of the fluctuations is known, which makes it possible to reformulate the objective function as a *robust* or *effective* objective function: $f_{\mathit{eff}} = \int_{-\infty}^{\infty} f(\mathbf{x}+\boldsymbol{\delta})\mathrm{pdf}(\boldsymbol{\delta})d\boldsymbol{\delta}$, where $\boldsymbol{\delta}$ denotes the variability in the input variables and $\mathrm{pdf}(\boldsymbol{\delta})$ is the probability density function of $\boldsymbol{\delta}$. However, for most problems it is impossible to find closed form expressions for f_{eff} and approximation methods are needed.

Two straightforward robust evaluation schemes in the context of EA are the SEM [10,9] and MEM [11] scheme that simply approximate the expected fitness by means of Monte-Carlo integration: $\hat{f}_{\mathit{eff}}(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}+\boldsymbol{\delta}_i)$, with $\boldsymbol{\delta}_i \sim \mathrm{pdf}(\boldsymbol{\delta})$. In the SEM scheme, a very crude approximation is obtained by taking $m = 1$, whereas in the MEM scheme, m can be set arbitrarily. Equivalent approaches were also proposed in [3] and [12]. Obviously, using such evaluation schemes yields noisy fitness approximations which can lead to undesirable effects in EA [1]. The advantage of using larger sample sizes is that the accuracy of the approximations increases, but this is at the cost of requiring extra objective function evaluations each generation. In effect, it introduces a trade-off between aiming for a high approximation accuracy and limiting the number of samples.

The SEM and MEM scheme can be said to form the basis of all other approaches that aim to find robust optima, but they have a clear weak point: noise is introduced in the form of approximation errors. Therefore, the main focus of other research on this topic is on dealing with the noise induced by the robustness approximations (e.g., [2]) or ways to obtain good (i.e., unbiased and high precision) robustness approximations with as limited additional sampling as possible (e.g., [3,4,8,7]).

When dealing with expensive objective function evaluations, using metamodels based on previously evaluated points is a good way to reduce function evaluations (e.g., [7,8]). However, metamodeling techniques are usually complex to implement and their construction sometimes becomes time expensive itself. Moreover, they still require an archive maintenance strategy that assures that the archive actually allows for accurate metamodeling.

In [3], a simple archive based approach was proposed for finding robust optima that does not require metamodeling. In this approach, the effective fitness of each individual $\mathbf{x}$ is approximated as the weighted mean of its own fitness and the fitness of previously evaluated neighboring points $\mathbf{x}'$ using:

$$\hat{f}_{\mathit{eff}}(x) = \frac{\sum_{\mathbf{x}'} w(\mathbf{x}') \cdot f(\mathbf{x}')}{\sum_{\mathbf{x}'} w(\mathbf{x}')}, \tag{1}$$

where $w(\mathbf{x}')$ is a weight function for which it should hold that $w(\mathbf{x}') \propto \text{pdf}(\delta)$. Although computing the distances between each pair $(\mathbf{x}, \mathbf{x}')$ introduces an overhead cost, in many cases it is fair to assume that the cost of evaluating one candidate solution is larger than this extra overhead cost.

The results reported on this scheme were promising and better compared to the other simple schemes that were considered. Moreover, this approach is a lot simpler than the metamodeling approaches. This motivates to further investigate whether this archive based approach can be improved to either become a simple alternative to metamodeling or to provide a useful archive maintenance method to assist metamodeling techniques. This paper will propose an archive maintenance scheme for robustness evaluation which enforces the retrieval of well-spread sets of points from the archive by including a resampling mechanism that aims for high local predictive capabilities.

The structure of this paper is as follows: Section 2 introduces the general concept of archive maintenance for finding robust solutions. Section 3 proposes a novel archive maintenance approach incorporated into an evaluation scheme. Section 4 presents some experimental results and Section 5 concludes with a discussion of the results and an outlook.

2 Archive Based EA for Finding Robust Solutions

The algorithm outlined in Figure 1 shows a generic framework of an archive based EA for finding robust optima. It is similar to a canonical EA, only a few extra steps are included involving the evaluation of candidate solutions.

For each individual, the algorithm attempts to select an appropriate sample set from the archive in order to obtain a robustness approximation according to (1). Note that in the approach of [3] all archive points were used for approximating the effective fitness. If there is no representative sample set present in the

```
 1: initialize parent population and archive
 2: while not terminate do
 3:     generate offspring from parents
 4:     for each offspring do
 5:         (optional: evaluate offspring and add to archive)
 6:         select archive points for robustness approximation
 7:         if not enough samples or no appropriate sample set available then
 8:             find appropriate additional sample points
 9:             evaluate additional sample points and add to archive
10:         end if
11:         compute robustness approximation using the selected samples
12:     end for
13:     select best offspring
14:     (optional: clean up archive)
15: end while
```

Fig. 1. General framework of an archive based EA for finding robust solutions

archive, the algorithm should decide which additional samples should be taken. Two optional steps are: 1) evaluate each individual first on the original objective function before evaluating its robustness, and 2) apply a method to clean up the archive and prevent it to grow out of bound.

Open issues are the selection of a set of samples from the archive and the selection of additional samples. Also, a criterion is needed for deciding whether additional samples are needed. Obviously, these issues are closely related.

3 A Novel Archive Based Approach

A major drawback of the approach presented in [3] is that the points in the archive are by no means guaranteed to be well-spread over the region of possible variation for each candidate solution. I.e., the archive points are selected by the optimization algorithm, which yields a distribution of points that is dependent on the way in which subsequent candidate solutions are selected. Especially in focused searching strategies such as ES, this might lead to an archive that holds limited information about the (local) objective function landscape.

In this paper, we propose to use a Latin Hypercube Sampling (LHS) [6] generated reference set for finding additional sample points for the evaluation of each individual. The main idea is that the most ideal set of points that could be taken from the archive would be a set of which the points are distributed in such a way that they could also have been obtained by means of a space filling experimental design. This is in line with the conclusions of [4] that LHS is the best sampling method to use in a MEM approach.

The algorithm in Figure 2 describes the archive based selection method. It takes as arguments a reference sample set $\mathbf{X}_{\text{ref}} = \{\mathbf{x}_r^{(1)}, \ldots, \mathbf{x}_r^{(m)}\}$ of m samples, which should be the LHS reference set for an individual for which a robustness estimate is required, and the archive $\mathbf{A} = \{(\mathbf{x}_a^{(1)}, f_a^{(1)}), (\mathbf{x}_a^{(2)}, f_a^{(2)}), \ldots\}$ which contains design point / objective function value tuples.

```
 1: procedure REFERENCE_SET_BASED_ARCHIVE_SELECTION($\mathbf{X}_{\text{ref}}$, $\mathbf{A}$)
 2:     $\mathbf{A}_{\text{sel}} \leftarrow \emptyset$
 3:     $\mathbf{X}_{\text{cand}} \leftarrow \emptyset$
 4:     for each $\mathbf{x}_r \in \mathbf{X}_{\text{ref}}$ do
 5:         $(\mathbf{x}_a, f_a) \leftarrow \{(\mathbf{x}_a, f_a) \in \mathbf{A} \mid \forall(\mathbf{x}'_a, f'_a) \in \mathbf{A} : d(\mathbf{x}_r, \mathbf{x}_a) \leq d(\mathbf{x}_r, \mathbf{x}'_a)\}$
 6:         if $(\forall \mathbf{x}'_r \in \mathbf{X}_r \mid d(\mathbf{x}_r, \mathbf{x}_a) \leq d(\mathbf{x}'_r, \mathbf{x}_a))$ then
 7:             $\mathbf{A}_{\text{sel}} \leftarrow \mathbf{A}_{\text{sel}} \bigcup \{(\mathbf{x}_a, f_a)\}$
 8:         else
 9:             $\mathbf{X}_{\text{cand}} \leftarrow \mathbf{X}_{\text{cand}} \bigcup \{\mathbf{x}_r\}$
10:         end if
11:     end for
12:     return $(\mathbf{A}_{\text{sel}}, \mathbf{X}_{\text{cand}})$
13: end procedure
```

Fig. 2. Reference set based archive selection

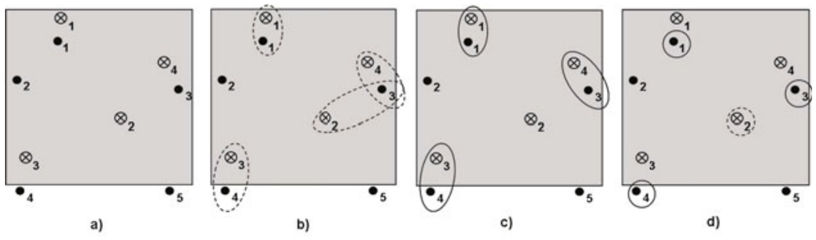


Fig. 3. An illustration of the LHS archive selection method

For each reference sample $\mathbf{x}_r$ in $\mathbf{X}_{\text{ref}}$ the algorithm searches for the closest point in the archive. Then, for each of these selected archive points, it is checked whether the reference points for which they are selected are also the closest reference points. The archive points for which this is the case are then, together with their objective function values, added to the set $\mathbf{A}_{\text{sel}}$ of selected archive points. For the archive points for which this is not the case one can conclude that the region around its reference point is underrepresented in the archive (making this reference point a good candidate for extra sampling). It is therefore added to the set $\mathbf{X}_{\text{cand}}$ of candidate sample points. In case that all reference points have been assigned archive points, then the reference point of the archive point / reference point pair that are located farthest apart from each other is selected as candidate for extra sampling[1]. This will assure that the archive is always updated in the region in which it needs the extra samples the most.

Figure 3 illustrates the proposed scheme: Step **a)** shows the archive points as solid circles and the reference samples with the $\otimes$-symbol. The box represents the region induced by the input noise. In step **b)** for each reference point, the closest archive point is identified. Then, in step **c)** it is checked whether the reference points are also the closest reference points of their selected archive

[1] This step is not shown in the algorithmic description of Figure 2.

```
1: $\mathbf{P} \leftarrow$ generate_initial_population()
2: $A \leftarrow \emptyset$
3: while not terminate do
4:     $\mathbf{O} \leftarrow$ generate_offspring($\mathbf{P}$)
5:     for each $\mathbf{x} \in \mathbf{O}$ do
6:         $\mathbf{X}_{\text{ref}} \leftarrow$ latin_hypercube_sampling($\mathbf{x}, \boldsymbol{\sigma}_\epsilon, m$)
7:         $(\mathbf{S}, \mathbf{X}_{\text{cand}}) \leftarrow$ reference_set_based_archive_selection($\mathbf{X}_{\text{ref}}, \mathbf{A}$)
8:         for a random $\mathbf{x}_c \in \mathbf{X}_{\text{cand}}$ do
9:             $f_c \leftarrow$ evaluate($\mathbf{x}_c$)
10:            $A \leftarrow A \bigcup \{(\mathbf{x}_c, f_c)\}$
11:        end for
12:        $\hat{f}_{eff}(\mathbf{x}) \leftarrow (\sum_{\{\mathbf{x_a}, f_a\} \in \mathbf{A}} w(\mathbf{x}_a) \cdot f_a)/(\sum_{\{\mathbf{x_a}, f_a\} \in \mathbf{A}} w(\mathbf{x}_a))$
13:    end for
14:    $\mathbf{P} \leftarrow$ select($\mathbf{O}$)
15: end while
```

Fig. 4. The new archive maintenance scheme incorporated into a general EA

points and in step **d)** the archive points for which this is the case are selected as archive points (solid circles) and the reference points for which this is not the case are selected as candidates for additional sampling (dashed circles).
The framework provided in Figure 1 combined with the archive selection scheme of Figure 2 yields an evaluation scheme for finding robust optima that can be integrated within any type of EA. Although there are still different ways to implement this scheme, space limitations do not allow for an extensive discussion.

This paper adopts the scheme as presented in Figure 4: For each individual, a reference set $\mathbf{X}_{\text{ref}}$ of m samples is used to obtain a set of selected archive points $\mathbf{S} = \{(\mathbf{x}_s^{(1)}, f_s^{(1)}), (\mathbf{x}_s^{(2)}, f_s^{(2)}), \ldots\}$ and a set $\mathbf{X}_{\text{cand}} = \{\mathbf{x}_c^{(1)}, \mathbf{x}_c^{(2)}, \ldots\}$ of suggested candidates for extra sampling. Then, in this variant, only one of the suggested candidate points for extra sampling is evaluated and added to $\mathbf{A}$. A robustness approximation is thereafter generated by considering all archive points and using the weighted function as in (1). Note that it is also possible to take only the set $\mathbf{S}$, however, we assume that by using the new resampling scheme, the archive is locally (i.e., around the current design point) well-spread and can therefore take all sample points. This should lead to more accurate robustness approximations in later stages (not being limited to a fixed sample size).

4 Experiments

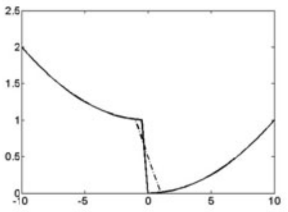
The proposed method, named ABRSS (*Archive Based Reference Set Selection*), is incorporated into a standard EA, the CMA-ES [5], and compared against other schemes also incorporated into the CMA-ES. These are: the SEM scheme, three MEM instances (with $m = 2$, $m = 5$, and $m = 10$), and an instance of the archive based scheme of [3] named PROX[2]. The experiments are performed on four 10D test problems (Figure 5) which represent two main robustness cases: f_1

[2] This name was used as this method selects archive points based on a proximity rule.

$$f_1(\mathbf{x}) = \left(1 - \prod_{i=1}^{N} f_{1a}(x_i)\right) + \sum_{i=1}^{N} \left(\frac{x_i}{10}\right)^2$$

$$f_{1a}(x_i) = \begin{cases} 0 & \text{, if } x_i < 0 \\ 1 & \text{, if otherwise} \end{cases}$$

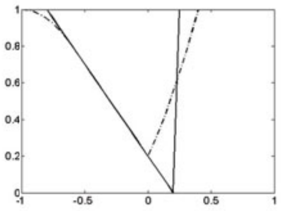
$$\mathbf{x} \in [-10, 10]^N \text{ and } \boldsymbol{\delta} \sim U(\mathbf{-1}, \mathbf{1})$$



$$f_2(\mathbf{x}) = 1 - \frac{1}{N} \sum_{i=1}^{N} f_{2a}(\mathbf{x})$$

$$f_{2a}(x_i) = \begin{cases} x_i + 0.8 & , -0.8 \leq x_i < 0.2 \\ 0 & \text{, otherwise} \end{cases}$$

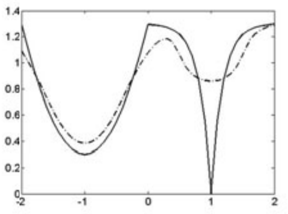
$$\mathbf{x} \in [-1, 1]^N \text{ and } \boldsymbol{\delta} \sim U(\mathbf{-0.2}, \mathbf{0.2})$$



$$f_3(\mathbf{x}) = c - \frac{1}{N} \sum_{i=1}^{N} f_{3a}(\mathbf{x})$$

$$f_{3a}(x_i) = \begin{cases} -(x_i + 1)^2 + 1 & , -2 \leq x_i < 0 \\ c \cdot 2^{-8|x_i - 1|} & , 0 \leq x_i < 2 \end{cases}$$

$$\mathbf{x} \in [-2, 2]^N \text{ and } \boldsymbol{\delta} \sim U(\mathbf{-0.5}, \mathbf{0.5})$$



$$f_4(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} f_{4a}(\mathbf{x})$$

$$f_{4a}(x_i) = \begin{cases} e^{-2\ln 2 \left(\frac{x-0.1}{0.8}\right)^2} \sqrt{|\sin(5\pi x_i)|} & , 0.4 < x_i \leq 0.6 \\ e^{-2\ln 2 \left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x_i) & \text{, otherwise} \end{cases}$$

$$\mathbf{x} \in [0, 1]^N \text{ and } \boldsymbol{\delta} \sim U(\mathbf{-0.0625}, \mathbf{0.0625})$$

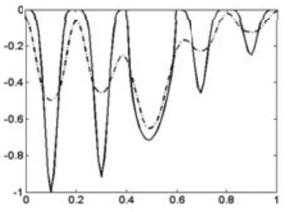


Fig. 5. The four test problems including 1D visualizations of the normal fitness functions (solid) and the effective fitness functions (dashed)

and f_2 are unimodal functions where the robust optimum is shifted with respect to the original optimum, and f_3 and f_4 are multimodal functions in which a local optimum becomes the robust optimum. As the main goal is to save on function evaluations, a limited evaluation budget of 5000 function evaluations is considered. The results are averaged over 50 runs for each scheme. For significance testing, a 95% confidence unpaired two-sided t-test was used.

The ABRSS scheme uses a reference set sample size of $m = 10$. The MEM schemes follow the suggestions of [4] by using LHS to obtain the sample sets. Also, the same disturbances are used for all individuals of the population. The PROX scheme uses the full archive for robustness approximation and does not use a duplicate avoidance scheme. Default strategy parameter settings as found in [5] are adopted for the CMA-ES.

4.1 Experimental Results

Figure 6 and Table 1 show the effective fitness of the best solution found after 1000 and 5000 evaluations respectively. Here, the effective fitness of the solutions is approximated using Monte-Carlo integration with $m = 1000$.

The experiments show that the ABRSS scheme is significantly outperforming all other schemes on the unimodal test functions f_1 and f_2; both in the early stages after 1000 evaluations as well as after 5000. Hence, the ABRSS scheme is both fast and accurate for a local optimization scheme. On these same functions, the PROX method also shows a good convergence speed in the beginning and good solution quality in the later stages, but, except on f_1 after 1000 evaluations, it is each time outperformed by at least one of the MEM approaches.

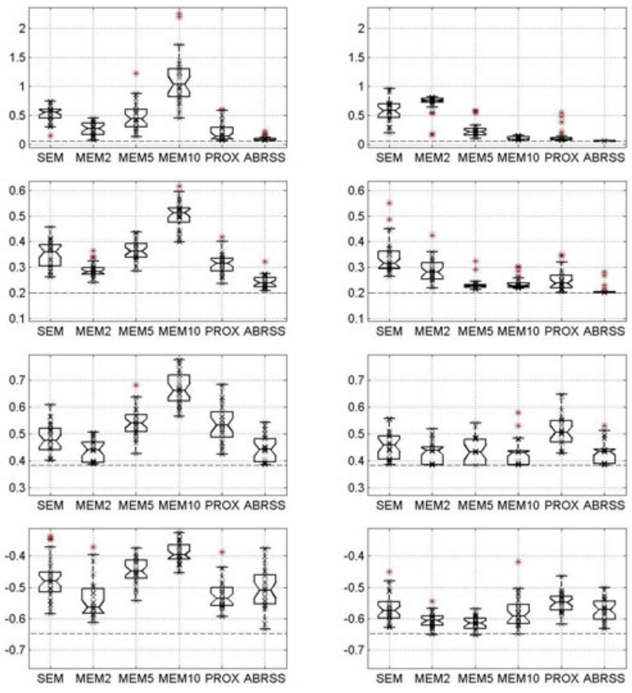


Fig. 6. Solution quality boxplots after 1000 (left) and 5000 (right) evaluations

Table 1. Solution quality data after 1000 (left) and 5000 (right) evaluations

		Quality after 1000 evaluations					Quality after 5000 evaluations				
	Method	Mean	Stddev	Median	Rank Sum	Rank	Mean	Stddev	Median	Rank Sum	Rank
f_1	SEM	0.529	0.144	0.549	3625	5	0.575	0.191	0.577	4180	5
	MEM2	0.263	0.112	0.272	2155	3	0.705	0.162	0.756	4700	6
	MEM5	0.498	0.248	0.436	3294	4	0.241	0.122	0.213	3114	4
	MEM10	1.111	0.421	1.033	4834	6	0.098	0.030	0.082	1807	2
	PROX	0.203	0.158	0.138	1689	2	0.135	0.120	0.083	2024	3
	ABRSS	**0.086**	0.039	**0.069**	**693**	**1**	**0.056**	0.002	**0.055**	**465**	**1**
f_2	SEM	0.348	0.052	0.361	3094	4	0.340	0.067	0.316	4648	6
	MEM2	0.289	0.027	0.283	1768	2	0.288	0.044	0.282	3843	5
	MEM5	0.365	0.040	0.363	3512	5	0.232	0.023	0.228	2099	2
	MEM10	0.504	0.056	0.513	4920	6	0.237	0.023	0.228	2337	3
	PROX	0.314	0.040	0.316	2377	3	0.249	0.038	0.238	2636	4
	ABRSS	**0.243**	0.025	**0.239**	**619**	**1**	**0.208**	0.019	**0.201**	**727**	**1**
f_3	SEM	0.481	0.055	0.475	2214	3	0.457	0.051	0.459	3084	5
	MEM2	**0.437**	0.042	**0.442**	**1252**	**1**	**0.431**	0.042	0.439	2350	3
	MEM5	0.543	0.054	0.540	3389	5	0.435	0.048	0.433	2071	2
	MEM10	0.668	0.062	0.661	4822	6	0.432	0.048	**0.432**	**2062**	**1**
	PROX	0.537	0.071	0.533	3192	4	0.510	0.057	0.507	4150	6
	ABRSS	0.446	0.046	**0.442**	1421	2	0.438	0.044	0.436	2573	4
f_4	SEM	-0.474	0.066	-0.480	2834	4	-0.566	0.047	-0.574	3178	4
	MEM2	**-0.539**	0.062	**-0.564**	**1405**	**1**	-0.605	0.023	-0.606	1804	2
	MEM5	-0.449	0.041	-0.448	3434	5	**-0.614**	0.022	**-0.613**	**1393**	**1**
	MEM10	-0.389	0.033	-0.396	4607	6	-0.578	0.050	-0.592	2701	3
	PROX	-0.523	0.047	-0.535	1772	2	-0.549	0.031	-0.548	4030	6
	ABRSS	-0.504	0.067	-0.509	2238	3	-0.572	0.035	-0.571	3184	5

Interestingly, on the multimodal cases f_3 and f_4, the results show a different picture, and the solution quality after 5000 evaluations of the ABRSS approach ranks 4th and 5th respectively. On f_2 the ABRSS and MEM2 show similar results and both are outperforming the other schemes, but after 5000 evaluations, the MEM2, MEM5, MEM10 and ABRSS scheme show practically no difference in performance. For f_4, the ABRSS is still relatively fast in the early stage, but outperformed by the MEM2 and PROX scheme. However, after 5000 function evaluations, the MEM2 and MEM5 scheme produce clearly better results. From this we can conclude that the ABRSS can be used for fast local optimization, but the sample variations of the MEM2 and MEM5 approach can induce an explorative behavior, favoring more robust peaks on the long run.

4.2 Notes on the Archive Quality

To get an insight in the development of the archive, we consider its growth on a 2D version of test problem f_1. Figure 7 shows, for a run of both the ABRSS scheme and the PROX scheme, the archive after 100, 200, and 300 generations. The small plots inside each plot are magnified versions on the interval $[0, 2]^2$ (i.e., the interval around the optimum). The asset of the archive maintenance scheme can be seen clearly: whereas the PROX method zooms in on a narrow region, the ABRSS scheme considers the whole region of uncertainty around the point on which it zooms in, yielding a locally well-spread set of archive points (and in effect generating more accurate fitness approximations). Although in higher dimensions it will take more time to build up a well-spread archive, the ABRSS scheme, in contrast to the PROX scheme, has the potential to do so.

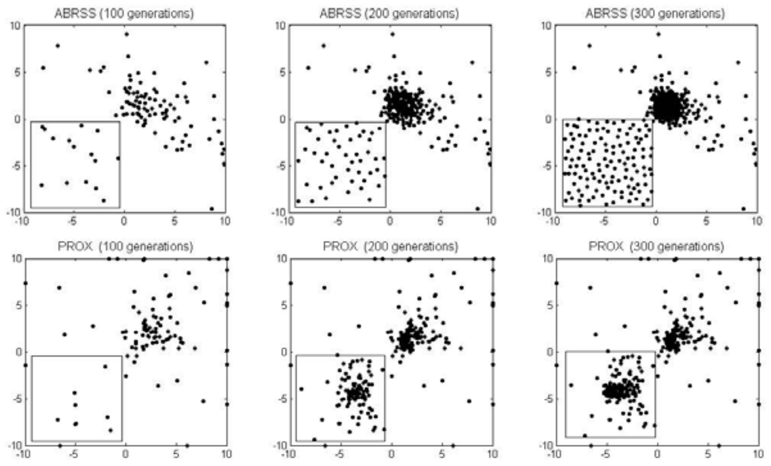


Fig. 7. Archive after 100, 200, and 300 generations of the ABRSS scheme (top row) and PROX scheme (bottom row) on a 2D version of test problem f_1. The zoomed view shows the interval $[0, 2]^2$

5 Conclusion and Outlook

This paper has presented an archive maintenance scheme that can be used within an evaluation scheme for finding robust optima. It extends the scheme presented in [3] by actively enforcing locally well-spread distributions of archive points. The experiments show that this way of archive maintenance can indeed improve the quality of the robust search algorithm. On unimodal landscapes, the ABRSS scheme has shown to yield a fast and accurate search algorithm, outperforming the MEM schemes both in speed and final solution quality. On multimodal landscapes, where each optimum has a different robustness, the ABRSS method seems to lose much power. Hence, it yields a local optimizer that zooms in on one optimum rather than considering multiple robust peaks. A logical topic for future research would be to investigate whether niching approaches could be incorporated to improve this scheme (i.e., apply this scheme multiple times on different peaks and a-posteriori select the most robust peak).

The archive maintenance scheme can also be applied in metamodeling approaches because it enforces well-spread local distributions of archive points that are also beneficial for metamodeling. It will remain open for future research to compare the ABRSS scheme to metamodeling schemes and to study its use within metamodeling schemes.

Acknowledgments. J. Kruisselbrink acknowledges financial support of the Netherlands Organization for Scientific Research (NWO); dossier no. 612.066.618.

References

1. Beyer, H.-G.: Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice. Computer Methods in Applied Mechanics and Engineering 186(2-4), 239–267 (2000)
2. Beyer, H.-G., Sendhoff, B.: Evolution strategies for robust optimization. In: IEEE Congress on Evolutionary Computation, pp. 1346–1353 (2006)
3. Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 119–128. Springer, Heidelberg (1998)
4. Branke, J.: Reducing the sampling variance when searching for robust solutions. In: Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 235–242 (2001)
5. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
6. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 42(1), 55–61 (2000)
7. 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)
8. Paenke, I., Branke, J., Jin, Y.: Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. IEEE Transactions on Evolutionary Computation 10(4), 405–420 (2006)
9. Tsutsui, S.: A comparative study on the effects of adding perturbations to phenotypic parameters in genetic algorithms with a robust solution searching scheme. In: IEEE Systems, Man, and Cybernetics Conference (SMC 1999), pp. 585–591 (1999)
10. Tsutsui, S., Ghosh, A.: Genetic algorithms with a robust solution searching scheme. IEEE Transactions on Evolutionary Computation 1(3), 201–208 (1997)
11. Tsutsui, S., Ghosh, A.: Effects of adding perturbations to phenotypic parameters in genetic algorithms for searching robust solutions. In: Advances in evolutionary computing: theory and applications, pp. 351–365 (2003)
12. Wiesmann, D., Hammel, U., Bäck, T.: Robust design of multilayer optical coatings by means of evolutionary algorithms. IEEE Transactions on Evolutionary Computation 2, 162–167 (1998)

Experimental Supplements to the Theoretical Analysis of Migration in the Island Model*

Jörg Lässig and Dirk Sudholt

International Computer Science Institute, Berkeley, CA 94704, USA

Abstract. In Lässig and Sudholt (GECCO 2010) the first running time analysis of a non-trivial parallel evolutionary algorithm was presented. It was demonstrated for a constructed function that an island model with migration can drastically outperform both panmictic EAs as well as parallel EAs without migration. This work provides additional empirical results that increase our understanding of why and when migration is essential for this function. We provide empirical evidence complementing the theoretical results, investigate the robustness with respect to the choice of the migration interval and compare various migration topologies using statistical tests.

1 Introduction

Evolutionary Algorithms (EAs) are successful strategies for hard optimization problems for which no efficient methods are known. For large-scale applications it is common to use parallel implementations, which have shown to be a successful strategy to speed up the computation in many cases [1,2]. Especially due to current multi-core architectures parallel EAs are highly relevant [3,4].

Despite many wide-spread applications, empirical investigations [5], and a long history of parallel EAs, the theory of these algorithms is not well developed and more fundamental research is needed [5].

Rigorous running time analysis allows to assess the performance of a given algorithm on a given problem or problem class. In Lässig and Sudholt [6] the first running time analysis of a non-trivial parallel evolutionary algorithm was presented. The purpose of this work was to establish an example where, for the first time, parallelization and migration could be proved to outperform panmictic populations and island models without migration. For this artificial problem, called LOLZ, it was proved that with overwhelming probability

– panmictic populations need exponential time to find a global optimum,
– separate subpopulations need exponential time to find a global optimum,
– a simple island model with properly configured migration needs only polynomial time to find a global optimum.

* The authors were supported by postdoctoral fellowships from the German Academic Exchange Service.

These results demonstrated the effectiveness of parallelization and migration from a rigorous theoretical perspective. In this work we take a more empirical view and address several questions that have been left open in [6]. The result for the island model holds for all migration topologies that fulfill a certain expansion property. It is not clear which of the commonly used migration topologies performs best for the problem LOLZ. Also the theoretical result required a very specific choice of the migration interval (i. e., the time between two migrations). Therefore, we consider the robustness of the island model for a broad range of values for the migration interval. Our investigations lead to additional insights in which settings parallelization leads to a drastic speed-up.

In Section 2 the algorithms, the function LOLZ and the theorems from [6] are reviewed. Section 3 then experimentally reproduces the theoretical results. In Section 4 the success rate of different migration intervals is investigated for different migration topologies and in Section 5 these investigations are statistically validated. Section 6 concludes our study.

2 Preliminaries and Previous Work

As our experiments will be heavily based on previous work [6] we first review these results. The (μ+1) EA is a panmictic steady-state algorithm that in each generation selects a parent uniformly at random and generates an offspring by mutation. The offspring replaces one of the worst individuals in the population, unless it is inferior to all individuals in the population.

Algorithm 1. Panmictic (μ+1) EA

```
Let $t := 0$ and initialize $P_0$ with $\mu$ individuals chosen uniformly at random.
repeat
    Choose $x \in P_t$ uniformly at random.
    Create $y$ by flipping each bit in $x$ independently with probability $1/n$.
    Choose $z \in P_t$ with worst fitness uniformly at random.
    if $f(y) \geq f(z)$ then $P_{t+1} = P_t \setminus \{z\} \cup \{y\}$ else $P_{t+1} = P_t$.
    Let $t = t + 1$.
```

To describe the island model let $P = P^1 \,\dot{\cup}\, P^2 \,\dot{\cup} \dots \dot{\cup}\, P^k$ be a partition of the whole population in multiple subpopulations or islands. A migration topology, given by a directed graph with vertices representing islands, describes the neighborhood structure for the islands. Algorithm 2 represents a *parallel EA*, where k subpopulations $P^i, i = 1, 2, \dots, k$ evolve independently as in the (μ+1) EA from Algorithm 1, except for special migration steps. Every τ steps migrants from one island, in this case copies of the island's best individual, are sent to all islands that are connected in the migration topology via a directed edge. The incoming migrants are included into the island using the same selection as in the panmictic (μ+1) EA: for each subpopulation, the received individual of highest fitness replaces a worst individual on the island, unless being inferior to it.

Algorithm 2. Parallel EA with migration

```
Let $t := 0$ and for all $1 \leq i \leq k$ initialize $P_0^i$ uniformly at random.
repeat
    For all $1 \leq i \leq k$ do in parallel
        if $t \bmod \tau = 0$ and $t > 0$ then
            Send an individual with maximum fitness in $P^i$ to all neighbored pop.
            Choose $y^i$ with maximum fitness among all incoming migrants.
        else
            Choose $x^i \in P_t^i$ uniformly at random.
            Create $y^i$ by flipping each bit in $x^i$ independently with probability $1/n$.
        Choose $z^i \in P_t$ with worst fitness uniformly at random.
        if $f(y^i) \geq f(z^i)$ then $P_{t+1}^i = P_t^i \setminus \{z^i\} \cup \{y^i\}$ else $P_{t+1}^i = P_t^i$.
    Let $t = t + 1$.
```

The value τ is called *migration interval.* The special case of $\tau = \infty$, i. e., no migration, is called the *parallel EA with independent subpopulations.* If $\tau < \infty$ and all subpopulations have size 1, this is called the *parallel (1+1) EA with migration* or shortly *island model.*

In [6] the authors introduced the following function.

Definition 1. *For a bit string* $x = x_1 \dots x_n$ *let* $\mathrm{LO}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_i$ *describe the number of leading ones and* $\mathrm{LZ}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} (1 - x_i)$ *describe the number of leading zeros. Let* $z, b, \ell \in \mathbb{N}$ *such that* $b\ell \leq n$ *and* $z < \ell$*. For a bit string* $x = x_1 \dots x_n$ *we abbreviate* $x_{(i-1)\ell+1} \dots x_{i\ell}$ *by* $x^{(i)}$ *and let*

$$\mathrm{LOLZ}_{n,z,b,\ell}(x) = \sum_{i=1}^{b} \prod_{j=1}^{(i-1)\ell} x_j \cdot \left[\mathrm{LO}(x^{(i)}) + \min\left(z, \mathrm{LZ}(x^{(i)})\right) \right].$$

The function is constructed in such a way that an evolutionary algorithm can fix bits from left to right. The bit string is divided into b blocks of length ℓ each. In the first block the algorithm can either collect leading ones or leading zeros. Both decisions lead to an equal gain in fitness for each leading bit. After a threshold of z fixed leading bits has been reached, only leading ones can lead to a larger fitness. Solutions where leading zeros have been gathered represent local optima that are by a Hamming distance of at least z away from all better search points. If z is large enough, it becomes nearly impossible to escape from this local optimum. In case the algorithm has collected leading ones and the first block only consists of ones, the fitness depends on the leading bits in the second block in the same way. Only if the algorithm manages to decide for leading ones in all blocks, a global optimum is reached.

In a panmictic population, the whole population tends to move towards one specific type of prefix. Hence, for each block there is a probability of about 1/2 that the whole population starts gathering leading zeros in the block, thus getting stuck in a local optimum that is hard to overcome. The probability of always making the correct decision is close to 2^{-b}, so it decreases exponentially with the number of blocks. The precise result from [6] reads as follows.

Theorem 1. *Consider the panmictic (μ+1) EA with $\mu \leq cn/(\log n)$ for an arbitrary constant $c > 0$ on $\mathrm{LOLZ}_{n,z,b,\ell}$ with $z = \omega(\log b)$, $b\ell \leq n$, and $z < \ell$. With probability at least $1 - \exp(-\Omega(z)) - 2^{-b}$ the (μ+1) EA does not find a global optimum within $n^{z/3}$ generations.*

Independent subpopulations without communication also tend to get stuck as even multiple independent populations cannot make up for the very small success probability of 2^{-b} if b is not too small.

Theorem 2. *Consider the parallel EA with $s \in \mathbb{N}$ independent subpopulations of size $\mu \leq cn/(\log n)$ each, $c > 0$ an arbitrary constant, on $\mathrm{LOLZ}_{n,z,b,\ell}$ with $z = \omega(\log b)$, $b\ell \leq n$, and $z < \ell$. With probability at least $1 - s\exp(-\Omega(z)) - s2^{-b}$ the (μ+1) EA does not find a global optimum within $n^{z/3}$ generations.*

In sharp contrast to these results, an island model with a well-chosen migration interval and a suitable migration topology was shown to be successful. First, the islands make independent decisions whether to gather leading ones or leading zeros. After some time, the islands that have chosen leading zeros will get stuck and the other islands will exhibit a larger fitness. If a migration happens at this point, the islands that have made the right decision are able to take over islands that are stuck in worse local optima. This way, information about the "good" decision can be spread throughout the islands, so that islands that are stuck in local optima can be re-activated to participate in the search on new blocks. A requirement for this spread of information is that the topology shows a certain degree of expansion. A topology with vertex set V is called *well-expanding* if for every subset $V' \subseteq V$ of size at most $|V'| \leq |V|^{\varepsilon}$ the total number of vertices reachable from any vertex in V' by a directed edge is at least $(2+\varepsilon)|V'|$ for some constant $\varepsilon > 0$. The hypercube and more dense graphs are well-expanding.

In the following result, the migration counter t in Algorithm 2 is initialized with a value of $\tau/2$ instead of 0. This restricts the migrations to take place roughly in the middle of each block.

Theorem 3. *Consider the parallel (1+1) EA with migration on a well-expanding migration topology with $\tau = n^{5/3}$ and μ subpopulations for $\mu \leq \mathrm{poly}(n)$ and $\mu \geq n^{\Omega(1)}$. Let the function $\mathrm{LOLZ}_{n,z,b,\ell}$ be parametrized according to $\ell = 2\tau/n = 2n^{2/3}$, $z = \ell/4 = n^{2/3}/2$, and $b \leq n^{1/6}/16$. If the migration counter starts at $\tau/2 = n^{5/3}/2$ then with probability $1 - \exp(-\Omega(n^{\varepsilon}))$ for some $\varepsilon > 0$ the algorithm finds a global optimum within $O(b\ell n) = O(n^2)$ generations.*

Note that the number of blocks b is very small, unless the problem dimension n is very large. The reason for this restriction is that we require migrations to happen roughly in the middle of each block, for all blocks. This can only be guaranteed if the number of blocks is rather small as the variances for all blocks accumulate quickly. If migration happens too early, before the threshold of z fixed leading bits has been reached, the fitness function does not give information whether gathering leading ones or leading zeros is the better choice. Migration might thus propagate the "wrong" type of individuals to other islands. If migration happens at the end of a block, individuals stemming from the same island tend to have correlated bit values for the next block. Hence, they all tend to make the same decision in the next block. If migration happens earlier, this gives all islands enough time to develop near-independent bit values for bits in the next block. This leads to a diversity that is essential for optimizing LOLZ.

We also remark that in [6] an extension of Theorem 3 to sparse topologies is presented. There, several blocks have to be merged, resulting in an even smaller number of blocks. As the number of blocks would lead to a trivial function for realistic problem dimensions, we decide to use a larger number of blocks in our experimental investigations. This makes LOLZ a more challenging benchmark function for all three algorithms. This also motivates the investigation of different values for the migration interval τ, since it is not clear which τ-value is optimal when faced with a much larger number of blocks.

3 Experimental Reproduction of the Theoretical Results

We first empirically reproduce the theoretical results from [6]. Our basic experimental setup is the optimization of $\text{LOLZ}_{n,z,b,\ell}$ for dimension $n = 1{,}000$ and $z = 10$. As argued before, instead of sticking to the block structure given in Theorem 3 we use $b = 10$ blocks of length $\ell = 100$ each. This means that the whole bit string is used for a fitness evaluation—note that this condition is neither required in the definition of LOLZ nor in Theorem 3. The population size for the panmictic (μ+1) EA was chosen as $\mu = 32$. All parallel EAs use $\mu = 32$ islands with subpopulations of size 1, each.

For the island model we considered four common topologies: a ring, a torus (i. e. a two-dimensional grid with edges wrapping around on all sides) with side lengths 4×8, a hypercube, and the complete graph K_μ. All edges are bidirectional. The migration interval was fixed to $\tau = 50{,}000$, which meets the condition $\ell = 2\tau/n$ from Theorem 3. In accordance with Theorem 3 we initialize the migration counter such that the first migration takes place in generation $\tau/2 = 25{,}000$.

All algorithms were stopped when either the global optimum had been found or each individual had at least z leading zeros in the block currently to be optimized, which means that at least z bits would have to be flipped in one mutation to get out of this local optimum. This event has a negligibly small probability of less than $n^{-z} = 10^{-30}$. The expected time until a local optimum is left is at least 10^{30}, so it makes sense to stop beforehand.

In our first experiments we simulate 1,000 runs for each algorithm and record the success rate, i. e., the fraction of runs that were stopped in a global optimum. We also record the mean number of generations until an algorithm has been stopped as well as the mean best fitness value in the final population. The latter corresponds to the mean number of leading bits that have been fixed in the best individual. The results are shown in Table 1.

For the panmictic (μ+1) EA no run was successful. The mean final fitness value was 114, hence, on average, the algorithm got stuck already in the second block, after only 92,367 generations. Independent subpopulations led to a higher success rate of 0.038 and the algorithm stopped with a higher mean best final

Table 1. Success rate, mean best fitness value after stopping and mean number of generations until runs were stopped for the considered algorithms

Algorithm	success rate	final fitness	# generations
Panmictic (μ+1) EA	0.0	114	92,367
Independent subpopulations	0.038	550	377,472
Island model on ring	0.995	999	709,704
Island model on torus	1.0	1000	655,858
Island model on hypercube	0.651	907	647,339
Island model on K_μ	0.327	651	344,759

fitness of 550. Note that the performance of the parallel EA with independent subpopulations is determined by the best out of $\mu = 32$ independent runs of a (1+1) EA. Here independent runs clearly outperform a panmictic population.

The island model performed far better than the two previous algorithms for all topologies. Surprisingly, the most sparse topologies, the ring and the torus, performed best. With a torus every run was successful and so was almost every run on the ring. For the hypercube the success rate decreased to 0.651 and for the complete graph K_μ it was even only 0.327.

In order to get a more detailed impression of the dynamics within a run, we repeated 100 runs for each algorithm and observed the number of "good" individuals over time. An individual is called good [6] if it has leading ones in its current block (i. e., the first block not completely filled with 1-bits). Note that goodness may change when a new block is reached. Unless we have to deal with correlated bit values, the probability of a good individual being good in the next block is 1/2. Figure 1 shows the number of good individuals over time, averaged over 100 runs for all algorithms. For runs that were stopped with a global optimum, the number of good individuals in the final generation was used to compute the average. We stopped recording once all runs were stopped.

The number of good individuals decreases quickly for the panmictic (μ+1) EA as well as independent subpopulations. For all island models during migration the number of good individuals increases as good individuals take over neighbored

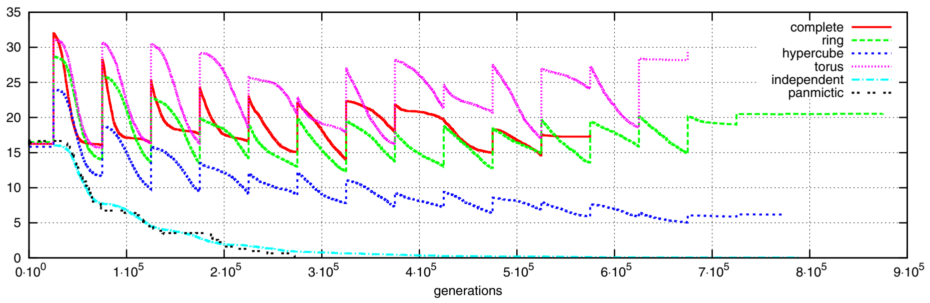


Fig. 1. Average number of good individuals in 100 runs for the panmictic (μ+1) EA, separate subpopulations, and the island model with $\tau = 50{,}000$

islands without good individuals. This effect is particularly pronounced for the torus, while the fluctuations are smallest for the hypercube. The number of good individuals is, in general, higher for the torus than for the hypercube, with the ring and K_μ in between.

4 Comparison of Topologies and Migration Intervals

The reason why dense topologies performed worse than expected from our theoretical results might lie in the different parameter settings concerning block lengths and the number of blocks. Theorem 3 requires few blocks, a rather small block length ℓ, and a very delicate balance between the block length ℓ and the migration interval τ, expressed by $\tau = \ell n/2$. In this setting migration is guaranteed to happen roughly in the middle of each block. With our larger value for b this cannot be guaranteed for two reasons. First, the variances for all blocks accumulate and the total variance becomes too high for large b. The other reason is that once i leading bits are fixed, the optimization is slowed down by a factor of $(1 - 1/n)^{-i}$ as only steps not flipping any of the leading bits can increase the fitness and the expected waiting time for such a step is $(1 - 1/n)^{-i}$. This factor can range from 1 for $i = 0$ to about $e = 2.718\ldots$ for $i = n$ as, unlike Theorem 3, we make use of the whole bit string. It is therefore not clear whether $\tau = \ell n/2$ is the best migration interval or whether slightly larger values are more helpful for optimizing LOLZ.

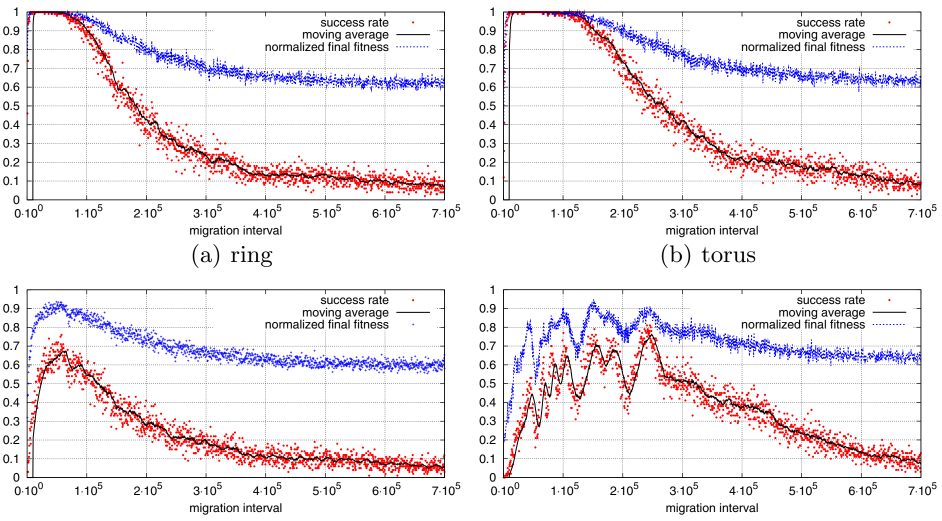
In a series of computationally expensive experiments we recorded the success rate in 100 runs for a broad range of migration intervals, increasing τ in steps of 500 from 500 to 700,000 (CPU time: 319.6h ring, 351.7h torus, 298.3h hypercube, 295.2h K_μ on dual-core Opteron 270 processors with 2.0 GHz, 8GB RAM DDR 400). All other parameters were chosen as in Section 3; in particular, the migration counter was always started at 25,000. The result is shown in Figure 2.

For all topologies the success rate is relatively high for values of τ around $\tau = 50{,}000$ (albeit it is not maximal for K_μ). For larger migration intervals $\tau \geq 250{,}000$ the success rate starts to decrease continuously. This can be explained with the fact that between two migrations the number of good individuals decreases roughly by factors of $1/2$ with each new block.

Sparse topologies like ring and torus seem to be robust w. r. t. the migration interval as for $\tau \leq 100{,}000$ the success rates are close to 1. The curve for K_μ is particularly interesting due to its fluctuations. These fluctuations appear to be random at first sight, but due to the large number of 100 runs and strong correlations between neighbored τ-values the empirical data is, in fact, reliable.

For large migration intervals the success rate for all algorithms is best for K_μ. If a migration takes place (roughly in the middle of the current block) all islands are taken over by good individuals. In contrast, sparser topologies do not utilize the rare migration events that efficiently.

For K_μ we would have expected that the best τ, i. e., the τ-value with the largest success rate is around $\tau = 50{,}000$. Figure 2(d) shows that this is not the case—values around multiples of 50,000 and especially 250,000 seem to be



(a) ring (b) torus

(c) hypercube (d) K_μ

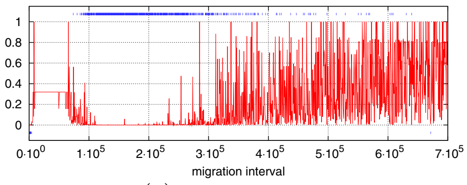
Fig. 2. Success rates, moving averages for 20 data points, and final fitness values for different topologies and migration intervals. The number of runs was 100 for each setting. The final fitness was normalized with a factor 0.001 to fit the interval $[0, 1]$.

better. This makes sense as with $\tau = k \cdot 50{,}000$, $k \in \mathbb{N}$, migration tends to take place in the middle of every k-th block. The reason why large migration intervals are good for K_μ might be that the number of good individuals is very high after each migration. The risk that all these individuals get stuck in one or a few blocks is hence very low, even if no migration takes place in the meantime. If only one individual manages to remain good, the next migration can turn all individuals into good individuals again.

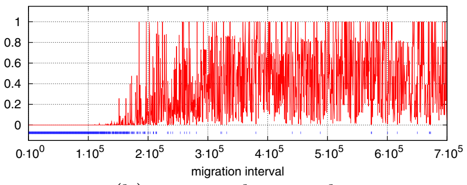
On the other hand, low migration intervals are a disadvantage for K_μ. Each migration has some risk for dense topologies because if it is conducted at the wrong place, all individuals can get stuck immediately. The reason is that if the best individual during such a migration step has leading zeros in its current block—which should be the case with probability about 1/2 unless the threshold of z fixed leading bits has been reached—then the complete population is dominated by individuals of this kind afterwards and the progress of the optimization stops. Hence, up to some point, the success rate is increasing, if the number of these risky migrations is reduced.

5 Statistical Validation

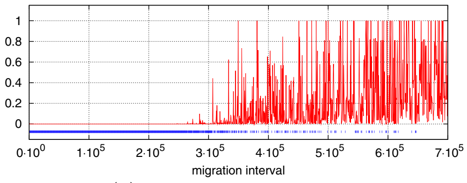
In this section we are aiming at ranking the different migration topologies for different ranges of the migration interval. This is done using statistical tests for success rates. As the underlying probability distributions are binomial distributions, we use t-tests for the comparison of two such distributions as described



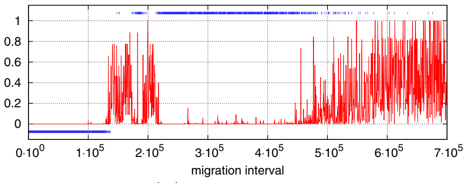
(a) ring vs. torus



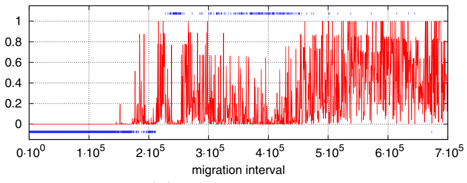
(b) ring vs. hypercube



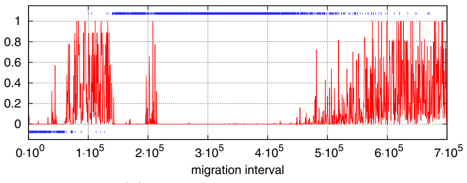
(c) torus vs. hypercube



(d) ring vs. K_μ



(e) torus vs. K_μ



(f) hypercube vs. K_μ

Fig. 3. Plot of the p-values of two-sided t-tests comparing success rates of two algorithms. Values below 0.01 indicate that the underlying success probabilities are different on a significance level of 0.01. These values are marked with bars outside the p-scale. A bar at the bottom indicates that the first algorithm has a higher success probability than the second (judging from which success rate was higher); a bar at the top indicates the opposite.

in [7]. Separate tests are made for each choice of the migration interval and for each pair of migration topologies. Figure 3 shows the resulting p-values of two-sided tests. Low p-values indicate that the two algorithms have different underlying success probabilities. In case p is large no conclusion can be made.

For very large migration intervals $\tau > 450{,}000$ all topologies show a similar behavior as migration hardly ever happens. For almost all other values the torus is significantly better than the hypercube. The same holds for the ring up to $\tau = 200{,}000$. Except for very small values $\tau \leq 4{,}000$ where the ring is better, the torus is better than the ring if τ is roughly in between 100,000 and 300,000.

Comparing K_μ to all other topologies, the ring works better for small migration intervals τ up to about 130,000 generations. For about 220,000 to 450,000 generations the opposite is the case. The torus shows a similar behavior, but the torus outperforms K_μ for a larger range of small τ-values, and for larger values K_μ is not always better than the torus. The performance of the hypercube (compared to K_μ) is similar, but slightly worse—this is consistent with the previous comparisons of ring, torus, and hypercube.

We conclude that the ring is the best choice for very small migration intervals $\tau \leq 4{,}000$. In this range migration tends to be harmful as in situations where less than z leading bits have been fixed, the "wrong" decision might be communicated to neighbored islands. Here the most sparse topology performs best. Contrarily, for large migration intervals, roughly $\tau \geq 200{,}000$, the most expanding topology K_μ performs best as here a good decision can spread to many islands that are stuck in local optima. The torus seems to be the best compromise for τ-values in between. The hypercube was never found to be the best topology in our setting.

6 Conclusion

Complementing theoretical results [6] on the function LOLZ where migration was proven to be essential, our empirical results show that island models with migration every 50,000 generations clearly perform better than panmictic populations and independent subpopulations without migration on LOLZ, in terms of success rates and final fitness values. Sparse migration topologies lead to a better performance than dense topologies. This result is remarkable since Theorem 3 only makes a statement about dense topologies. Our empirical results suggest that a similar or even a stronger statement might hold for sparse topologies.

An extensive study of different migration intervals, along with statistical tests, revealed that sparse migration topologies are better for small migration intervals, while dense topologies are better for larger migration intervals.

References

1. Alba, E.: Parallel evolutionary algorithms can achieve super-linear performance. Information Processing Letters 82(1), 7–13 (2002)
2. He, J., Yao, X.: Analysis of scalable parallel evolutionary algorithms. In: Proceedings of CEC 2006, pp. 120–127 (2006)
3. Cantú-Paz, E., Goldberg, D.E.: On the scalability of parallel genetic algorithms. Evolutionary Computation 7(4), 429–449 (1999)
4. Tomassini, M.: Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time. Springer, Heidelberg (2005)
5. Skolicki, Z., De Jong, K.: The influence of migration sizes and intervals on island models. In: Proceedings of GECCO 2005, pp. 1295–1302. ACM, New York (2005)
6. Lässig, J., Sudholt, D.: The benefit of migration in parallel evolutionary algorithms. In: Proceedings of GECCO 2010, pp. 1105–1112. ACM, New York (2010)
7. Wineberg, M., Christensen, S.: Statistical analysis for evolutionary computation: introduction. In: Companion of GECCO 2009, pp. 2949–2976. ACM, New York (2009)

General Scheme for Analyzing Running Times of Parallel Evolutionary Algorithms

Jörg Lässig and Dirk Sudholt

International Computer Science Institute, Berkeley, CA 94704, USA

Abstract. We present new methods for the running time analysis of parallel evolutionary algorithms with spatially structured populations. These methods are applied to estimate the speed-up gained by parallelization in pseudo-Boolean optimization. The possible speed-up increases with the density of the topology. Surprisingly, even sparse topologies like ring graphs lead to a significant speed-up for many functions while not increasing the total number of function evaluations. We also give practical hints towards choosing the minimum number of processors that gives an optimal speed-up.

Keywords: Parallel evolutionary algorithms, runtime analysis, island model, spatial structures.

1 Introduction

Parallel evolutionary algorithms (EAs) form a popular class of heuristics with many applications to difficult problems [1,2]. Due to the increasing number of CPU cores, exploiting possible speed-ups by parallel computations is nowadays more important than ever. Despite the long history [3] and very active research in this area [4], the theoretical foundation of parallel EAs is still in its infancy. One way of gaining insight into the capabilities and limitations of parallel EAs is by means of rigorous running time analysis [5]. By asymptotic bounds on the running time we can compare different implementations of parallel EAs and assess the speed-up gained by parallelization in a rigorous manner.

In [6] the authors presented the first running time analysis of a parallel evolutionary algorithm with a non-trivial migration topology. It was demonstrated for a constructed problem that migration is essential in the following way. A suitably parametrized island model with migration has a polynomial running time while the same model without migration as well as comparable panmictic populations need exponential time, with overwhelming probability.

In this work we take a broader view and consider the speed-up gained by parallelization for various common pseudo-Boolean functions and function classes of varying difficulty. A general method is presented that can be used to prove upper bounds on the running time of parallel EAs. This method is then tailored towards different spatial structures often used in fine-grained or cellular evolutionary algorithms: ring graphs, torus graphs, and the complete graph. We also show how running time bounds for parallel EAs can be derived from upper bounds for panmictic EAs by the fitness-level method in an automated way. Table 1 summarizes the resulting running time bounds for the considered algorithms and problem classes, where p(1+1) EA is an abbreviation for the parallel (1+1) EA (Algorithm 1) with crossover probability $p_c = 1 - \Omega(1)$.

Table 1. Asymptotic bounds for expected parallel running times $\mathrm{E}(T^{\mathrm{par}})$ and expected sequential running times $\mathrm{E}(T^{\mathrm{seq}})$. Optimal population sizes μ were chosen in cases where the bound on $\mathrm{E}(T^{\mathrm{seq}})$ does not contain μ. For unimodal functions d denotes the number of different function values. See [7] for bounds for the (1+1) EA. The bounds in the last three columns are proven in this work.

		(1+1) EA	p(1+1) EA Ring	p(1+1) EA Grid	p(1+1) EA K_μ
OneMax	$\mathrm{E}(T^{\mathrm{par}})$	$\Theta(n \log n)$	$O(n)$	$O(n)$	$O(n + \frac{n \log n}{\mu})$
			if $\mu \geq \log n$	if $\mu \geq \log n$	$= O(n)$ if $\mu \geq \log n$
	$\mathrm{E}(T^{\mathrm{seq}})$	$\Theta(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n\mu + n \log n)$
LO	$\mathrm{E}(T^{\mathrm{par}})$	$\Theta(n^2)$	$O(n^{3/2})$	$O(n^{4/3})$	$O(n + \frac{n^2}{\mu}) = O(n)$
			if $\mu \geq (en)^{1/2}$	if $\mu \geq (en)^{2/3}$	if $\mu \geq \Omega(n)$
	$\mathrm{E}(T^{\mathrm{seq}})$	$\Theta(n^2)$	$O(n^2)$	$O(n^2)$	$O(n\mu + n^2)$
unimodal	$\mathrm{E}(T^{\mathrm{par}})$	$O(dn)$	$O(dn^{1/2})$	$O(dn^{1/3})$	$O(d + \frac{dn}{\mu}) = O(d)$
			if $\mu \geq (en)^{1/2}$	if $\mu \geq (en)^{2/3}$	if $\mu \geq \Omega(n)$
	$\mathrm{E}(T^{\mathrm{seq}})$	$O(dn)$	$O(dn)$	$O(dn)$	$O(d\mu + dn)$
Jump_k	$\mathrm{E}(T^{\mathrm{par}})$	$\Theta(n^k)$	$O(n^{k/2})$	$O(n^{k/3} + n)$	$O(n + \frac{n^k}{\mu}) = O(n)$
$k \geq 2$			if $\mu \geq (en^k)^{1/2}$	if $\mu \geq (en^k)^{2/3}$	if $\mu \geq \Omega(n^{k-1})$
	$\mathrm{E}(T^{\mathrm{seq}})$	$\Theta(n^k)$	$O(n^k)$	$O(n^k)$	$O(n\mu + n^k)$

2 Preliminaries

We consider the maximization of a pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$. It is easy to adapt the method for minimization. The number of bits is always denoted by n. The following well known example functions have been chosen because they exhibit different probabilities for finding improvements in a typical run of an EA. For a search point $x \in \{0,1\}^n$ write $x = x_1 \dots x_n$, then $\mathrm{OneMax}(x) := \sum_{i=1}^n x_i$ counts the number of ones in x and $\mathrm{LO}(x) := \sum_{i=1}^n \prod_{j=1}^i x_i$ counts the number of leading ones in x. A function is called *unimodal* if every non-optimal search point has a Hamming neighbor (i. e., a point with Hamming distance 1 to it) with strictly larger fitness. Observe that LO is unimodal and for LO every non-optimal point has exactly one Hamming neighbor with a better fitness. For $1 \leq k \leq n$

$$\mathrm{Jump}_k := \begin{cases} k + \sum_{i=1}^n x_i, & \text{if } \sum_{i=1}^n x_i \leq n-k \text{ or } x = 1^n, \\ \sum_{i=1}^n (1 - x_i) & \text{otherwise.} \end{cases}$$

This function has been introduced by Droste, Jansen, and Wegener [7] as a function with tunable difficulty as evolutionary algorithms typically have to perform a jump to overcome a gap by flipping k specific bits.

We are interested in the following performance measures. First we define the *parallel optimization time* T^{par} as the number of generations until the first global optimum is evaluated. The *sequential optimization time* T^{seq} is defined as the number of function evaluations until the first global optimum is evaluated. In both measures we allow ourselves to neglect the cost of the initialization as this only adds a fixed term to the running times.

Our method for proving upper bounds is related to the fitness-level method [5,7]. The idea is to partition the search space into sets $A_1, \dots, A_m$ called *fitness levels* that are ordered w. r. t. fitness values. We say that an algorithm is in A_i or on level i if the current best individual in the population is in A_i. An evolutionary algorithm where the best fitness value in the population can never decrease (called an *elitist* EA) can only improve the current fitness level. If one can derive lower bounds on the probability of leaving a specific fitness level towards higher levels, this yields an upper bound on the expected running time.

Theorem 1 (Fitness-level method). *For two sets $A, B \subseteq \{0,1\}^n$ and a fitness function f let $A <_f B$ if $f(a) < f(b)$ for all $a \in A$ and all $b \in B$. Partition the search space into non-empty sets $A_1, A_2, \dots, A_m$ such that $A_1 <_f A_2 <_f \cdots <_f A_m$ and A_m only contains global optima. For an elitist EA let s_i be a lower bound on the probability of creating a new offspring in $A_{i+1} \cup \cdots \cup A_m$, provided the population contains a search point in A_i. Then the expected number of iterations of the algorithm to find the optimum is bounded by*

$$\sum_{i=1}^{m-1} \frac{1}{s_i} \, .$$

Note that the method can also be applied to other elitist optimization methods.

In the following we apply the fitness-level method to parallel EAs. For the considered EAs we assume that there is a *topology*, given by a directed graph. Islands represent vertices of the topology and directed edges indicate neighborhoods between the islands. Unless mentioned otherwise we assume that in the migration topology edges can be used in both directions. Our methods for proving upper bounds require that the islands run elitist evolutionary algorithms. All islands create new offspring independently by mutation and/or recombination among

Algorithm 1. Parallel (1+1) EA with crossover

```
For all $1 \le i \le k$ choose $x^i \in \{0,1\}^n$ uniformly at random.
repeat
    For all $1 \le i \le k$ do in parallel
        Create $y^i$ by flipping each bit in $x^i$ with probability $1/n$.
        if $f(y^i) \ge f(x^i)$ then $x^i := y^i$.
        Send copies of $x^i$ as migrants to all neighbored islands.
        Choose $z^i$ with maximum fitness among all incoming migrants.
        With probability $p_c$ replace $z^i$ by a crossover of $z^i$ and $x^i$.
        if $f(z^i) \ge f(x^i)$ then $x^i := z^i$.
```

individuals in the island. Additional recombinations might be performed during migration, with parents from different islands, but the focus of our method is on variations that happen within single islands. In our applications we use a simple (1+1) EA for all islands. In order to include common principles in cellular EAs [2], we generalize the (1+1) EA by allowing (arbitrary) crossover operations during migration, each one happening with a fixed crossover probability p_c (see Algorithm 1).

3 Proving Upper Bounds for Parallel EAs

Now we describe how to prove upper bounds on the running time of parallel EAs. In contrast to panmictic EAs, in an island model several islands might participate in the search for improvements from the current-best fitness level. The number of islands may vary over time according to the spread of information.

The following theorem transfers upper bounds for panmictic EAs derived by the fitness-level method into upper bounds for parallel EAs in a systematic way. This means that the method is not only applicable to present analyses of EAs that use the fitness-level method. It can also be used to transfer any future analyses of panmictic EAs to parallel EAs.

Theorem 2 (Fitness-level method for parallel EAs). *Consider a partition of the search space into fitness levels $A_1 <_f A_2 <_f \cdots <_f A_m$ such that A_m only contains global optima. Let s_i be (a lower bound on) the probability that a fixed island running an elitist EA creates a new offspring in $A_{i+1} \cup \cdots \cup A_m$, provided the island contains a search point in A_i. Let μ_t for $t \in \mathbb{N}$ denote (a lower bound on) the number of islands that have discovered an individual in $A_i \cup \cdots \cup A_m$ in the t-th generation after the first island has found such an individual. Then the expected parallel running time of the parallel EA on f is bounded by*

$$E(T^{\text{par}}) \leq \sum_{i=1}^{m-1} \sum_{t=0}^{\infty} (1 - s_i)^{\sum_{j=1}^{t} \mu_j} .$$

Proof. Let T_i denote the random time until the first island finds an individual on a fitness level $i+1, \dots, m$, starting with at least one individual on fitness level i in the whole population. The expected parallel running time can be written as

$$E(T^{\text{par}}) = \sum_{i=1}^{m-1} \mathrm{E}\left(T_i\right) = \sum_{i=1}^{m-1} \sum_{t=1}^{\infty} \mathrm{Prob}\left(T_i \geq t\right) = \sum_{i=1}^{m-1} \sum_{t=0}^{\infty} \mathrm{Prob}\left(T_i \geq t+1\right).$$

A necessary condition for $T_i \geq t+1$ is that during all t generations after the first individual has reached fitness level i all islands are unsuccessful in finding an improvement. In the j-th of these generations there are at least μ_j islands, each being successful with probability at least s_i. Using that the islands create new offspring independently, the probability of all islands being unsuccessful is at most $(1-s_i)^{\mu_j}$. Thus,

$$\sum_{i=1}^{m-1}\sum_{t=0}^{\infty}\text{Prob}\,(T_i \geq t+1) \leq \sum_{i=1}^{m-1}\sum_{t=0}^{\infty}\prod_{j=1}^{t}(1-s_i)^{\mu_j} = \sum_{i=1}^{m-1}\sum_{t=0}^{\infty}(1-s_i)^{\sum_{j=1}^{t}\mu_j} . \square$$

The upper bound from Theorem 2 is very general as it does not restrict the communication among the islands in any way. These aspects are hidden in the definition of the variables μ_t. When looking at one particular fitness level, say level i, we also speak of islands being *informed* if and only if they contain an individual on level i. The variable μ_t then gives the number of informed islands t generations after the first island has been informed.

The spread of information obviously depends on the migration topology, the migration interval, and the selection strategies used to choose migrants that are sent and how migrants are included in the population. The basic method works for all choices of these design aspects. We elaborate on these aspects and then move on to more specific scenarios where we can obtain more concrete results.

With a migration interval of $\tau > 1$ the μ_t-value remains fixed for periods of τ generations. For appropriate t then $\mu_t = \mu_{t+1} = \cdots = \mu_{t+\tau-1}$. As the μ_t-values are non-decreasing with t, the sum of μ-values is at least $\sum_{j=1}^{t}\mu_j \geq \tau\sum_{j=1}^{t/\tau}\mu_{(j-1)\tau+1}$. This implies the following simplified upper bound.

Corollary 1. *For a parallel EA with migration interval τ the bound from Theorem 2 simplifies to*

$$E(T^{\text{par}}) \leq \sum_{i=1}^{m-1}\sum_{t=0}^{\infty}(1-s_i)^{\tau\sum_{j=1}^{t/\tau}\mu_{(j-1)\tau+1}} .$$

The values $\mu_{(j-1)\tau}$ can be estimated like the values μ_j in a setting with $\tau = 1$. In order to keep the presentation simple, in the following applications we only consider the case that $\tau = 1$, i. e., migration happens in every generation. This reflects common principles used in fine-grained or cellular evolutionary algorithms. The following considerations can always be combined with the above arguments to handle migration intervals larger than 1.

Communication between islands might also be probabilistic. Migration can be implemented with stochastic components determining when to migrate, where to send individuals and which individuals to send. Also crossover for incoming migrants can be disruptive. In the light of these effects the numbers μ_t of informed islands are random variables. We state the following upper bounds with respect to a value p_+ that represents a lower bound on the probability that in one generation a specific informed island informs a specific neighbored island.

4 Parallel EAs with Ring Structures

We first consider parallel EAs with sparse topologies. Often ring graphs are used as topologies [2]. Rings can either be unidirectional, in which case there is exactly one directed cycle, or bidirectional, when all edges are undirected. Before proving a method for obtaining general upper bounds on ring graphs, we show the following simple lemma.

Lemma 1. *For all* $0 \leq xy \leq 1$ *it holds* $(1-x)^y \leq 1 - xy/2$.

Proof. Combining $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$ and $e^{-x} \leq 1 - x/2$ for $0 \leq x \leq 1$, we get $(1-x)^y \leq e^{-xy} \leq 1 - xy/2$. □

Theorem 3. *Let* p_+ *be (a lower bound on) the probability that a specific island on fitness level* i *informs a specific neighbor in the topology in one generation. The expected parallel running time of the parallel EA on* f *with an infinite unidirectional or bidirectional ring is bounded by*

$$\sum_{i=1}^{m-1} \frac{3}{(p_+ s_i)^{1/2}} \,.$$

The same holds for finite rings of size at least $\mu \geq \max\{(p_+/s_i)^{1/2}\}$.

Proof. For the current best fitness level let $\xi\,(k)$ denote the random number of generations until at least k islands are informed. For the unidirectional ring we have $\mathrm{E}\,(\xi\,(k)) \leq k/p_+$ since a new island is informed with probability at least p_+. In fact, this argument holds for all strongly connected topologies and in particular for the bidirectional ring. Setting $k := (p_+/s_i)^{1/2}$, after an expected number of $k/p_+ = (p_+ s_i)^{-1/2}$ generations we always have k informed islands. Estimating the remaining expected time for improvements (counting from time $\xi(k)$ on) as in Theorem 2, we have $\mu_t \geq (p_+/s_i)^{1/2}$. Along with Lemma 1,

$$\begin{aligned} \mathrm{E}\,(T^{\mathrm{par}}) &\leq \sum_{i=1}^{m-1} \frac{1}{(p_+ s_i)^{1/2}} + \sum_{t=0}^{\infty} (1 - s_i)^{t \cdot (p_+/s_i)^{1/2}} \\ &\leq \sum_{i=1}^{m-1} \frac{1}{(p_+ s_i)^{1/2}} + \sum_{t=0}^{\infty} \left(1 - \frac{(p_+ s_i)^{1/2}}{2}\right)^t = \sum_{i=1}^{m-1} \frac{3}{(p_+ s_i)^{1/2}} \,. \end{aligned}$$ □

As remarked in the proof, the bound from Theorem 3 holds for arbitrary strongly connected topologies as the unidirectional ring is a worst case for the μ_t-values. Compared to a single island, if $p_+ = \Omega(1)$ in a ring the expected waiting time for every fitness level can be replaced by its square root. We make this precise for concrete functions in the following theorem.

Theorem 4. *The following holds for the parallel (1+1) EA with* $p_c \leq 1 - \Omega(1)$ *on a unidirectional or bidirectional ring:*

- $E(T^{\mathrm{par}}) = O(n)$ *for* OneMax *if* $\mu \geq \sqrt{en}$,
- $E(T^{\mathrm{par}}) = O(d\sqrt{n})$ *for every unimodal function with* d *function values if* $\mu \geq \sqrt{en}$,
- $E(T^{\mathrm{par}}) = O(n + n^{k/2})$ *for* Jump_k *if* $\mu \geq \sqrt{en^k}$.

Proof. An informed island informs a neighbored island in case no crossover is performed, hence $p_+ \geq 1 - p_c$. Let $c := 3/(1-p_c)^{1/2} = O(1)$. For OneMax we choose the canonical partition $A_i := \{x \mid \mathrm{OneMax}(x) = i\}$. The probability of increasing the current fitness from fitness level i is at least $s_i \geq (n-i) \cdot$

$1/(en)$ since there are $n-i$ Hamming neighbors of larger fitness and a specific Hamming neighbor is created with probability at least $1/n \cdot (1-1/n)^{n-1} \geq 1/(en)$. Theorem 3 gives an upper bound of

$$c\sum_{i=0}^{n-1}\sqrt{\frac{en}{n-i}} = c\sqrt{en}\sum_{i=1}^{n}\frac{1}{\sqrt{i}} \leq c\sqrt{en}\int_0^n \frac{1}{\sqrt{i}}\,\mathrm{d}i \leq 2c\sqrt{en}\cdot\sqrt{n} = O(n).$$

For unimodal functions we choose a partition $A_1, \dots, A_d$ where A_i contains all search points with the i-th smallest function value. The probability of improving the fitness from level i is at least $s_i \geq 1/(en)$ because there is at least one search point in the next fitness level which is at Hamming distance one. Theorem 3 gives an upper bound of

$$c\sum_{i=1}^{d-1}\sqrt{en} \leq cd\sqrt{en}.$$

For Jump_k functions and $i < n-k$ the fitness levels A_i are chosen equivalently to OneMax, but to reach the highest level A_{n-k+1} a specific bit string with Hamming distance k has to be created, which has probability at least

$$s_{n-k} \geq \left(\frac{1}{n}\right)^k \cdot \left(1-\frac{1}{n}\right)^{n-k} \geq \left(\frac{1}{n}\right)^k \cdot \left(1-\frac{1}{n}\right)^{n-1} \geq \frac{1}{en^k}\ .$$

Hence, Theorem 3 gives an upper bound of

$$c\sum_{i=1}^{n-k-1}\sqrt{\frac{en}{n-i}} + c\sqrt{en^k} \leq 2c\sqrt{en}\cdot\sqrt{n} + c\sqrt{en^k}. \qquad \square$$

With a slightly different calculation for the highest $n/\log n$ fitness levels we can also show that for OneMax the requirement on the ring size can be relaxed towards $\mu \geq \log n$ while maintaining an expected parallel optimization time of $O(n)$. This results in an optimal sequential running time. The basic idea is to wait until $\log n$ islands are informed and then estimating the remaining expected time with an increased success probability of at least $s_i(\log n)/2$. Details are omitted due to space constraints.

Theorem 5. *Under the conditions of Theorem 4, on* OneMax *with a relaxed condition $\mu \geq \log n$ on the ring size still $E(T^{\text{par}}) = O(n)$.*

5 Parallel EAs with Two-Dimensional Grids and Tori

For two-dimensional grids and tori we adapt Theorem 2 in a similar manner. We also consider applications of the resulting theorem similar to the applications for ring graphs.

Theorem 6. *Let p_+ be defined as in Theorem 3. The expected parallel running time of the parallel EA on f with an infinite two-dimensional grid is bounded by*

$$\sum_{i=1}^{m-1} \frac{10}{p_+^{2/3} s_i^{1/3}} .$$

The same holds for a finite torus of size at least $\mu \geq \max\{(p_+/s_i)^{2/3}\}$.

Proof. For the current best fitness level let $\xi(k)$ denote the random time until at least k islands are informed. Using standard techniques, it is not hard to show that for an infinite grid or a two-dimensional torus of size $\mu \geq k$ the expected time until k islands are informed is bounded by $8/p_+ \cdot \sqrt{k}$.
Setting $k := (p_+/s_i)^{2/3}$, after an expected number of $8/p_+ \cdot \sqrt{k} = 8p_+^{-2/3} \cdot s_i^{-1/3}$ generations we always have k informed islands. Estimating the remaining expected time for improvements as in Theorem 2 and using Lemma 1,

$$\begin{aligned} \mathrm{E}\left(T^{\mathrm{par}}\right) &\leq \sum_{i=1}^{m-1} \frac{8}{p_+^{2/3} s_i^{1/3}} + \sum_{t=0}^{\infty} (1 - s_i)^{t \cdot (p_+/s_i)^{2/3}} \\ &\leq \sum_{i=1}^{m-1} \frac{8}{p_+^{2/3} s_i^{1/3}} + \sum_{t=0}^{\infty} \left(1 - \frac{p_+^{2/3} s_i^{1/3}}{2}\right)^t = \sum_{i=1}^{m-1} \frac{10}{p_+^{2/3} s_i^{1/3}} . \qquad \square \end{aligned}$$

Compared to a single island, in a torus the expected waiting time for every fitness level can be replaced by its third root. This leads to improved upper bounds for unimodal functions and Jump_k.

Theorem 7. *The following holds for the parallel (1+1) EA with $p_c \leq 1 - \Omega(1)$ on an infinite grid or a torus with μ vertices:*

- $E(T^{\mathrm{par}}) = O(n)$ *for* OneMax *if* $\mu \geq \sqrt[3]{en}$,
- $E(T^{\mathrm{par}}) = O(dn^{1/3})$ *for every unimodal function with d function values if* $\mu \geq (en)^{1/3}$,
- $E(T^{\mathrm{par}}) = O(n + n^{k/3})$ *for* Jump_k *if* $\mu \geq e^{1/3} n^{k/3}$.

The proof follows immediately from Theorem 6 by taking over the fitness levels and the s_i-values from Theorem 4. Details are omitted due to space restrictions.

6 Parallel EAs with Complete Topologies

Finally, we consider the densest topology, the complete graph K_μ, as migration structure. Whenever one island finds a new fitness level i then each island will be on fitness level i in the next generation with probability at least p_+ (or on a higher fitness level in case different islands find different improvements).

Theorem 8. *Let p_+ be defined as in Theorem 3. The expected parallel optimization time of a parallel EA on f with a complete topology is*

$$E(T^{\mathrm{par}}) \leq \frac{4m}{p_+} + \frac{4}{\mu} \sum_{i=1}^{m-1} \frac{1}{s_i} .$$

Proof. We estimate the expected time until at least $\mu/2$ islands are informed after an improvement. If more than $\mu/2$ islands are uninformed, the expected number of islands that become informed in one generation is at least $p_+\mu/2$. By standard drift analysis arguments [8] the desired expectation is bounded by $2/p_+$. We apply results from Theorem 2 to estimate the expected remaining optimization time:

$$\sum_{t=0}^{\infty}(1-s_i)^{\mu t/2} = \sum_{t=0}^{\infty}\left((1-s_i)^{\mu/2}\right)^t = \frac{1}{1-(1-s_i)^{\mu/2}} .$$

Now we consider two cases. If $s_i \cdot \mu/2 \leq 1$ by Lemma 1 we have $1-(1-s_i)^{\mu/2} \geq 1-\left(1-\frac{s_i\mu}{4}\right) = \frac{s_i\mu}{4}$. Otherwise, if $s_i \cdot \mu/2 > 1$ we have $1-(1-s_i)^{\mu/2} \geq 1-e^{-s_i\mu/2} \geq 1-\frac{1}{e}$. Thus,

$$\sum_{i=1}^{m-1}\frac{1}{1-(1-s_i)^{\mu/2}} \leq \sum_{i=1}^{m-1}\max\left\{\frac{1}{1-1/e}, \frac{4}{\mu\cdot s_i}\right\} \leq m\cdot\frac{e}{e-1}+\frac{4}{\mu}\sum_{i=1}^{m-1}\frac{1}{s_i} .$$

Adding the expected waiting times until $\mu/2$ islands are informed and observing $e/(e-1)+2/p_+ \leq 4/p_+$ yields the claimed bound. □

The topology leads to a maximal spread of information. In comparison to the previous sections, we obtain the best upper bounds for the considered function classes.

Theorem 9. *Let $\mu \in \mathbb{N}$. The following holds for the expected parallel optimization time of the parallel (1+1) EA with topology K_μ:*

- *$E(T^{\text{par}}) = O(n+(n\log n)/\mu)$ for* OneMax*, which is $O(n)$ if $\mu \geq \log n$,*
- *$E(T^{\text{par}}) = O(d+nd/\mu)$ for every unimodal function with d function values, which is $O(d)$ if $\mu \geq n$,*
- *$E(T^{\text{par}}) = O(n+(n\log n+n^k)/\mu)$ for* Jump$_k$*, which is $O(n)$ if $\mu \geq n^{k-1}$.*

Proof. Let $c := 4/(1-p_c) = O(1)$. We take over the fitness levels from Theorem 4. By Theorem 8,

$$E(T^{\text{par}}) \ \leq\ cn+\frac{4}{\mu}\sum_{i=0}^{n-1}\frac{en}{n-i} \ =\ cn+\frac{4en}{\mu}\sum_{i=1}^{n}\frac{1}{i} \ =\ O(n+n(\log n)/\mu)$$

using that the n-th harmonic number is $O(\log n)$. For unimodal functions

$$E(T^{\text{par}}) \ \leq\ cd+\frac{4}{\mu}\sum_{i=1}^{d-1}en \ =\ cd+\frac{4den}{\mu} \ =\ O(d+nd/\mu) .$$

For Jump$_k$ we have for $E(T^{\text{par}})$ the upper bound

$$cn+\frac{4}{\mu}\left(\sum_{i=1}^{n-k-1}\frac{en}{n-i}+en^k\right) \leq cn+\frac{4en}{\mu}(\log n+n^{k-1}) = O\left(n+n\frac{\log n+n^{k-1}}{\mu}\right). \ \square$$

7 Conclusions

We have provided a new method for the running time analysis of parallel evolutionary algorithms, including applications to a set of well-known and illustrative example functions. Our method provides a way of automatically transforming running time bounds obtained for panmictic EAs via the fitness-level method to parallel EAs. Besides a general theorem, we have provided methods tailored towards specific topologies: complete graphs, rings, and torus graphs. Our techniques are now ready to be applied to further algorithms and problems.

The applications revealed insights which are remarkable in their own right (see Table 1). Compared to upper bounds for a single panmictic island by the fitness-level method, for ring graphs the expected waiting time for an improvement can be replaced by its square root in the parallel optimization time. This leads to speed-up of order $\log n$ for OneMax and of order $\sqrt{n}$ for unimodal functions like LO. On Jump_k the speed-up is even of order $n^{k/2}$. A similar effect is observed for torus graphs where the expected waiting time can be replaced by its third root, i. e., the speed-ups are even stronger here. For the complete graph parallel EAs can decrease the parallel running time also on LO and Jump_k to $O(n)$. In all these results the population size can be chosen in such a way that the total number of function evaluations does not increase, in an asymptotic sense. The "optimal" population sizes have been stated explicitly, therefore giving hints on how to parametrize parallel EAs.

Acknowledgment. The authors were supported by postdoctoral fellowships from the German Academic Exchange Service.

References

1. Nedjah, N., de Macedo Mourelle, L.: Parallel Evolutionary Computations. Springer, Heidelberg (2006)
2. Tomassini, M.: Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time. Springer, Heidelberg (2005)
3. Cantú Paz, E.: A survey of parallel genetic algorithms. Technical report, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana Champaign, Urbana, IL (1997)
4. Rudolph, G.: Takeover time in parallel populations with migration. In: BIOMA 2006, pp. 63–72 (2006)
5. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In: Sarker, R., Yao, X., Mohammadian, M. (eds.) Evolutionary Optimization, pp. 349–369. Kluwer, Dordrecht (2002)
6. Lässig, J., Sudholt, D.: The benefit of migration in parallel evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 1105–1112. ACM, New York (2010)
7. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science 276, 51–81 (2002)
8. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)

Negative Drift in Populations

Per Kristian Lehre

Technical University of Denmark, 2800 Lyngby, Denmark

Abstract. An important step in gaining a better understanding of the stochastic dynamics of evolving populations, is the development of appropriate analytical tools. We present a new drift theorem for populations that allows properties of their long-term behaviour, e. g. the runtime of evolutionary algorithms, to be derived from simple conditions on the one-step behaviour of their variation operators and selection mechanisms.

1 Introduction

Drift analysis is one of the primary mathematical techniques used to estimate the runtime of evolutionary algorithms (EAs) and other randomised search heuristics. Based on Hajek [4], drift analysis was introduced to evolutionary computation by He and Yao [5]. The dynamics of the EA on a fitness function is aggregated into a real-valued stochastic process $X_0, X_1, \ldots,$ by mapping each element of the state space Ω of the EA to a real value using a potential function $g : \Omega \rightarrow \mathbb{R}$. If the potential function is appropriately defined, drift theorems allow properties about the long-term behaviour of the EA to be deduced from conditions on the mean one-step drift of the process, defined as $\mathbf{E}\left[\Delta_t\right]$, where $\Delta_t := X_{t+1} - X_t$. While this approach has proven effective to analyse search heuristics that maintain only a single search point, e. g. simulated annealing [10], and the (1+1) EA [2], there are few applications of drift analysis on search heuristics that maintain several search points, e. g. population-based EAs. Even for simple fitness functions, highly non-trivial potential functions seem necessary to aggregate the state of the population [9]. Other approaches may be needed to analyse population-based search heuristics in more complex scenarios.

We introduce a new drift theorem particularly aimed at analysing population-based search heuristics. An essential feature of the theorem is that the effects of the variation operator and the effects of the selection mechanism are decoupled in the conditions of the theorem, thus alleviating the state aggregation problem. The result applies to any population-based process of the type described in Algorithm 1 below. The algorithm keeps a vector $P_{t\geq 0} \in \Omega^\lambda$ of λ search points. The vector will be referred to as a *population*, and its elements as *individuals*. Given a population P_t, the next population P_{t+1} is generated by sampling and perturbing, according to the variation operator p_{mut}, λ individuals in P_t. An iteration of the of outer loop is called a *generation*. Variation operators are represented as transition matrices $p_{\text{mut}} : \Omega \times \Omega \rightarrow [0, 1]$, where $p_{\text{mut}}(x, y)$ represents the probability of perturbing an individual x into an individual y. For a given transition matrix p_{mut}, we associate a Markov process $X_{t\geq 0}$, where for all $t > 0$, the

1 Population Selection-Variation Algorithm

```
Require: Finite state space $\Omega$, transition matrix $p_{\mathrm{mut}}$, and $P_0 \in \Omega^\lambda$.
1: for $t = 0, 1, 2, \ldots$ until termination condition do
2:    for $i = 1$ to $\lambda$ do
3:       Choose a parent index $I_t(i) \in \{1, \ldots, \lambda\}$, and set $x := P_t(I_t(i))$.
4:       Sample $x'$ according to $p_{\mathrm{mut}}(x)$, and set $P_{t+1}(i) := x'$.
5:    end for
6: end for
```

state transition probability is given by $\mathbf{Pr}[X_{t+1} = x' \mid X_t = x] := p_{\mathrm{mut}}(x, x')$, i.e. $X_{t\geq 0}$ corresponds to a random walk of a single individual in Ω according the variation operator. The selection of individuals are specified by the vector $I_t \in \{1, 2, \ldots, \lambda\}^\lambda$ of indices, where the i-th element $I_t(i)$ represents the "parent" of the i-th individual. The selection mechanism is unspecified, but will typically depend on an objective function on Ω. The sequence of index vectors $I_{t\geq 0}$ is associated with a stochastic process $R_{t\geq 0}$ on $\{0, 1, \ldots, \lambda\}^\lambda$, defined as $R_t(i) := \sum_{j=1}^{\lambda} [I_t(j) = i]$ for all $i, 1 \leq i \leq \lambda$. The i-th element $R_t(i)$ represents the number of times the individual with index i was selected during generation t. The expectation $\mathbf{E}[R_t(i)]$ is called the *reproductive rate* of the i-th individual in generation t.

2 Negative Drift Theorem for Populations

Theorem 1. *Given Alg. 1 with positive transition matrix p_{mut} over a finite state space Ω, and a function $g : \Omega \to \mathbb{N}^+$. Pick two positive integers $a(n)$ and $b(n)$ such that $d(n) := b(n) - a(n) \geq 0$. Let $T(n)$ denote the earliest point in time $t \geq 0$ such that $g(P_t(j)) \leq a(n)$ holds for some $j, 1 \leq j \leq \lambda$. If there are $D(n) \geq 1$, and $\kappa(n) > 0$, and constants $\delta, \delta_2, \delta_3 > 0, \alpha_0 \geq 1$, such that for all $t \geq 0$ and integers i, j, k, and l where $a(n) \leq i \leq b(n)$ and $1 \leq l + k \leq j$, it holds*

1. $\mathbf{E}[R_t(i) \mid a(n) < g(P_t(i)) < b(n)] \leq \alpha_0$ *for all* i, $1 \leq i \leq \lambda$,
2. $\mathbf{E}\left[e^{-\kappa(n)\Delta_t(i)} \mid a(n) < g(X_t) < b(n)\right] < 1/(\alpha_0(1+\delta))$
3. $\mathbf{E}\left[e^{-\kappa(n)(g(X_{t+1})-b(n))} \mid g(X_t) \geq b(n)\right] < D(n)$
4. $\mathbf{Pr}[\Delta_t(i) = -l \wedge \Delta_{t+1}(i-l) = -k] \leq e^{\kappa(n)d(n)(1-\delta_2)}\mathbf{Pr}[\Delta_t(i) = -l-k]$
5. $\mathbf{Pr}[\Delta_t(i) = -j] \leq \mathbf{Pr}[\Delta_t(i-k) = -l] \cdot \delta_3$

where

- $R_t(i) := \sum_{j=1}^{\lambda} [I_t(j) = i]$,
- $X_{t\geq 0}$ *is the Markov process on* Ω *associated with* p_{mut}, *and*
- $\Delta_t(i) := (g(X_{t+1}) - g(X_t) \mid g(X_t) = i)$,

then for all time bounds $L(n) > 0$,

$$\mathbf{Pr}[T(n) \leq L(n) \mid g(P_0) \geq b(n)] = O\left(\lambda L(n)^2 D(n) d(n) e^{-\kappa(n)d(n)\delta_2}\right).$$

The result can be described informally as follows. The theorem assumes a potential function g over a search space Ω, and a goal potential $a(n) \geq 0$. Any search point $x \in \Omega$ with potential $g(x) \leq a(n)$ will be called a *solution*. If the algorithm satisfies the five conditions, then the theorem provides an upper bound on the probability that a solution has been found within a chosen number $L(n)$ of generations. The conditions are w.r.t. the reproductive rate α_0 and the random walk X_t. The first two conditions mean that if the potential of the random walk is close to $a(n)$, then the random walk should have a negative drift towards higher potential values. The requirement on the negative drift is proportional to the reproductive rate. The larger the reproductive rate, the larger negative drift is required for the theorem to hold. The third condition is a milder requirement on the negative drift when the random walk is far from the goal potential. The fourth condition limits the advantage of reducing the potential by a given value during two, instead of one step. The last condition states that the probability of reducing the potential by much, should not be much larger than reducing the potential a little. Note that the drift conditions 2-5 are w.r.t. a random walk of a single individual, and not w.r.t. the population. Hence, these conditions can be verified independently of the selection mechanism and fitness function. For economy of use, it is therefore helpful to derive special versions of the theorem for specific settings of Ω, p_{mut} and g, as will be illustrated in Section 3.

We now explain the proof idea. We focus on the event that an individual x reaches a g-value below $b(n)$, and aim to show that within $L(n)$ generations, all of its ancestors either become extinct, or have drifted back to g-values above $b(n)$. The ancestors will be modelled as a *non-selective family tree*, a concept introduced in Lehre and Yao [7]. The nodes in the family tree correspond to the ancestors of x, where node x is the root node. A path in the family tree is called a *lineage*. We will prune the tree, and only consider the part of the tree that corresponds to individuals with g-values below $b(n)$, i.e. any subtree that is rooted in an individual with g-value above $b(n)$ is removed. The number of times an individual with index i is selected, is given by the random variable $R_i(t)$. Different individuals have different offspring distributions. To simplify the analysis, we consider the family tree as if it had been subject to a modified selection process. Here, the number of times each individual is selected is distributed according to $R_{i^*}(t)$, where i^* is the index of the individual with highest reproductive rate. Thus, each individual will be selected as often as it would have been, had it been the individual with highest reproductive rate. Assuming condition 1, each member of the family tree will on average be selected α_0 times. A consequence of modifying the selection process is that the family tree grows quicker than the real family tree. And as there is no selective differences, each lineage corresponds to a random walk.

The proof consist of two parts. The first part provides an upper bound on the number of different lineages in the family tree. The second part provides an upper bound on the probability that a given lineage of length $L(n)$ will reach a solution. The final result is obtained by combining these two parts using a union bound. We start with the second part, and apply Hajek's drift theorem [4].

Theorem 2 (Hajek [4]). *Let $X_0, X_1, \dots$ be the random variables describing a Markov process over the state space Ω, and $g : \Omega \to \mathbb{R}_0^+$ a function that assigns to each state a non-negative real number. Pick two real numbers $a(n)$ and $b(n)$ which depend on a parameter n such that $0 \leq a(n) < b(n)$ holds. Let the random variable $T(n)$ denote the earliest point in time $t \geq 0$ that satisfies $g(X_t) \leq a(n)$. If there are $\kappa(n) > 0$ and $p(n) > 0$ such that for all $t \geq 0$, the condition*

$$\mathbf{E}\left[e^{-\kappa(n)(g(X_{t+1})-g(X_t))} \mid a(n) < g(X_t) < b(n)\right] \leq 1 - 1/p(n)$$

holds, then for all time bounds $L(n) \geq 0$

$$\mathbf{Pr}\left[T(n) \leq L(n) \mid g(X_0) \geq b(n)\right] \ \leq\ e^{\kappa(n)(a(n)-b(n))} \cdot L(n) \cdot D(n) \cdot p(n),$$

where $D(n) := \max\left\{1, \mathbf{E}\left[e^{-\kappa(g(X_{t+1})-b(n))} \mid b(n) \leq g(X_t)\right]\right\}$.

This theorem will be applied later in the proof of Theorem 1. However, note that if the conditions in Theorem 1 hold, then the conditions in Theorem 2 hold for the Markov process associated with the transition matrix p_{mut}.

Knowing that a single lineage will not find the optimum within polynomial time, we now estimate the number of different lineages in the family tree. The number of lineages is trivially bounded by the number of family tree members, which can be analysed using *multi-type branching processes.*

Definition 1 (Multi-Type Branching Process [3]). *A multi-type branching process with d types is a Markov process $Z_0, Z_1, Z_2, \dots$ which for all $t \geq 0$, is given by $Z_{t+1} := \sum_{j=1}^{d} \sum_{i=1}^{Z_{tj}} \xi_i^{(j)}$, where for all j, $\xi_i^{(j)} \in \mathbb{N}_0^d$ are i.i.d. random vectors having expectation $\mathbf{E}\left[\xi^{(j)}\right] =: (m_{j1}, m_{j2}, \dots, m_{jd})^{\intercal}$. The associated matrix $M := (m_{hj})_{d\times d}$ is called the* mean matrix *of the process.*

A multi-type branching process can be thought of as a population of individuals of d types. The vector component Z_{tj} represents the number of individuals of type j in generation t. An individual survives one generation, during which it may produce some offspring. The offspring produced by an individual depends on its type j, and is given by an independent random vector $\xi^{(j)}$, where the vector component $\xi_i^{(j)}$ is the number of offspring of type i. Each entry m_{hj} in the mean matrix represents the expected number of offspring a type h-individual will have of type j-individuals. The expectation of a multi-type branching process can be calculated from its mean matrix by $\mathbf{E}\left[Z_t\right]^{\intercal} = \mathbf{E}\left[\mathbf{E}\left[Z_t \mid Z_{t-1}\right]\right]^{\intercal} = \mathbf{E}\left[Z_{t-1}\right]^{\intercal} M = \cdots = \mathbf{E}\left[Z_0\right]^{\intercal} M^t$. Matrix powers M^t of irreducible matrices can be determined using the Perron-Frobenius theorem, where irreducibility is defined as follows.

Definition 2 (Irreducible matrix [11]). *A $d \times d$ non-negative matrix M is* irreducible *if for every pair i, j of its index set, there exists a positive integer t such that $m_{ij}^{(t)} > 0$, where $m_{ij}^{(t)}$ are the elements of the t-th matrix power M^t.*

Note that positive matrices are irreducible. The following statement of the Perron-Frobenius theorem is taken from [3].

Theorem 3 (Perron-Frobenius). *If M is an irreducible matrix with non-negative elements, then it has a unique positive eigenvalue ρ, called the* Perron root *of M, that is greater in absolute value than any other eigenvalue. All elements of the left and right eigenvectors $u = (u_1, ..., u_d)^\intercal$ and $v = (v_1, ..., v_d)^\intercal$ that correspond to ρ can be chosen positive and such that $\sum_{k=1}^{d} u_k = 1$ and $\sum_{k=1}^{d} u_k v_k = 1$. Also, $M^n = \rho^n A + B^n$, where $A = (v_i u_j)_{i,j=1}^{d}$ and B are matrices where $AB = BA = 0$, and there are constants $\rho_1 \in (0, \rho)$ and $C > 0$ such that none of the elements of the matrix B^n exceeds $C\rho_1^n$.*

Hence, the asymptotics of the matrix power M^t depends primarily on the Perron root. This can be used to bound the expected number of descendants from a single individual of type h.

Lemma 1. *Let $Z_0, Z_1, ...$ be a multi-type branching process with irreducible mean matrix $M = (m_{ij})_{d\times d}$ and Perron root $\rho < 1$ with corresponding right eigenvector v. The number of descendants L_t of the initial individual after $t > 0$ generations satisfies $\mathbf{E}[L_t \mid Z_0 = e_h] \leq \frac{\rho}{1-\rho} \cdot \frac{v_h}{v^*}$, where $e_h, 1 \leq h \leq d$, denote the standard basis vectors, and $v^* := \min_{1\leq i\leq d} v_i$.*

Proof. The proof follows [3, p. 122]. By Theorem 3, matrix M has a unique largest eigenvalue ρ, and all the elements of the corresponding right eigenvector v are positive, implying $v^* > 0$. By using that $v_j \geq v^*$ for all j, we get $\mathbf{E}[L_t \mid Z_0 = e_h] \leq \frac{1}{v^*} \sum_{r=1}^{t} \sum_{j=1}^{d} \mathbf{E}[Z_{rj} v_j \mid Z_0 = e_h]$. As seen above, the expectation on the right hand side can be expressed as $\mathbf{E}[Z_r \mid Z_0 = e_h]^\intercal = \mathbf{E}[Z_0 \mid Z_0 = e_h]^\intercal M^r$. Additionally, by taking into account the starting conditions, $Z_{0h} = 1$ and $Z_{0j} = 0$, for all indices $j \neq h$, this simplifies further to $\frac{1}{v^*} \sum_{r=1}^{t} \sum_{j=1}^{d} \sum_{i=1}^{d} \mathbf{E}\left[Z_{0i} v_j m_{ij}^{(r)} \mid Z_0 = e_h\right] = \frac{1}{v^*} \sum_{r=1}^{t} \sum_{j=1}^{d} v_j m_{hj}^{(r)}$. Finally, by iterating $M^r v = M^{r-1}(Mv) = \rho M^{r-1} v$, which on coordinate form gives $\sum_{j=1}^{d} v_j m_{hj}^{(r)} = \rho^r v_h$, one obtains the final bound $\frac{v_h}{v^*} \sum_{r=1}^{t} \rho^r \leq \frac{\rho}{1-\rho} \cdot \frac{v_h}{v^*}$. □

We formalise the non-selective family tree as a multi-type branching process. Each family tree member corresponds to an individual in the branching process. We have $d(n) := b(n) - a(n)$ types, and the type of an individual x is given by $g(x) - a(n)$. Each family tree member is selected in expectation α_0 times per generation, so an individual of type i, will in expectation have $\alpha_0 p_{ij}$ offspring of type j, where p_{ij} is the probability that the the variation operator produces an offspring of type j from an individual of type i.

Definition 3 (Mean Matrix of EA). *Given Algorithm 1 with reproductive rate α_0, and two integers $0 \leq a(n) < b(n), d(n) := b(n) - a(n)$. The* associated $d(n) \times d(n)$ mean matrix *is defined as $m_{ij} := \alpha_0(p_{ij} + 1/d(n)^2)$ if $j > i$, and $m_{ij} := \alpha_0 p_{ij}$ if $j \leq i$, where $p_{ij} := \mathbf{Pr}[\Delta_t(i + a(n)) = j - i]$.*

The extra term $\alpha_0/d(n)^2$ in the mean matrix is added for technical reasons, and will lead to overestimation of the survival probability. If the Markov chain p_{mut} limited to the set of states x in Ω where $g(x) \leq b(n)$, is irreducible, then the mean matrix is irreducible. To apply Lemma 1, we consider next the Perron root and the associated eigenvector.

Lemma 2. *If Algorithm 1 satisfies the conditions in Theorem 1, then the associated mean matrix M has Perron root $\rho(M) \leq 1/(1+\delta)$ for a constant $\delta > 0$.*

Proof. The Frobenius bound for the Perron root of a matrix M states that $\rho(M) \leq \max_i \sum_j m_{ij}$ [6]. However, when applied directly to our matrix, this bound is insufficient for our purposes. Instead, we can consider the transformation SMS^{-1}, for the invertible matrix $S := \text{diag}(e^{\kappa}, e^{2\kappa}, ..., e^{d(n)\kappa})$. Note that $\det(SAS^{-1}) = \det(S)$ for any $d(n) \times d(n)$ matrix A. So if ρ is an eigenvalue of M, then $0 = \det(M - \rho I) = \det(S(M - \rho I)S^{-1}) = \det(SMS^{-1} - \rho I)$, and ρ must also be an eigenvalue of SMS^{-1}. It follows that $\rho(M) = \rho(SMS^{-1})$. By using the Frobenius bound along the rows of matrix SMS^{-1}, which has elements $(SMS^{-1})_{ij} = m_{ij}e^{-\kappa(j-i)}$, we can bound $\rho(M)$ for large $d(n)$ by

$$\begin{aligned}\rho(SMS^{-1}) &\leq \max_{1\leq i\leq d(n)} \alpha_0 \sum_{j=1}^{d(n)} p_{ij}e^{-\kappa(j-i)} + \sum_{j=i+1}^{d(n)} \frac{\alpha_0}{e^{\kappa(j-i)}d(n)^2} \\ &\leq \max_{1\leq i\leq d(n)} \alpha_0 \sum_{j=-\infty}^{\infty} \mathbf{Pr}\left[\Delta_t(i+a(n)) = j-i\right] e^{-\kappa(j-i)} + \frac{\alpha_0}{d(n)} \\ &= \max_{1\leq i\leq d(n)} \alpha_0 \mathbf{E}\left[e^{-\kappa\Delta_t(i+a(n))}\right] + \frac{\alpha_0}{d(n)} \leq \frac{1}{1+\delta/2}.\end{aligned}$$

□

Lemma 3. *Let M be the mean matrix associated with Algorithm 1, v the right eigenvector corresponding to the Perron root of M, and $v_* := \min_i v_i$ the minimal component of this eigenvector. If the conditions of Theorem 1 are satisfied, then it holds for all indices $h, 1 \leq h \leq d(n)$, that $p_{d(n)h}\frac{v_h}{v^*} \leq \delta_3 e^{\kappa d(n)(1-\delta_2)}$.*

Proof. Minc's bound for the principal ratio of positive matrices [8] can be generalised as follows, $\frac{v_h}{v^*} = \max_k \frac{v_h}{v_k} = \max_k \frac{\rho v_h}{\rho v_k} = \max_k \frac{\sum_j m_{hj}v_j}{\sum_j m_{kj}v_j} \leq \max_{k,j} \frac{m_{hj}}{m_{kj}}$. It now suffices to bound the ratio $p_{d(n)h}m_{hj}/m_{kj}$ for all values of h, j and k. In the case where $k \leq j$, we have $p_{d(n)h}m_{hj}/m_{kj} \leq m_{hj}d(n)^2 \leq 2d(n)^2$. In the case where $h \leq j < k$, condition 5 implies that $p_{d(n)h} \leq \delta_3 p_{kj}$, so $p_{d(n)h}m_{hj}/m_{kj} \leq m_{hj}\delta_3 \leq 2\delta_3$. Finally, when $j < k$ and $j < h$, condition 4 and 5 imply that

$$\frac{p_{d(n)h}m_{hj}}{m_{kj}} = \frac{p_{d(n)h}p_{hj}}{p_{kj}} \leq \frac{\delta_3 p_{d(n)h}p_{hj}}{p_{d(n)j}} \leq \delta_3 e^{\kappa(n)d(n)(1-\delta_2)}$$ □

Proof (of Theorem 1). Consider the event that an individual x with g-value higher than $b(n)$ obtains an offspring with g-value less than $b(n)$. We model the non-selective family tree corresponding to individual x pruned to g-values in the interval $b(n)$ to $a(n)$, as a multi-type branching process with $d(n)$ types and mean matrix given by Definition 3. Any search point with g-value less than $a(n)$ is called a *solution*. To bound the probability q that a given lineage of length at most $L(n)$ reaches a solution, we apply Theorem 2 with the parameter $p(n) := \alpha_0(1+\delta)/(\alpha_0(1+\delta)-1)$, yielding $q = O(L(n)D(n)e^{-\kappa(n)d(n)})$. If there are k lineages in the family tree, then by union bound, the probability that any lineage reaches a solution is less than kq. The number of lineages is less than the number L of family tree members. Hence, the probability that the family tree contains a solution is less than $\sum_{k=1}^{\infty} \mathbf{Pr}[L=k]\, kq = q\mathbf{E}[L] \leq q\sum_{h=1}^{d(n)} p_{d(n)h}\mathbf{E}[L \mid Z_0 = e_h]$. By applying Lemmas 1, 2, and 3, this is no more than $q\sum_{h=1}^{d(n)} \frac{\rho}{1-\rho}\frac{p_{d(n)h}v_h}{v^*} = O(qd(n)e^{\kappa(n)d(n)(1-\delta_2)})$. Finally, by noting that there can be no more than $\lambda L(n)$ such family trees within $L(n)$ generations, the probability that any individual within the first $L(n)$ generations is a solution, is by union bound $O(\lambda L(n)^2 D(n)d(n)e^{-\kappa(n)d(n)\delta_2})$. □

3 Applications to Evolutionary Algorithms

We now derive a simplified variant of Theorem 1 tailored to runtime analysis of EAs on Pseudo-boolean functions. We consider bitwise mutation $p_{\text{bit}}(x,y) = (\frac{\chi}{n})^{H(x,y)}(1-\frac{\chi}{n})^{n-H(x,y)}$, where H denotes Hamming-distance, and the constant parameter $\chi > 0$ determines the mutation rate. The potential function g is defined as the Hamming distance to some search point $x^* \in \{0,1\}^n$.

Theorem 4. *Given Algorithm 1 on $\Omega = \{0,1\}^n$ with transition matrix p_{bit}, mutation rate χ, and population size $\lambda = \text{poly}(n)$. Let $a(n)$ and $b(n)$ be positive integers s.t. $b(n) \leq n/\chi$ and $d(n) := b(n) - a(n) = \omega(\ln n)$. For an $x^* \in \{0,1\}^n$, let $T(n)$ be the smallest $t \geq 0$, s.t. $H(P_t(j), x^*) \leq a(n)$ for some $j, 1 \leq j \leq \lambda$. Let $R_t(i) := \sum_{j=1}^{\lambda}[I_t(j) = i]$. If there are constants $\alpha_0 \geq 1$ and $\delta > 0$ s.t.*

1. $\mathbf{E}[R_t(i) \mid a(n) < H(P_t(i), x^*) < b(n)] \leq \alpha_0,$ *for all* $i, 1 \leq i \leq \lambda,$
2. $\psi := \ln(\alpha_0)/\chi + \delta < 1,$ *and*
3. $\dfrac{b(n)}{n} < \min\left\{ \dfrac{1}{5}, \dfrac{1}{2} - \dfrac{1}{2}\sqrt{\psi(2-\psi)} \right\},$

then there exists a constant $c > 0$ such that $\mathbf{Pr}\left[T(n) \leq e^{cd(n)}\right] \leq e^{-\Omega(d(n))}$.

Proof. We apply Theorem 1 over the interval $[a(n), b(n)]$, where the distance function is defined as $g(x) := H(x, x^*)$. W.l.o.g., we assume that $x^* = 1^n$. The *first condition* holds immediately. For the *second and third conditions*, note that $\mathbf{E}\left[e^{-\kappa\Delta_t(i)}\right] = M_{\Delta_t(i)}(-\kappa)$, where $M_{\Delta_t(i)}$ is the moment-generating function (m.g.f.) for the drift $\Delta_t(i) := (g(X_{t+1}) - g(X_t) \mid g(X_t) = i)$. The drift can be expressed as the sum of two, independent random variables $\Delta_t(i) = \Delta_t^+(i) - \Delta_t^-(i)$, where $\Delta_t^+(i)$ is the number of 1-bits that are flipped into 0-bits, and $\Delta_t^-(i)$ is the number of 0-bits that are flipped into 1-bits. These variables are binomially distributed. A binomially distributed random variable X with parameters n and p has m.g.f. $M_X(t) = (1-p+pe^t)^n$. Setting $\kappa = \ln(2+\varepsilon+\delta_2)$, where ε and δ_2 are constants that will be determined later, the positive drift component is

$$M_{\Delta_t^+(i)}(-\kappa) = \left(1 - \frac{\chi}{n} + \frac{\chi}{n(2+\varepsilon+\delta_2)}\right)^{n-i} \leq \exp\left(-\frac{(1-\frac{i}{n})(1+\varepsilon+\delta_2)\chi}{(2+\varepsilon+\delta_2)}\right),$$

and the negative drift component is

$$M_{\Delta_t^-(i)}(\kappa) = \left(1 - \frac{\chi}{n} + \frac{\chi(2+\varepsilon+\delta_2)}{n}\right)^i \leq \exp\left(\frac{i(1+\varepsilon+\delta_2)\chi}{n}\right).$$

The random variables $\Delta_t^+(i)$ and $\Delta_t^-(i)$ are independent, so the m.g.f. of the drift is given by the product $M_{\Delta_t(i)}(t) = M_{\Delta_t^+(i)}(t)M_{\Delta_t^-(i)}(-t)$. Hence, by taking into account that $i < b(n)$, we get

$$M_{\Delta_t(i)}(t) \leq \exp\left(\frac{(1+\varepsilon+\delta_2)\chi}{2+\varepsilon+\delta_2}\left(\frac{b(n)(3+\varepsilon+\delta_2)}{n} - 1\right)\right).$$

This bound holds for all $\varepsilon > 0$. We would like to maximise $b(n)$ with respect to ε, subject to condition 2 of Theorem 1, i.e. for the constant $\delta' := e^{\delta\chi} - 1$, we would like to maximise the right hand side of the inequality

$$b(n) \leq \frac{n}{3+\varepsilon+\delta_2}\left(1 - \frac{(2+\varepsilon+\delta_2)\ln(\alpha_0(1+\delta'))}{(1+\varepsilon+\delta_2)\chi}\right).$$

We choose $\varepsilon = \frac{\sqrt{\psi(2-\psi)}+(2+\delta_2)\psi-1-\delta_2}{1-\psi}$. By adjusting parameter δ, one can ensure that $1/5 < \psi < 1$, and hence that $\varepsilon > 0$ for an appropriate choice of δ_2. This choice of ε gives $\frac{b(n)}{n} < \min\{\frac{1}{5}, \frac{1}{2} - \frac{1}{2}\sqrt{\psi(2-\psi)}\}$. The second condition of Theorem 1 is therefore satisfied for the parameter δ'. For the *third condition*, it is sufficient to use the upper bound $\mathbf{E}\left[e^{-\kappa(n)(g(X_{t+1})-b(n))} \mid g(X_t) > b(n)\right] \leq \mathbf{E}\left[e^{-\kappa\Delta_t(i)} \mid g(X_t) > b(n)\right]$ and observe that the upper bound on the m.g.f. of the drift is bounded from above by a constant $D(n) = O(1)$ for all i. For the *fourth condition*, it holds for all h, j, k where $1 \leq k < j < h \leq d(n)$ that the ratio $p_{hj}p_{jk}/p_{hk}$ is no more than

$$\frac{\binom{h+a(n)}{h-j}\left(\frac{\chi}{n}\right)^{h-j}\binom{j+a(n)}{j-k}\left(\frac{\chi}{n}\right)^{j-k}}{\binom{h+a(n)}{h-k}\left(\frac{\chi}{n}\right)^{h-k}\left(1-\frac{\chi}{n}\right)^{n-h+k}} \leq e^{\chi}\binom{h-k}{h-j} \leq e^{\chi}2^{h-k} \leq e^{\kappa(n)(1-\delta_2')d(n)},$$

where the final inequality holds for some constant δ_2' because $h - k \leq d(n)$, and $\kappa(n) > \ln 2$. Thus the fourth condition is satisfied. Finally, for the *fifth condition*, it holds for all integers i, j, k and l where $1 \leq l \leq k \leq j < i \leq d(n)$ that

$$\begin{aligned}\frac{p_{il}}{p_{jk}} &\leq \frac{\binom{i+a(n)}{i-l}\left(\frac{\chi}{n}\right)^{i-l}}{\binom{j+a(n)}{j-k}\left(\frac{\chi}{n}\right)^{j-k}\left(1-\frac{\chi}{n}\right)^{n-(j-k)}} \leq \frac{\binom{i+a(n)}{i-j}\binom{j+a(n)}{j-k}\binom{k+a(n)}{k-l}\left(\frac{\chi}{n}\right)^{i-l}}{\binom{j+a(n)}{j-k}\left(\frac{\chi}{n}\right)^{j-k}e^{-\chi}} \\ &= \binom{i+a(n)}{i-j}\left(\frac{\chi}{n}\right)^{i-j}\binom{k+a(n)}{k-l}\left(\frac{\chi}{n}\right)^{k-l}e^{\chi} \leq e^{\chi}\end{aligned}$$

where the last inequality holds because $\binom{m}{k}(\frac{\chi}{n})^k \leq (\frac{m\chi}{n})^k \leq 1$ for any $m \leq n/\chi$ and $k \leq m$. Because of the conditions $d(n) = \omega(\ln n)$ and $\lambda = \text{poly}(n)$, we have $\lambda D(n)d(n) \leq e^{c'd(n)}$ for any constant $c' > 0$, when n is sufficiently large. By setting $L(n) = e^{cd(n)}$ and choosing c and c' sufficiently small, the theorem now follows from Theorem 1. □

Table 1. Parameter settings where non-elitist EAs with bit-wise mutation rate χ/n are ineffective (cf. Corollary 1)

Selection mechanism	Parameter settings
Linear ranking selection	$\eta < e^\chi$
k-tournament selection	$k < e^\chi$
(μ,λ)-selection	$\lambda < \mu e^\chi$
Any in cellular EAs	$\Delta(G) < e^\chi$

Finally, we give some example applications. We first consider Jump_m, which for any $m, 1 \leq m < n$, is defined as $\text{Jump}_m(x) = |x|_1$ if $|x|_1 < n - m$ or $|x|_1 = n$, and $\text{Jump}_m(x) = 0$ otherwise, where $|x|_1 := \sum_{i=1}^n x_i$. Jump_m has a local optimum separated from the global optimum by a gap of size m. The EA must either jump the gap, or make a random walk in the gap. We consider tournament selection, but the theorem can easily be generalised to other selection mechanisms.

Theorem 5. *For any $m < n(1-\varepsilon)/5, 0 < \varepsilon < 1$, with $m = \omega(\ln n)$, the probability that a non-elitist EA using k-tournament selection, $k \geq 1$, population size $\lambda = \text{poly}(n)$, and bitwise mutation rate $\chi/n \leq 1/m$, optimises Jump_m within e^{cm} generations is $e^{-\Omega(m)}$, for some constant $c > 0$.*

Proof. We apply Theorem 4, with the parameters $a(n) = 0$, $b(n) = m, \alpha_0 = 1$, and optimum $x^* = 1^n$. Individuals with no more than m 0-bits, which we call *gap-individuals*, never win tournaments containing individuals with less than m 0-bits. Optimistically assuming that all individuals are gap-individuals, the individuals are selected uniformly at random. Hence, the first condition holds. We have $\psi = \delta$, so the second condition holds for any $\delta, 0 < \delta < 1$. By assumption, $n/5 > m$, and choosing δ sufficiently small also gives $(n/2)(1 - \sqrt{\psi(2-\psi)}) > m$, so the third condition also holds. □

The function $\text{Needle}(x) := \prod_{i=1}^n x_i$ can be analysed analogously to Jump_m.

Theorem 6. *The probability that a non-elitist EA using k-tournament selection, $k \geq 1$, population size $\lambda = \text{poly}(n)$, and bitwise mutation, optimises* Needle *within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

We now consider scenarios where the selective pressure is too low.

Corollary 1. *The probability that a non-elitist EA with population size $\lambda = \text{poly}(n)$, bitwise mutation rate χ/n, and maximal reproductive rate bounded by $\alpha_0 < e^\chi - \delta$, for a constant $\delta > 0$, optimises any function with a polynomial number of optima within e^{cn} generations is $e^{-\Omega(n)}$, for some constant $c > 0$.*

Proof. By Theorem 4, choosing $a(n) = 0$ and $b(n) = c'n$, $c' > 0$ sufficiently small, a given optimum is found in e^{cn} generations with probability $e^{-\Omega(n)}$. By union bound, the probability that any of $r = \text{poly}(n)$ optima is found within e^{cn} generations is $re^{-\Omega(n)} = e^{-\Omega(n)}$. □

A larger reproductive rate than e^{χ} is therefore necessary for a non-elitist EA to be effective. Table 1 summarises parameter settings in some common selection mechanisms that render non-selective EAs ineffective. The parameter $\eta \in [1,2]$ in linear ranking selection is directly related to the reproductive rate by $\alpha_0 = \eta$. In k-tournament selection, and in (μ,λ)-selection, the reproductive rate is bounded by $\alpha_0 \leq k$, respectively $\alpha_0 < \lambda/\mu$. The reproductive rate in cellular EAs [1] is bounded by $\alpha_0 \leq \Delta(G)$, i.e. the degree of the neighbourhood graph G. E.g., non-elitist cEAs with $\chi = 1$ on the ring graph are ineffective.

4 Conclusion

A new drift theorem for analysis of non-elitist populations has been introduced. The conditions of the theorem decouple the effects of the selection mechanism and the variation operator — they are w.r.t. a random walk of a single individual, and not the population as a whole — thus simplifying the analysis of EAs greatly. The proof of the theorem combines results about multi-type branching processes with drift analysis. A special case of the theorem for Pseudo-boolean functions is derived, and applied on several selection mechanisms and example functions.

Acknowledgements. This work was supported by Deutsche Forschungsgemeinschaft (DFG) grant no. WI 3552/1-1, and EPSRC grant no. EP/D052785/1.

References

1. Cantú-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis 10(2), 141–171 (1998)
2. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
3. Haccou, P., Jagers, P., Vatutin, V.: Branching Processes: Variation, Growth, and Extinction of Populations. Cambridge University Press, Cambridge (2005)
4. Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. Advances in Applied Probability 14(3), 502–525 (1982)
5. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Natural Computing 3(1), 21–35 (2004)
6. Kolotilina, L.Y.: Bounds and inequalities for the Perron root of a nonnegative matrix. Journal of Mathematical Sciences 121(4), 2481–2507 (2004)
7. Lehre, P.K., Yao, X.: On the impact of the mutation-selection balance on the runtime of evolutionary algorithms. In: Proc. of FOGA 2009, pp. 47–58. ACM, New York (2009)
8. Minc, H.: On the maximal eigenvector of a positive matrix. SIAM Journal on Numerical Analysis 7(3), 424–427 (1970)
9. Neumann, F., Oliveto, P.S., Witt, C.: Theoretical analysis of fitness-proportional selection: landscapes and efficiency. In: Proc. of GECCO 2009, pp. 835–842. ACM, New York (2009)
10. Sasaki, G.H., Hajek, B.: The time complexity of maximum matching by simulated annealing. Journal of the ACM 35(2), 387–403 (1988)
11. Seneta, E.: Non-Negative Matrices. George Allen & Unwin Ltd., London (1973)

Log(λ) Modifications for Optimal Parallelism

Fabien Teytaud and Olivier Teytaud

TAO, LRI, UMR 8623(CNRS - Université Paris-Sud),
bat 490 Universite Paris-Sud 91405 Orsay Cedex France
fteytaud@lri.fr

Abstract. It is usually considered that evolutionary algorithms are highly parallel. In fact, the theoretical speed-ups for parallel optimization are far better than empirical results; this suggests that evolutionary algorithms, for large numbers of processors, are not so efficient. In this paper, we show that in many cases automatic parallelization provably provides better results than the standard parallelization consisting of simply increasing the population size λ. A corollary of these results is that logarithmic bounds on the speed-up (as a function of the number of computing units) are tight within constant factors. Importantly, we propose a simple modification, termed log(λ)-correction, which strongly improves several important algorithms when λ is large.

1 Introduction

Evolutionary algorithms (EAs) are well known robust and simple optimization algorithms. It is usually said that EAs are highly parallel, because they are population based [5]. In this paper we study the case for which we have a large number of processors, and we note that the theoretical bounds are far better than the empirical results for the current version of the algorithms in continuous domains. In Section 2 we summarize the state of the art for complexity lower bounds in EA and parallel EAs, especially for the continuous case. Section 3 shows how an optimal speed-up for parallel EAs can be reached; this is an automatic construction of a parallel algorithm with asymptotically optimal speed-up. Section 4 shows that this optimal speed-up is not reached by several well known algorithms. Section 5 shows experimentally the efficiency of parallel algorithms derived from our theoretical analysis; a similar modification, termed log(λ)-correction, is applied to several classical algorithms. Section 6 concludes. Due to length constraints, all proofs have been reported to http://www.lri.fr/~teytaud/ppsn10long.pdf

2 Complexity Bounds for Evolutionary Algorithms

We consider optimization in a domain S, subset of a normed vector space (S might be a non-empty subset of $\mathbb{R}^d$, or bistrings, the theoretical analysis below does not require anything more). For $\epsilon > 0$, we define $N(\epsilon)$ to be the maximum integer n such that there exist n distinct points $x_1, \ldots, x_n \in S$ with $\|x_i - x_j\| \geqslant 2\epsilon$

for all $i \neq j$. In particular, $N(\epsilon) = |S|$ when ϵ is small enough in the case of a finite domain S, and $\log N(\epsilon) \sim N \log(1/\epsilon)$ when $\epsilon \to 0$ if the domain $S \subset \mathbb{R}^N$ is bounded with non-empty interior. For a domain included in $\mathbb{R}^N$, we then consider the convergence ratio $CR = \frac{\log N(\epsilon)}{N n_{\epsilon,\frac{1}{2}}}$, where:

- $n_{\epsilon,\frac{1}{2}}$ is the number of evaluations necessary for ensuring, with probability at least $\frac{1}{2}$, a distance $||\hat{x} - x^*||$ at most ϵ between the approximation $\hat{x}$ of the optimum and the optimum x^*. The choice of the $\frac{1}{2}$ is arbitrary; other constants lead to similar results.
- N is the dimension of the search space.

A faster algorithm means CR larger. Convergence rate is usually defined as $\exp(-CR)$. Following [13] we prefer the convergence ratio as it is more convenient for expressing speed-ups; the speed-up between two algorithms is just the ratio between their convergence ratios, and the number of iterations for reaching a given precision is proportional to the inverse of the convergence ratio. Table 1 summarizes known bounds on the convergence ratio, and distinguishes:

- (μ, λ)-ES and $(\mu+\lambda)$-ES, respectively non-elitist and elitist Evolution Strategies; in the former case, the μ best points among λ generated points are selected, whereas in the latter case the μ best points among the union of (i) the λ generated points, and (ii) the previous population, are selected.
- full ranking (FR) evolution strategies and selection-based (SB) evolution strategies; in the former case, the optimization algorithm is informed of the complete ranking of the μ selected points, whereas in the latter case the optimization algorithm is only informed of which μ points are the best ones. For example, $(\mu/\mu, \lambda)$-ES, i.e. the new parent is simply the average of the μ best points (intermediate recombination), are selection-based, whereas weighted recombination is full ranking. For $\mu = 1$, there's no difference between FR and SB.

These concepts will be formalized below (Eq. 1-4) and we will study the optimal speed-ups, i.e. the convergence ratio as a function of λ.

3 Automatic Speculative Parallelization

A solution (in some cases) for automatic parallelization of an algorithm consists in developing the tree of possible futures, to compute separately all branches, and then to discard bad (non chosen) branches. This is a form of speculative parallelization [4]. We here show that this simple approach can be applied to EAs. We have to introduce a somehow tedious formalization; this is necessary for the mathematical formalization of our proofs. As already pointed out in [14], most EAs can be rewritten as follows:

$$(x_{n\lambda+1}^{O_1,O_2}, \ldots, x_{(n+1)\lambda}^{O_1,O_2}) = O_1(\theta, I_n) \qquad \text{(generation)} \qquad (1)$$

$$\forall i \in [\![n\lambda+1, (n+1)\lambda]\!], y_i = f(x_i^{O_1,O_2}) \qquad \text{(fitness)} \qquad (2)$$

$$g_n^{O_1,O_2} = g(y_{n\lambda+1}, \ldots, y_{(n+1)\lambda}) \qquad \text{(selection)} \qquad (3)$$

$$I_{n+1} = O_2(I_n, \theta, g_n^{O_1,O_2}), \qquad \text{(update)} \qquad (4)$$

Table 1. Upper bound on the convergence ratio; also some lower bounds on the convergence ratio for $\lambda = 2N$ for the sphere function, in the last row - these lower bounds from [13] show that a linear speed-up can be achieved w.r.t. λ constant for $\lambda = 2N$ (compare with the first row). The first row is the general case [14]; it holds in all cases, and is sometimes better than other rows (when λ is small). The second row is when the level sets of fitness functions have VC-dimension V in $\mathbb{R}^N$. The third row is just the application of the second row to the case of convex quadratic functions ($V = \Theta(N^2)$). The fourth row is the special case of the sphere function [13]. The tightness of the $\log(\lambda)$ dependency will be shown in this paper.

Framework	SB-(μ,λ)-ES	SB-$(\mu+\lambda)$-ES	FR-(μ,λ)-ES	FR-$(\mu+\lambda)$-ES
General case	$\frac{1}{N}\left(\lambda - \frac{1}{2}\log(2\pi\lambda)\right)$	$\frac{1}{N}\left(\log\binom{\lambda}{\mu}\right)$	$\frac{1}{N}\left(\lambda - \frac{1}{2}\log(2\pi\lambda)\right)$ $\times \log(\mu!)$	$\frac{1}{N}\left(\log\binom{\lambda}{\mu}\right)$ $\times \log(\mu!)$
VC-dimension V	$\frac{V}{N}\log(\lambda)$	$\frac{V}{N}\log(\lambda+\mu)$	$\frac{V}{N}(4\mu + \log(\lambda))$	$\frac{V}{N}(4\mu + \log(\lambda))$
Quadratic case	$O(N\log(\lambda))$	$O(N\log(\lambda+\mu))$	$O(N(\mu+\log(\lambda)))$	$O(N(\mu+\log(\lambda)))$
Sphere function	$(1+\frac{1}{N})\log(\lambda)$	$(1+\frac{1}{N})\log(\mu+\lambda)$	$2\log(\lambda)$	$O(\mu+\log(\lambda))$
Sphere function with $\lambda = 2N$			$\Omega(1)$	$\Omega(1)$

for some fixed O_1, O_2, I_0, some random variable θ, and g with values in a set of cardinality K, where:

- I_0 is the initial state and I_n is the internal state at iteration n;
- θ is the random seed;
- g^{O_1,O_2} is the information used by the algorithm, typically in our case the indices of the selected points (and possibly their ranking in the FR case);
- $x_k^{O_1,O_2}$ is the k^{th} visited point and y_k is its fitness value (y_k should, theoretically, be indexed with O_1, O_2 as well);
- (O_1, O_2) is the optimization algorithm, with:
  - O_1 is the function generating the new population (as a function of the random seed and of the internal state);
  - O_2 is the function updating the internal state as a function of the random seed and of the extracted information g.

(note that $g_n^{O_1,O_2}$ and $x_n^{O_1,O_2}$ both depend on θ and f; we drop the indices for the sake of clarity.) We will term such an optimization algorithm a λ-optimization algorithm; this means that λ fitness values are computed at each iteration. The optimization algorithm is defined by O_1, O_2, I_0, θ; in cases of interest (below) we will use the same θ and the same I_0 for all algorithms and therefore only keep the dependency in O_1 and O_2 in notations. In EAs, g_n has values in a discrete domain; typically, either g_n has values in the set of the finitely many possible ranking of the individuals; or g_n has values in the finite set of possible vectors of ranked indices of selected individuals. g_n is in both cases the only information that the algorithm extracts from the fitness function. In the FR case and $\mu = \lambda$, for example g_n is

$(sign(y_{n\lambda+i} - y_{n\lambda+j})_{(i,j)\in[\![1,\lambda]\!]^2})$ where $sign(t) = 1$ for $t \geq 0$ and $sign(t) = -1$ otherwise. In the SB case for (μ, λ)-ES, the formulation is a bit more tedious:

$$g_n = \{I = \{i_1, \ldots, i_\mu\} \subset [\![1, \lambda]\!]^\mu; Card\ I = \mu \text{ and}$$
$$k \in I \wedge k' \in [\![1, \lambda]\!] \setminus I \Rightarrow y_{n\lambda+k} \leq y_{n\lambda+k'}\}.$$

An important property is that the set of possible values for g_n has cardinality $K < \infty$; K can be bounded as follows:

- (μ, λ)-ES (evolution strategies) with equal weights; then $K \leq \lambda!/(\mu!(\lambda-\mu)!)$;
- (μ, λ)-ES with weights depending on the rank; then $K \leq \lambda!/(\lambda - \mu)!$;
- $(1 + \lambda)$-ES; then $K \leq \lambda + 1$;
- $(1, \lambda)$-ES; then $K \leq \lambda$.

K will be termed the *branching factor* of the algorithm. The branching factor, and bounds in Table 1 on the branching factor, have been used in [13] for proving results shown in Table 1; we will use it here for proving lower bounds on the parallelization of EAs; the lower the branching factor, the better the speed-up. We will say that a λ'-optimization algorithm O'_1, O'_2 simulates a λ-optimization algorithm O_1, O_2 with speed-up D if and only if

$$\forall\theta, \forall n \geq 0, \forall i \in [\![1, \lambda]\!], x^{O'_1,O'_2}_{n\lambda'+i} = x^{O_1,O_2}{}_{nD\lambda+i}. \tag{5}$$

θ is the random seed; it is removed of indices for short as discussed above, rigorously all the x's depend on it. We now show how we can automatically build O', which is equivalent to O, but with $\lambda' > \lambda$ evaluations at the same time and a known speed-up.

Theorem 1. (Automatic parallelization of EAs and tightness of the log(λ) speed-up.) *Consider a λ-optimization algorithm (O_1, O_2) as in Eqs 1-4 with branching factor K, and consider λ' such that for some $D \geq 1$:*

$$\lambda\frac{K^D - 1}{K - 1} = \lambda'. \tag{6}$$

Then, there is a λ'-optimization algorithm which simulates (O_1, O_2) with speed-up D.

Remark. The speed-up is therefore $D = \frac{\log(1+\frac{\lambda'}{\lambda}(K-1))}{\log(K)}$.

4 Real World Algorithms Don't All Reach the Optimal Speed-Up

In this section we show that the one-fifth rule, the self-adaptation and the cumulative step-size adaptation all do not reach the optimal speed-up (the optimal speed-up is log(λ) for $\lambda \to \infty$, see Table 1 and [13]) when using the natural parallelization consisting in increasing λ to the number of processors and evaluating one individual per core. More precisely, these classical algorithms have bounded speed-up as a function of λ (i.e. speed-up $O(1)$ as $\lambda \to \infty$). In all sections below, we consider optimization in the continuous domain, with Gaussian mutations and define $\eta^* = \sigma_{n+1}/\sigma_n$ (η^* depends on n, but we will consider a fixed value of n here and therefore we will drop this dependency in the notation η^*).

The main important point is that the convergence rate is lower bounded by η^*; formally, $CR \leq \mathbb{E} - \log(\eta^*)$. Roughly speaking, it is not possible to decrease the distance to the optimum by z at each iteration (on average, logarithmically), if you don't divide the step-size by z, on average. Therefore, it will be sufficient, in the sequel, to lower-bound η^* for various classical step-size adaptation rules, independently of λ, in order to show that the step-wise adaptation does not provide an optimal convergence rate as $\lambda \to \infty$ (an optimal convergence rate should be $\eta^* = O(1/\log(\lambda)))$. More precisely, as η^* is a random variable, we have to show that the expected logarithm of η^*, i.e. $\mathbb{E}\log\eta^* = \mathbb{E}\log(\sigma_{n+1}/\sigma_n)$ is lower bounded by a constant $> -\infty$. The following sections (4.1, 4.2, 4.3) use this fact for showing the poor efficiency of the usual algorithm for $\lambda \to \infty$.

4.1 One-Fifth Rule

The one-fifth rule [8] is the oldest and most well known algorithm for adapting the step-size. The one-fifth rule can be applied in different manners to $(\mu/\mu, \lambda)$ algorithms. Consider $\hat{p}$ equal to the ratio between (i) the number of generated individuals with fitness better than the center of the Gaussian generating the offspring (ii) the number of generated individuals; $0 \leq \hat{p} \leq 1$. A first possible implementation of the one-fifth rule is

$$\hat{p} \leq 1/5 \Rightarrow \eta^* = K_1 \in]0,1[\text{ and } \hat{p} > 1/5 \Rightarrow \eta^* = K_2 > 1 \tag{7}$$

$$\text{and a second version is } \eta^* = K_3^{(\hat{p}-1/5)} \text{ for some } K_3 > 1. \tag{8}$$

Proposition 1. *The one-fifth rule, implemented as in Eq. 7 or in Eq. 8, has the property that for each iteration n, there is $C > -\infty$ such that $\mathbb{E}\log(\frac{\sigma_{n+1}}{\sigma_n}) > C$.*
Therefore, we have shown that with the one-fifth rule, the convergence ratio (and therefore the convergence rate) is $O(1)$ (as $\lambda \to \infty$; convergence rates and ratios are defined in Section 2).

4.2 Self-adaptation (SA)

The proof of the limited speed-up for SA requires the following lemma.

Lemma. *The expected logarithm of the average (arithmetic or geometric average) of the μ smallest of λ independent standard log-normal random variables, with $\mu/\lambda \to k > 0$ and $\mu > 0$, is lower bounded by some constant $> -\infty$. More formally, if $N_{(1)}, \ldots, N_{(\lambda)}$ are sorted standard independent Gaussian variables, and $L_{(i)}$ is $\exp(N_{(i)})$, then*

$$\inf_{\lambda>0} \mathbb{E}\log\frac{1}{\mu}\sum_{i=1}^{\mu}\exp(N_{(i)}) > -\infty \text{ and } \inf_{\lambda>0} \mathbb{E}\frac{1}{\mu}\sum_{i=1}^{\mu}N_{(i)} > -\infty.$$

Proposition 2. *Consider a SA algorithm in which σ_{n+1} is the average (geometric or arithmetic average) of $\sigma_n \times L_1, \sigma_n \times L_2, \ldots, \sigma_n \times L_\lambda$, for $L_1, \ldots, L_\lambda$ as in the lemma above. Then, there exists some $C > -\infty$ such that $\mathbb{E} \log(\frac{\sigma_{n+1}}{\sigma_n}) > C$.*

Remark. Rescaling the N_i by any constant (equivalently, $L_i = \exp(kN_i)$ for some $k > 0$) does not change the result.

4.3 Cumulative Step-Size Adaptation

It has been experimentally shown in [3] that CMA has a poor speed-up as a function of λ. Empirically, the Estimate of Multivariate Normal Algorithm (EMNA) [7] has a much better behavior, but the speed-up curve becomes constant as a function of λ, instead of logarithmic, for λ large [10]. We here show formally that Cumulative Step-size Adaptation (CSA) does not reach optimal speed-up $\log(\lambda)$. We (classically) formalize an iteration of CSA in dimension N as follows:

$$w_i \geq 0, \sum_{i=1}^{\mu} w_i = 1 \quad (9)$$

$$\mu_{eff} = \frac{1}{\sum_{i=1}^{\mu}(w_i^2)} \quad (10)$$

$$\chi_N > 0 \ , \ ||p_c|| \geq 0 \quad (11)$$

$$d_\sigma = 1 + 2\max(0, \sqrt{\frac{\mu_{eff}-1}{N+1}} - 1) \quad (12)$$

$$c_\sigma = \frac{\mu_{eff}+2}{N+\mu_{eff}+3} \quad (13)$$

$$\sigma_{n+1} = \sigma_n \exp\left(\left(\frac{||p_c||}{\chi_N} - 1\right) \cdot \frac{c_\sigma}{d_\sigma}\right).$$

($||.||$ does not have to be a norm, we just need Eq. 11). These assumptions, to the best of our knowledge, hold in all current implementations of CSA. We then show the following

Proposition 3. *For any dimension N, there exists $C > 0$ such that, for any λ, $\eta_n^* = \frac{\sigma_{n+1}}{\sigma_n} \geq C$.*

This proposition shows that $\eta^* \geq \exp(-1)$; this implies that $CR \leq 1$, *i.e.* for cumulative step-size adaptation the speed-up is $O(1)$ for $\lambda \to \infty$.

5 Experimental Speed-Up

Theorem 1 proves that this automatic parallelization reaches $\log(\lambda)$, which is asymptotically optimal within a constant factor, but there are algorithms for which automatic parallelization works only for λ very large, in particular when the full ranking of selected individuals is used (because in this case the branching factor K is much bigger). Therefore, in this section, we will provide other tricks than the automatic parallelization for ensuring the suitable $\log(\lambda)$ property. In all cases below, we keep a parallelization based on the simple principle of one individual per processor, but we modify either the selection ratio or the step-size adaptation rule, so that this principle leads to much better speed-ups. If the speed-up is bounded, then σ is divided by, at most, a fixed constant, independently of λ. If we want to reach the "$\log(\lambda)$" speed-up, then we must decrease $\log(\lambda)$ by $\Theta(\log(\lambda))$; *i.e.* divide σ by an exponent of λ. We will here apply $\sigma \leftarrow \sigma / \max(1, (\zeta\lambda)^{1/N})$ for some value of ζ. We consider, CMSA, EMNA and CMA-ES. CMA-ES is interesting; as it is a FR-(μ, λ)-ES, and therefore has a big branching factor $K = \lambda!/(\lambda - \mu)!$, and therefore the automatic parallelization becomes efficient only for huge numbers of processors - we will show below simple tricks empirically solving this trouble.

5.1 The log(λ) Correction for CMSA

CMSA is the algorithm for which implementing the $\log(\lambda)$ correction is the easiest: we just have to modify the selection ratio μ/λ. We give experimental results in Fig. 1, and a more detailed presentation and analysis of this correction can be found in [9].

5.2 The log(λ) Correction for EMNA

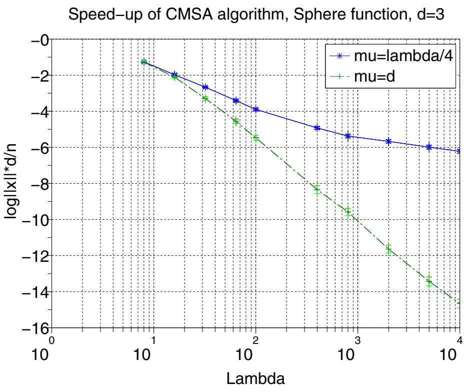
We present results of the isotropic EMNA (the step-size is the same in all directions), on the sphere function. The presented numbers are the mean progress of the log of the distance to the optimum, multiplied by the dimension[1], estimated with the following experimental conditions:

(1) Column "baseline": the standard EMNA algorithm from [7], with $\mu = \lambda/4$;
(2) Column "+QR": EMNA, plus the quasi-random mutations as defined in [12];
(3) Column "+log(λ)": the same as "+QR", except that we add the $\log(\lambda)$ correction, i.e. we modify σ according to formula $\sigma \leftarrow \sigma/\max(1, (0.15\lambda)^{1/N})$ (which ensures that $\log(\sigma)$ decreases by $\tilde{}\log(\lambda)$ as requested above);
(4) Column "+weighting": the same as "+log(λ)", except that we apply the reweighting as in [11] (this reweighting is based on the density of the Gaussian used for the offspring; variants of reweighting based on the ranks can be found in [1,2]).

In all cases the initial step-size is $\sigma = 1$ and the initial point is randomly drawn on the unit sphere with radius $\sqrt{N}$ with N the dimension. The 3 following columns provide the p-value of the comparison between a column and the previous column; the significance is very high. Then, the last column presents the normalized convergence rate of the algorithm with QR and reweighting, but without the $\log(\lambda)$-correction; with this column, we can check that the improvement is due to the $\log(\lambda)$ correction and not to the combination QR+reweighting. This is detailed in Figure 1 (left), with result in Table 2. Interestingly, the $\log(\lambda)$ correction is not efficient if we do not apply the reweighting trick from [11]. This is somewhat natural, as the $\log(\lambda)$ correction strongly increases the risk of premature convergence, which is reduced by the reweighting.

[1] It is known that the log-distance to the optimum decreases linearly with the dimension; therefore we multiply the results by the dimension in order to have homogeneous results for various dimensions. Following the theoretical analysis in [13], we expect an improvement as the dimension increases, which is confirmed experimentally here.

```
Initialize $\sigma \in \mathbb{R}$, $y \in \mathbb{R}^N$.
while Halting criterion not fulfilled do
  for $l = 1..\lambda$ do
    $z_l = \sigma N_l(0, Id)$
    $y_l = y + z_l$
    $f_l = f(y_l)$
  end for
  if "Reweighting" version then
    Let $w(i) = 1/density(x_i)$
    // with $density$ the proba. density
    // used for generating the offspring.
  else
    Let $w(i) = 1$
  end if
  Sort the indices by increasing fitness:
   $f_{(1)} < f_{(2)} < \cdots < f_{(\lambda)}$.
  $z^{avg} = \frac{1}{\sum_i^\mu w_{(i)}} \sum_{i=1}^{\mu} w_{(i)} z_{(i)}$
  $\sigma = \sqrt{\frac{\sum_{i=1}^{\mu} w(i)||z_{(i)} - z^{avg}||^2}{\sum_{i=1}^{\mu} w(i) \times N}}$
  if $\log(\lambda)$ version then
    $\sigma = \sigma / \max(1, (0.15\lambda)^{1/N})$.
  end if
  $y = y + z^{avg}$
end while
```



λ	CMA	CMA with log(λ)-correction
Dimension 2		
$8 \times N$	-0.100±0.001	**-0.177±0.001**
$8 \times N^2$	-0.0741±0.0009	**-0.134±0.001**
Dimension 10		
$8 \times N$	-0.0338±6e-05	**-0.0389±0.0001**
$8 \times N^2$	-0.00971±6e-05	**-0.0174±0.0001**
Dimension 30		
$8 \times N$	-0.0107±1e-05	**-0.0118±2e-05**
$8 \times N^2$	-0.00188±1e-05	**-0.00370±1.e-05**

Fig. 1. Left: The EMNA algorithm with weighted averages. N_l is a Gaussian random variable, or a Gaussian quasi-random variable for "QR" versions (see text). **Right, top:** Example of the limited speed-up of real-world algorithms, and the strong improvement provided by a simple correction. n is the number of iterations; the algorithms run until fitness value 10^{-10} is reached, x is the best point so far. This experiment is done in dimension 3, and we plot the log-distance to the optimum normalized by the dimension and the number of generations of the algorithm (the lower the result, the better). The usual initialization $\frac{\mu}{\lambda} = \frac{1}{4}$ is outperformed, by far, by $\min(d, \lfloor\lambda/4\rfloor)/\lambda$. **Right, bottom:** Comparison between CMA and CMA with log(λ)-correction in various dimensions. The maximum number of function evaluations is 400 (in dimension 2), 10 000 (in dimension 10) and 90 000 (in dimension 30), and the constant ζ involved in the λ correction (Eq. 14) $0.4^{1/2}$ in dimension 2, 1 in dimension 10, $1.3^{1/30}$ in dimension 30. In all cases the λ-correction provides an improvement. Whereas in the case of EMNA we could use the same constant in all cases and the results were very stable as a function of the constant, with CMA we had to modify the constant ζ as a function of the dimension in order to get good results.

5.3 The log(λ) Correction for CMA-ES

We propose to add the following line in CMA, after the computation of σ:

$$\sigma = \sigma / \max(1, (\zeta\lambda)^{1/N}). \tag{14}$$

This formula avoids the bad behavior pointed out in Proposition 3 and experimentally strongly improves the results. We consider f as the best fitness found by the algorithm after a fixed number of evaluations. We report the mean of $\frac{N \cdot \log(f)}{\#evaluations}$

Table 2. Convergence rates of EMNA. We see that (i) QR works very well (ii) reweighting does not always improve the results (it has been published as a tool against premature convergence and not as a tool for fastening EMNA) (iii) the $\log(\lambda)$ correction greatly improves the results, but only if reweighting is applied; this is somewhat natural, as, without reweighting, the $\log(\lambda)$ correction increases the risk of premature convergence.

Dimension, lambda	Baseline	+QR	+$\log(\lambda)$	+weight	P-value for			QR+weight
					+QR	+ $\log(\lambda)$	+weight	but no $\log(\lambda)$
2,20	-1.61	-1.91	-0.66	-2.43	0.00	1	0	-2.02
2,60	-2.04	-2.13	-0.27	-3.95	0.00	1	0	-2.17
2,200	-2.17	-2.27	-0.17	-5.31	6e-16	1	0	-2.16
2,600	-2.22	-2.27	-0.14	-6.44	4e-15	1	0	-2.27
2,2000	-2.22	-2.38	-0.13	-7.68	0	1	0	-2.32
2,6000	-2.33	-2.51	-0.13	-8.85	0	1	0	-2.38
3,30	-2.09	-2.49	-0.69	-1.67	0.00	1	0	
3,300	-2.53	-2.59	-0.21	-6.02	0.00	1	0	
3,3000	-2.65	-2.87	-0.16	-8.52	0	1	0	
3,9000	-2.77	-2.94	-0.15	-9.63	3e-16	1	0	
5,50	-2.72	-2.96	-0.54	-3.28	1e-12	1	0	-2.72
5,500	-3.08	-3.26	-0.31	-6.97	2e-14	1	0	-3.00
5,5000	-3.35	-3.63	-0.22	-9.56	0	1	0	-3.32
5,15000	-3.53	-3.74	-0.20	-10.84	1e-15	1	0	-3.53
20,200	-5.56	-5.89	-2.52	-2.24	1e-09	1	0.74	-3.30
20,2000	-6.81	-7.17	-1.44	-11.27	1e-13	1	0	-6.29
20,60000	-7.93	-8.09	-0.87	-16.17	1e-08	1	0	-7.96
40,400	-8.36	-8.83	-5.35	-1.31	3e-09	1	1	
40,1200	-9.27	-9.54	-4.33	-2.94	8e-05	1	0.97	
40,4000	-10.00	-10.15	-3.47	-8.25	3e-05	1	0	
40,12000	-10.38	-10.48	-2.88	-16.30	0.01	1	0	

and the mean of $\log(f)$ in Fig. 1. The number of function evaluations is $100N^2$. Following [3], we experiment two size of population, $\lambda = 8N$ and $\lambda = 8N^2$. If the dimension is small (2) we almost have a speed-up of 2 independently of the size of the population. However, if the dimension becomes larger (10 or 30) we have a good speed-up only if the size of the population is large ($\lambda = 8N^2$). The results are good, but not very good, and CMA with this correction is still far from the efficiency of CMSA or EMNA for large population size; we guess however that improvements of our formula above are possible, and also we guess that modifying the rule for computing the new parent should be adapted for λ large.

6 Conclusion

The new results in this paper are as follows. First, we have shown in Section 3 that theoretical bounds in [13] are tight for their dependencies in λ. In particular, well parameterized algorithms should have a speed-up $\Theta(\log(\lambda))$. Second, we have shown in Section 4 that many current algorithms do not match this tight dependency. Propositions 1, 2 and 3 show that the speed-up is $O(1)$ for the one-fifth rule, the self-adaptation, and cumulative self-adaptation respectively. The tightness is shown by an explicit construction of a parallel version of EA, which can readily be applied also for direct search methods [6] as well; thanks to this explicit construction, we provide an automatic parallelization with, provably, asymptotically better results. Related experimental results are shown in Section 5. They show that parallel algorithms derived from our analysis are faster and in some cases by far than algorithms based on simply increasing λ; moreover the new version is not more difficult to implement.

References

1. Arnold, D.V.: Weighted multirecombination evolution strategies. Theor. Comput. Sci. 361(1), 18–37 (2006)
2. Arnold, D.V., Scott Van Wart, D.C.: Cumulative step length adaptation for evolution strategies using negative recombination weights. 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. 545–554. Springer, Heidelberg (2008)
3. 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)
4. Calder, B., Reinman, G.: A comparative survey of load speculation architectures. J. Instruction-Level Parallelism 2 (2000)
5. Cantú-Paz, E.: Efficient and accurate parallel genetic algorithms. Kluwer Academic Publishers, Boston (2000)
6. Conn, A., Scheinberg, K., Toint, L.: Recent progress in unconstrained nonlinear optimization without derivatives. Mathematical Programming 79, 397–414 (1997)
7. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)
8. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
9. Teytaud, F.: How we can derive parameters of ES for parallel optimization. In: EvoStar 2010 (accepted, 2010)
10. Teytaud, F., Teytaud, O.: On the parallel speed-up of Estimation of Multivariate Normal Algorithm and Evolution Strategies. In: Proceedings of EvoStar 2009, pp. 655–664 (2009)
11. Teytaud, F., Teytaud, O.: Why one must use reweighting in estimation of distribution algorithms. In: Proceedings of Gecco, pp. 453–460 (2009)
12. Teytaud, O.: When does quasi-random work? In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 325–336. Springer, Heidelberg (2008)
13. 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)
14. Teytaud, O., Gelly, S.: General lower bounds for evolutionary computation. In: Proceedings of PPSN, pp. 21–31 (2006)

The Linkage Tree Genetic Algorithm

Dirk Thierens

Institute of Information and Computing Sciences
Universiteit Utrecht, The Netherlands
`dirk.thierens@cs.uu.nl`

Abstract. We introduce the *Linkage Tree Genetic Algorithm (LTGA)*, a competent genetic algorithm that learns the linkage between the problem variables. The LTGA builds each generation a *linkage tree* using a hierarchical clustering algorithm. To generate new offspring solutions, the LTGA selects two parent solutions and traverses the linkage tree starting from the root. At each branching point, the parent pair is recombined using a crossover mask defined by the clustering at that particular tree node. The parent pair competes with the offspring pair, and the LTGA continues traversing the linkage tree with the pair that has the most fit solution. Once the entire tree is traversed, the best solution of the current pair is copied to the next generation. In this paper we use the normalized variation of information metric as distance measure for the clustering process. Experimental results for fully deceptive functions and nearest neighbor NK-landscape problems with tunable overlap show that the LTGA can solve these hard functions efficiently without knowing the actual position of the linked variables on the problem representation.

1 Introduction

In general, the search bias of recombination operators can be beneficial to the search efficiency in the following two cases:

1. Different partial structures of two good solutions can be juxtaposed by crossover to form a new good solution.
2. Common partial structures shared by two good solutions are shielded from crossover disruption, and the new solution inherits the common partial structures, while it randomly samples the subspace where the parents disagree.

In this paper we focus on the first case. In a standard genetic algorithm this case can only work if the partial structures are not disrupted too often by recombination. This can be achieved by designing the solution representation and/or the crossover operator in an appropriate way. However, if there is not enough domain knowledge to design a suitable representation and/or crossover operator, we have to induce this knowledge from a population of solutions. Learning what variables form important partial solutions - and therefore should be protected from disruption by crossover - is called *linkage learning*. During the past decade a number of linkage learning evolutionary algorithms have been proposed [1][4][9].

In this paper we introduce an alternative linkage learning GA called the *Linkage Tree Genetic Algorithm (LTGA)*. The LTGA builds each generation a linkage tree using a hierarchical clustering algorithm. To generate new offspring it traverses the linkage tree and uses the clustering specified at each tree node as a crossover mask. The next section explains how the LTGA works. Section 3 discusses the distance measure we use to learn the linkage between variables and groups of variables. We also compare the LTGA to related linkage learning GAs. Section 4 shows experimental results of the LTGA on deceptive trap functions and nearest neighbor NK-landscape problems with tunable overlap. Finally, Section 5 concludes the paper.

2 Linkage Tree Genetic Algorithm

2.1 Linkage Tree

Linkage learning evolutionary algorithms aim to identify which variables should be treated as a dependent set of variables during the exploration phase. In this paper we learn linkage between variables - and groups of variables - by building a hierarchical cluster using a proximity distance that measures how correlated the variables or groups of variables are in the current population.

Definition 1. *The* Linkage Tree *of a population of solutions is the hierarchical cluster tree of the problem variables using an agglomerative hierarchical clustering algorithm with a distance measure D. The distance measure $D(X_1, X_2)$ measures the degree of dependency between two sets of variables X_1 and X_2.*

Algorithm. HIERARCHICAL CLUSTERING

```
1. Compute the proximity matrix using metric $D$.
2. Assign each variable to a single cluster.
3. Repeat until one cluster left:
4.  Join two nearest clusters $c_i$ and $c_j$ into $c_{ij}$.
5.  Remove $c_i$ and $c_j$ from the proximity matrix.
6.  Compute distance between $c_{ij}$ and all clusters.
7.  Add cluster $c_{ij}$ to the proximity matrix.
```

An agglomerative hierarchical clustering algorithm proceeds bottom-up. First, each problem variable is assigned to a single cluster. Then, the clustering algorithm recursively joins the closest clusters until only one cluster is left. For a problem of length ℓ the linkage tree has ℓ leaf nodes (the clusters having a single problem variable) and $\ell - 1$ internal nodes. Each (internal or leaf) node of the linkage tree divides the set of problem variables into two mutually exclusive subsets. One subset is the cluster of variables at that node, while the other subset is the complementary set of problem variables. The LTGA uses this division of the problem variables as a crossover mask. The variables specified in the cluster of a specific node are swapped between two parent solutions to generate an offspring pair. For instance, assume LTGA crosses the parent pair $(00001111, 00110011)$ at the internal node of the linkage tree with cluster (x_0, x_1, x_4, x_5). The values at position 0, 1, 4 and 5 are swapped which results into the offspring pair $(00000011, 00111111)$. The LTGA evaluates the fitness of the offspring and holds a competition between the parent pair and the offspring pair. If one of the children is better than both parents the offspring pair replaces the parent pair, and LTGA continues to traverse the linkage tree with the new pair. If none of the two children is better than both parents, LTGA continues its tree traversal with the parent pair. When the tree is completely traversed, the best solution of the current pair is copied to the next generation. To increase the efficiency, LTGA always checks whether the offspring solutions are actually different from their parents, if not no call to the fitness function is done, and the algorithm simply proceeds with its tree traversal.

The order in which the tree is traversed, is the opposite order of the merging of the clusters by the hierarchical clustering algorithm. Therefore, LTGA first crosses the clusters which are the least dependent on each other - this is, they are least linked. The clustering algorithm initially pushes all single variable clusters on a stack. Then, it iteratively joins the two closest clusters and pushes the joint cluster on the stack. When all clusters are merged the stack will consist of $2\ell - 1$ clusters, ℓ of them being single variable clusters. During tree traversal, LTGA simply pops the clusters of the stack and uses them as crossover mask. Note that half of the crossovers are actually single bit flips, so LTGA is also performing a bitwise search. Of course this is redundant in applications where the LTGA is combined with another local search algorithm, in this case the single variable clusters are not pushed on the stack, effectively reducing the number of crossover masks by half.

Algorithm. LINKAGE TREE GENETIC ALGORITHM (LTGA)

```
 1. Create initial population of size $N$.
 2. Repeat until stop criteria met:
 3.  Build the linkage tree.
 4.  Do for $N$ solution pairs:
 5.   Traverse one step in the linkage tree.
 6.   Set crossover mask to the current clustering.
 7.   Cross solution pair using the crossover mask.
 8.   When one offspring is better than both parents:
 9.    Replace parent pair with the offspring pair.
10.   If tree fully traversed:
11.    Copy best solution to next population.
12.    Else, go to step 5.
```

3 Hierarchical Clustering Using Mutual Information

The hierarchical clustering algorithm requires a distance measure that measures the amount of linkage between two clusters of variables. In [7] a hierarchical clustering algorithm is outlined that uses a mutual information based distance measure. In this paper we will use the same distance measure to build the linkage tree of a population of solutions. Assume X_k is a discrete, random variable with probability mass function $p(X_k)$. The entropy H is then defined as $H(X_k) = -\sum_i p_i(X_k) \log p_i(X_k)$. The mutual information I between a set of random variables is defined as $I(X_1, \ldots, X_\ell) = \sum_{k=1}^{\ell} H(X_k) - H(X_1, \ldots, X_\ell)$. Mutual information is particularly interesting for use in a hierarchical clustering algorithm due to its *grouping property.* The grouping property states that the mutual information between three clusters of random variables C_1, C_2 and C_3 is equal to the sum of the mutual information between two clusters C_1 and C_2, plus the mutual information between the union of the two clusters $C_1 \cup C_2$ and $C3$: $I(C_1, C_2, C_3) = I(C_1, C_2) + I((C_1 \cup C_2), C_3)$. It is important to realize that the mutual-information I is a similarity measure between objects but is not a distance measure. Hierarchical clustering requires a distance measure between clusters to build the cluster decomposition. A distance measure based on mutual information is the *variation of information* $d(X_1, X_2)$ which is the difference between the joint entropy $H(X_1, X_2)$ and the mutual information $I(X_1, X_2)$:

$$d(X_1, X_2) = H(X_1, X_2) - I(X_1, X_2) = H(X_1|X_2) + H(X_1|X_2).$$

In hierarchical clustering we are comparing clusters of different sizes so it is preferable to normalize the distance measure $d(X_1, X_2)$ by dividing it by the total information as represented by the entropy H:

$$D(X_1, X_2) = \frac{d(X_1, X_2)}{H(X_1, X_2)} = 2 - \frac{H(X_1) + H(X_2)}{H(X_1, X_2)}.$$

Interestingly, D is a metric with $0 \leq D(X_1, X_2) \leq 1$. In the experiments in Section 4 we will use this metric as distance measure for the hierarchical clustering.

3.1 Related Work

The LTGA is related to the ClusterMI [3] and to the Dependency Structure Matrix Genetic Algorithm (DSMGA) [13]. The DSMGA uses a (non-hierarchical) clustering algorithm to learn the linkage between the variables. DSMGA applies a bitwise hillclimbing search algorithm and a Minimum Description Length (MDL) measure to find a clustering that accurately models the building-block structure of the problem. This model is used to perform building-block wise crossover. The ClusterMI computes a hierarchical clustering using the mutual information between all the problem variables. To compute the distance between 2 clusters of variables ClusterMI computes the mean of the mutual information between the variables of each cluster. However, as mentioned above, mutual information is a similarity measure, not a distance measure, and it is not clear what it actually means to take the average mutual information as the distance measure for the hierarchical clustering algorithm. Obviously, taking the average mutual information between variables as a measure between clusters is much more efficient than computing an actual distance between clusters. Perhaps it might be a good compromise to use the average linkage distance D between variables as an approximation of the distance measure between clusters.

The main difference between ClusterMI and DSMGA on the one hand, and the LTGA on the other, is that the former two are Estimation of Distribution Algorithms (EDAs) whose aim is to learn a probabilistic model of the current population. Therefore, they try to learn the exact partitioning model that directly represents the building-block structure of the fitness function. Both the ClusterMI and the DSMGA use a MDL measure as originally proposed in the Extended Compact Genetic Algorithm (ECGA) [5] to induce this particular clustering. Learning a single partitioning however makes these algorithms vulnerable to modeling errors. This vulnerability expresses itself by the rather large minimal population sizes required for entropy-based model building in discrete estimation of distribution algorithms [14]. The LTGA is more robust in that sense. It builds a complete linkage tree and recombines parent solutions at different levels of dependencies - or linkage - between variable clusters. LTGA does not generate new solutions by sampling from a probability distribution. Instead, it repeatedly recombines a parent pair in search of a better offspring. This extensive exploration of the parent pair is guided by the structure of the fitness landscape, as expressed by the induced linkage tree.

The LTGA can actually be looked upon as a hybrid between standard genetic algorithms and estimation of distribution algorithms. By building a probabilistic model EDAs capture the global structure of the fitness landscape. Single good solutions however might contain certain detailed information that is not (well) captured in the global probability model. A GA using crossover or mutation might be better suited to preserve the intricate details of a good solution. Through the extensive recombination and mutation - combined with the family elitism - the LTGA inherits the exploration and exploitation capabilities of GAs, while the guidance by the linkage tree makes it respect the global structure of the fitness landscape as done by EDAs.

Finally, recent work on network crossover appears to share some interesting similarities with the linkage tree crossover [6] [11]. In future work we plan to compare these methods.

4 Experimental Results

4.1 Deceptive Trap functions

We have tested the LTGA on the deceptive mk-trap function DTF [2]. DTF is a binary, additively decomposable function composed of m trap functions DT_i, each defined on a separate group of k bits (the total problem length is $\ell = mk$): $DTF(x_1 \ldots x_\ell) = \sum_{i=0}^{m-1} DT_i(x_{ik}...x_{ik+k-1})$ with $x_i \in \{0, 1\}$. Call u the number of bits in such a group that are equal to 1:

$$DT_i(x_{ik}...x_{ik+k-1}) = \begin{cases} k, & if\ u = k \\ k - 1 - u, & otherwise. \end{cases}$$

Clearly, the global optimal solution is the string of all 1-bits. The number of local optima is $2^m - 1$, and a hillclimbing algorithm quickly becomes trapped in one of these local optima. Furthermore, all schemata of order less than k are deceiving. This means that the schema fitness of the schemata containing the local optima - consisting of bits 0 - are better than the competing schemata that contains the optimal bits 1. Any standard GA that uses a disruptive crossover operator - like uniform crossover - in combination with a moderate selection pressure, will quickly converge to the deceptive local optima. When the individual trap functions are tightly linked - this is, they are represented in the string as consecutive blocks - a GA using a position biased crossover like 1- or 2-point crossover will quickly find the global optimum. However, when the trap functions are randomly scattered over the string, no position biased crossover can mix the different optimal substrings to form the global optimum. In order to find the global optimum, a standard GA will have to apply a crossover operator that is not position biased, like uniform crossover, however, to balance the high disruption rate of uniform crossover the GA will have to increase the selection pressure dramatically. Unfortunately, with such a high selection pressure the population size also needs to increase dramatically in order to prevent premature convergence. As a result, the number of function evaluations needed to find the optimal solution by the standard GA scales exponentially in the problem length - this is, the number of trap functions DT_i for any fixed length k [4][12].

In the **first experiment**, we test the LTGA on deceptive functions with deception length $k = 5$. The number of mk-trap functions varies from 5 to 20 with increments of 5, the problem length thus varies from 25 to 100 with increments of 25. We fix the population size at $N = 128$ which is enough to find the optimal solution in at least 24 out of 25 independent runs. In this experiment the initial population is a population of random local optimal strings, obtained by running a bitwise hillclimbing algorithm. During the LTGA search no local search is applied anymore. Table 1 shows the first hitting time, this is the number of function evaluations needed to find the global optimum for the first time. We also show the growth ratio of the median number of function evaluations, and the growth ratio of the CPU runtime. The ratio are relative to the value of the smallest problem length ($\ell = 25$). A least squares fit indicates that the growth ratio of the number of functions evaluations is of order $\Theta(\ell \log \ell)$, while the growth ratio of the CPU runtime is of order $\Theta(\ell^{2.9} \log \ell)$.

Table 1. The 1^{st}, 2^{nd}, and 3^{rd} quartile of the number of functions evaluations needed to hit the global optimum for the first time. The population size $P = 128$. The last two columns show the growth ratio of the number of evaluations and the CPU runtime.

Length	Blocks	Evals Q1	Evals Q2	Evals Q3	Evals Ratio	Runtime Ratio
25	5	13095	15069	16253	1	1
50	10	38073	43933	45074	2.9	4
75	15	65889	70397	73615	4.7	10
100	20	96367	97818	102146	6.5	25

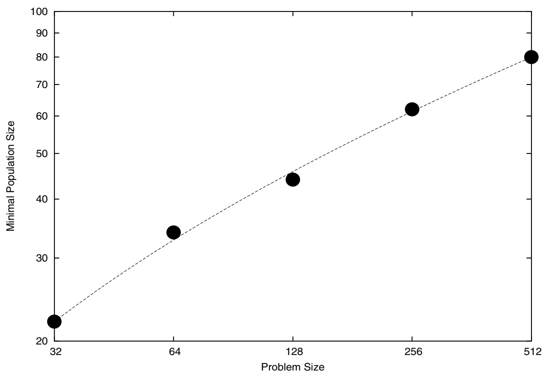


Fig. 1. The minimal population size needed for $k = 4$ problems with increasing problem length. The fitted curve is of order $\Theta(m^{0.14} \ln m)$.

In the **second experiment**, we investigate how the minimal population size scales with the problem length. The deception length is fixed at $k = 4$, the problem length varies from $\ell = 32$ to $\ell = 512$ each time doubling the string length, which corresponds to the number of blocks $m = 8, 16, 32, 64, 128$. In order to be able to compare the LTGA's scaling behavior with that of the ECGA and the ClusterMI algorithms in Figure 1 of [3], we count a run successful when at least 29 out of 30 runs have at least $m - 1$ trap functions correct. There is also no local search involved, so the LTGA is run on a random initial population. Figure 1 shows the results and a least-squares fit of $P = 8\ m^{0.14} \ln m$. LTGA's scaling behavior is thus $\Theta(m^{0.14} \ln m)$, while the ECGA and the ClusterMI have a scaling behavior of $\Theta(m^{1.1} \ln m)$. For instance for $\ell = 256$ or $m = 64$ the minimal population size is $P = 60$ for LTGA, while it is well above $P > 10000$ for ECGA and ClusterMI. Of course, that does not mean that LTGA requires less functions evaluations. To generate a single solution for the next population, LTGA traverses the entire linkage tree and evaluates a new offspring couple at each internal node where the offspring is different from the parents. Actually, in this experiment the overall number of function evaluations are similar for LTGA, ECGA, DSMGA, and ClusterMI.

4.2 Nearest Neighbor NK-Landscape with Tunable Overlap

In the previous section we investigated the ability of the LTGA to learn what problem variables should be linked together. Deceptive trap functions are ideal benchmark functions to test this. In this section we want to investigate how the LTGA deals with problems having overlap between the different subproblems. Therefore, we look at the performance on NK-landscape problems with nearest neighbor interactions and tunable overlap (nn-NK) [8][10]. A big advantage of nn-NK problems above the more general NK problems is that we can compute the global optimal solution with dynamic programming when using the knowledge of the position of the subproblems [8]. The LTGA of course does not have access to this positional information, its task is to learn the linkage from the current population of solutions. The nn-NK are interesting as benchmark functions because we can tune the amount of overlap between the subproblems and see whether the LTGA is still successful in finding the global optimum. Formally, a nn-NK function is defined by its length (ℓ), the size of the subproblems (k), the amount of overlap between the subproblems (o), and the number of subproblems (m). The first subproblem is defined at the first k string positions. The second subproblem is defined at the last o positions of the first subproblem and the next $(k - o)$ positions. All remaining subproblems are defined in a similar way. As an example, a nn-NK problem with $\ell = 65$, $k = 5$, $0 = 3$, and $m = 31$ has the following positions of the subproblems: (0 1 2 3 4)(2 3 4 5 6)(4 5 6 7 8)...(56 57 58 59 60)(58 59 60 61 62)(60 61 62 63 64). The relationship between the problem variables is $\ell = k + (m - 1)(k - o)$.

Table 2. Results for fixed length ($\ell = 65$) nn-NK problem with varying overlap

Overlap	Blocks	PopSize	Evals Q1	Evals Q2	Evals Q3	Runtime Ratio
1	16	180	123394	141147	165765	1
2	21	230	191619	224935	243720	1.5
3	31	240	217810	249200	265669	2
4	61	240	179407	220736	235499	2

Table 3. Results for fixed overlap ($o = 4$) nn-NK problem with varying length

Length	Blocks	PopSize	Evals Q1	Evals Q2	Evals Q3	Evals Ratio	Runtime Ratio
20	16	70	1719	4154	5613	1	1
40	36	140	38400	54021	76181	13	15
60	56	220	158549	182514	210568	44	100
80	76	380	393156	478951	540138	115	350
100	96	600	936167	1013846	1120435	244	1000

Table 2 and 3 show the number of function evaluations of the first hitting time and the growth ratio. Table 2 fixes the problem length $\ell = 65$ and varies the overlap $o \in \{1, 2, 3, 4\}$, or equivalently the number of blocks $m \in \{16, 21, 31, 61\}$. Table 3 fixes the overlap $o = 4$ and varies the problem length $\ell \in \{20, 40, 60, 80, 100\}$, or equivalently the number of blocks $m \in \{16, 36, 56, 76, 96\}$. Both tables show the minimal population size required to find the global optimum in at least 24 out of 25 independent runs. To determine this value we start with a population that is sufficiently large to reliably find the global optimum. Next we iteratively decrease the population size time by 10, until we encounter the first size where the LTGA no longer finds the global optimum at least 24 out of 25 runs. For each independent run a different random nn-NK problem is generated. We also run a bitwise hillclimber to each random initial solution. The LTGA thus learns its first linkage tree from a population of local optima. Thereafter, the hillclimber is not applied anymore.

In Table 2 it can be seen that for a fixed length problem the population size, the number of function evaluations, and the CPU runtime, hardly varies as a function of the overlap (especially for the overlap values $o =\in \{2, 3, 4\}$). Table 3 shows how the population size, the number of function evaluations, and the growth ratio of the number of function evaluations and the CPU runtime varies with increasing problem length (or increasing block number). A least squares fit indicates a scaling behavior of $\Theta = (\ell^{1.5} \ln \ell)$ for the minimal population size, $\Theta = (\ell^{3.1} \ln \ell)$ for the median number of function evaluations, and $\Theta = (\ell^{4.4} \ln \ell)$ for the corresponding CPU runtime.

To investigate the use of learning the linkage tree we have also run the LTGA without measuring the distance between clusters. So whenever the distance measure D is called to compute a value, we simply return a random number in [0...1]. This basically replaces our linkage guided crossover by uniform random crossover masks. Almost none of these runs ever found the global optimum which shows that learning the linkage in a hierarchical tree is beneficial even for problems with overlapping building blocks.

5 Conclusion

We have introduced the Linkage Tree Genetic Algorithm (LTGA). A linkage tree is the tree obtained by a hierarchical clustering procedure using a distance measure that represents the linkage between the problems variables or between clusters of problem variables. Each generation, the LTGA builds a new linkage tree from the current population. To generate new offspring solutions, the LTGA chooses two parent solutions and traverses the entire linkage tree starting from the root. At each tree node the LTGA recombines the two parent solutions according to the crossover mask specified by the clustering at that particular node. Whenever one of the two offspring has a better fitness score than both parents, the offspring pair replaces the parent pair, and the LTGA continues its traversal of the tree, now crossing the offspring pair. If none of the offspring improves on both its parents, LTGA simply continues with the parent pair. When the entire linkage tree has been visited, the best solution from the current pair is copied to the population of the next generation.

In this paper we have used as distance measure the normalized variation of information metric. This metric is based on mutual information between items and cluster of items, and exploits the grouping property of mutual information to build a hierarchical or nested set of clusters. Experimental results for fully deceptive functions and nearest neighbor NK-landscape problems with tunable overlap show that the LTGA can solve these hard functions efficiently without knowing the actual position of the linked variables on the problem representation.

References

1. Chen, Y.-P., Lim, M.-H. (eds.): Linkage in Evolutionary Computation. Studies in Computational Intelligence, vol. 157. Springer, Heidelberg (2008)
2. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Whitley, L.D. (ed.) Proceedings of the Second Workshop on Foundations of Genetic Algorithms, pp. 93–108. Morgan Kaufmann, San Francisco (1993)
3. Duque, T.S., Goldberg, D.E.: A new method for linkage learning in the ECGA. In: Proceedings of the 11th annual conference on Genetic and Evolutionary Computation, pp. 1819–1820. ACM, New York (2009)
4. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer Academic Publishers, Dordrecht (2002)
5. Harik, G.R., Lobo, F.G., Sastry, K.: Linkage learning via probabilistic modeling in the extended compact genetic algorithm. In: Pelikan [9], pp. 39–61
6. Hauschild, M., Pelikan, M.: Network crossover performance on NK landscapes and deceptive problems. Technical Report MEDAL No. 2010003, Missouri Estimation of Distribution Algorithms Laboratory (January 2010)
7. Kraskov, A., Stögbauer, H., Andrzejak, R.G., Grassberger, P.: Hierarchical clustering using mutual information. EuroPhysics Letters 70(2), 278–284 (2005)
8. Pelikan, M., Sastry, K., Butz, M.V., Goldberg, D.E.: Performance of evolutionary algorithms on random decomposable problems. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 788–797. Springer, Heidelberg (2006)
9. Pelikan, M., Sastry, K., Cantú-Paz, E. (eds.): Scalable Optimization via Probabilistic Modeling. Studies in Computational Intelligence, vol. 33. Springer, Heidelberg (2006)
10. Pelikan, M., Sastry, K., Goldberg, D.E., Butz, M.V., Hauschild, M.: Performance of evolutionary algorithms on NK landscapes with nearest neighbor interactions and tunable overlap. In: Raidl, G. (ed.) Proceedings of the 11th annual conference on Genetic and Evolutionary Computation, pp. 851–858. ACM, New York (2009)
11. Stonedahl, F., Rand, W., Wilensky, U.: Crossnet: a framework for crossover with network-based chromosomal representations. In: Proceedings of the 10th annual conference on Genetic and Evolutionary computation, pp. 1057–1064. ACM, New York (2008)
12. Thierens, D., Goldberg, D.E.: Mixing in genetic algorithms. In: Proceedings of the International Conference on Genetic Algorithms (ICGA), pp. 38–47 (1993)
13. Yu, T.-L., Goldberg, D.E.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: Cattolico, M. (ed.) Proceedings of the 8th annual conference on Genetic and Evolutionary computation, pp. 1385–1392. ACM, New York (2006)
14. Yu, T.-L., Sastry, K., Goldberg, D.E., Pelikan, M.: Population sizing for entropy-based model building in discrete estimation of distribution algorithms. In: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation, pp. 601–608. ACM, New York (2007)

An Analysis of the XOR Dynamic Problem Generator Based on the Dynamical System

Renato Tinós[1] and Shengxiang Yang[2]

[1] Department of Physics and Mathematics, FFCLRP, University of São Paulo (USP)
14040-901, Ribeirão Preto, S.P., Brazil
rtinos@ffclrp.usp.br

[2] Department of Information Systems and Computing, Brunel University
Uxbridge, Middlesex UB8 3PH, U.K.
shengxiang.yang@brunel.ac.uk

Abstract. In this paper, we use the exact model (or dynamical system approach) to describe the standard evolutionary algorithm (EA) as a discrete dynamical system for dynamic optimization problems (DOPs). Based on this dynamical system model, we analyse the properties of the XOR DOP Generator, which has been widely used by researchers to create DOPs from any binary encoded problem. DOPs generated by this generator are described as DOPs with permutation, where the fitness vector is changed according to a permutation matrix. Some properties of DOPs with permutation are analyzed, which allows explaining some behaviors observed in experimental results. The analysis of the properties of problems created by the XOR DOP Generator is important to understand the results obtained in experiments with this generator and to analyze the similarity of such problems to real world DOPs.

1 Introduction

The study of evolutionary algorithms (EAs) for dynamic optimization problems (DOPs) has attracted a rapidly growing interest in recent years due to its importance to real world applications of EAs, where, often, new solutions should be found in short time after a change in the problem [2]. Most researches on EAs for DOPs focus on experimental investigation, and very few investigate the theory behind DOPs [8,9,7,1,3,6]. In [7], the standard genetic algorithm (GA) with mutation and selection is investigated in DOPs with regular changes (see Section 3) based on the dynamical system approach (or exact model) of the GA [11]. Despite demanding a large number of equations to track all possible solutions represented by the individuals of the GA, the use of the exact model is attractive as it allows a complete description of the population dynamics [5].

In this paper, we use the dynamical system approach to analyze the XOR DOP Generator [12,14]. In the XOR DOP Generator, DOPs are created from any binary encoded stationary problem, which allows comparing different algorithms in environments with different properties. This paper investigates the properties of the problems generated by the XOR DOP Generator, which is relevant in order to understand the results obtained in the experiments with this generator and to analyze the similarity of such problems to real world problems.

2 Exact Model of the GA in Stationary Environments

In the exact model proposed by Vose [11], the standard GA is described as a discrete dynamical system [5]. In a GA with binary codification, an individual of a population codifies a possible solution $\mathbf{x} \in \{0,1\}^l$. In the exact model, all possible solutions are represented in an n-dimensional discrete space χ, where each possible solution is enumerated as $\{0, 1, \ldots, n-1\}$ and $n = 2^l$. A population is then defined by an n-dimensional vector $\mathbf{p}$, where each element defines the proportion of each possible solution in the population with size N. As the sum of the elements of $\mathbf{p}$ is equal to 1, population vectors can be described as members of a simplex Λ. This way, the population evolution can be described as a trajectory in the simplex and population vectors can be used to describe the probability distribution of the individuals in the search space. Thus, a generational operator $\mathcal{G} : \Lambda \rightarrow \Lambda$ can be defined. The vector $\mathcal{G}(\mathbf{p})$ describes the expected next population [11], i.e., the average over all possible populations of the next generation with variance inversely proportional to the population size N. In the limit $N \rightarrow \infty$ (infinite population), the variance goes to zero, and the evolution in the stationary case is deterministically described by the trajectory $\mathbf{p}$, $\mathcal{G}(\mathbf{p})$, $\mathcal{G}^2(\mathbf{p})$, ... In this way, in generation t for the stationary case, the expected population vector for the infinite population model is given by:

$$\mathbf{p}_t = \mathcal{G}^t(\mathbf{p}_0), \tag{1}$$

where $\mathbf{p}_0$ is the initial population vector. When fitness proportional selection and flip mutation are employed, the generational operator can be written as:

$$\mathcal{G}(\mathbf{p}) = \frac{UF\ \mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}, \tag{2}$$

where $F = \mathrm{diag}(\mathbf{f})$ is a diagonal matrix generated from the fitness vector $\mathbf{f}$ and U is the mutation matrix. The analysis of Eq. (2) can provide insights in understanding the behavior of the GA. The fixed points of $\mathcal{G}$, i.e., points where $\mathcal{G}(\mathbf{p}) = \mathbf{p}$, are given by the eigenvectors of UF. For each eigenvector $\mathbf{p}$, an eigenvalue $\mathbf{f}^{\mathrm{T}}\mathbf{p}$, corresponding to the average fitness of $\mathbf{p}$, can be computed. As UF has only positive values, there is only one eigenvector in Λ, corresponding to the eigenvalue with the largest absolute value [5]. Then, all trajectories in Λ converge to this fixed point, i.e. the system is asymptotically stable [11]. The remaining eigenvectors are not properly fixed points, as, for example, they can lie outside the simplex. However, they play an important role in the evolutionary process as they can change the trajectory in the simplex and can create metastable states that can trap finite populations for several generations [4].

3 Dynamic Optimization Problems

A DOP is an optimization problem where at least one change occurs during the evolutionary process. When a change occurs, the generational operator $\mathcal{G}$ is altered and at least one possible trajectory realized by the population in the simplex Λ is affected. It can be observed that not all modifications in the generational operator can be described as a change. Another important observation is that a change does not necessarily imply a modification in the population or in its current trajectory. For example, if the change does not modify the current trajectory of the population to the fixed point, no effect will be observed in the evolutionary process. The same occurs if the population has converged to the fixed point and this one is not modified by the change.

As the generation operator is modified after a change, Eq. (1) is not valid anymore for every generation t in a DOP. If we consider that changes occur only between the application of two consecutive generational operators, the following equation is valid for the infinite population case:

$$\mathbf{p}_t = \mathcal{G}_t(\mathbf{p}_{t-1}), \tag{3}$$

where $\mathcal{G}_t$ is the generational operator in generation $t \geq 1$.

A series of generational operations between two consecutive changes is called here a change cycle. The first change cycle begins in the first generation of the evolutionary process and ends one generation before the first change, while the last change cycle begins in the generation after the last change and ends until the last generation of the evolutionary process.

The change cycle duration d_e is the number of consecutive generations in change cycle e. If change cycle e begins at generation t_e, then

$$\mathcal{G}_{t_e} = \mathcal{G}_{t_e+1} = \mathcal{G}_{t_e+2} = \ldots = \mathcal{G}_{t_e+d_e-1}, \tag{4}$$

where $d_e > 0$. In abuse of notation, we define now $\mathcal{G}_e$ as the generational operator in change cycle e. In this way, for the infinite population case, the population in generation t is now given by:

$$\mathbf{p}_t = \mathcal{G}_e^{(t-\sum_{i=1}^{e-1} d_i)} \mathcal{G}_{e-1}^{d_{e-1}} \ldots \mathcal{G}_3^{d_3} \mathcal{G}_2^{d_2} \mathcal{G}_1^{d_1}(\mathbf{p}_0), \tag{5}$$

where $e > 0$. It can be observed that a DOP can be viewed as a sequence of stationary processes, where the initial population in the i-th change cycle is the last population generated in the change cycle $i-1$. The minimum value of d_i is one generation, which is the case where the generational operator is modified just one generation after the prior change, while the maximum value of d_i is equal to the index of the current generation, which is the case where the problem is stationary (until the current generation) and Eq. (5) reproduces Eq. (1).

In this paper, we are interested in a class of dynamic problems defined here as DOPs with permutation, which is defined below.

Definition 1. *A* ***DOP with permutation*** *is a DOP where the fitness landscape in change cycle* $e-1$ *is modified according to a permutation matrix, i.e.,*

$\mathbf{f}_e = \sigma_{\mathbf{k}_e}\mathbf{f}_{e-1}$, *where* $\sigma_{\mathbf{k}_e}$ *is a permutation matrix mapping the element at position* $\mathbf{i}$ *of the vector* $\mathbf{f}_{e-1}$ *to the element at position* $\mathbf{i} \oplus \mathbf{k}_e$ *of the vector* $\mathbf{f}_e$*, where* $\oplus$ *is the bitwise exclusive-or (XOR), or addition modulo 2, operator. The vector* $\mathbf{i} \in \{0,1\}^l$ *indicates the position of the element in the fitness vector. The vector* $\mathbf{k}_e \in \{0,1\}^l$ *controls the permutation of the elements of the fitness vector.*

In a DOP with permutation, the fitness values are preserved in the search space, i.e., they are only resorted. In [7], DOPs with regular changes, which are a special subset of DOPs with permutation (Definition 1) where the transitional rule is deterministic and belongs to a permutation group where $\sigma_{\mathbf{k}_{e+t}} = (\sigma_{\mathbf{k}_e})^t$ for $t \geq 0$, are defined. As a consequense, in DOPs with regular changes, the fixed points can be computed and the asymptotic states can be then analyzed [7].

4 The XOR DOP Generator

The XOR DOP Generator [14] can generate DOPs from any binary encoded problem. In the XOR DOP Generator, given a stationary problem with fitness function $f(\mathbf{x}_t)$ and the solution $\mathbf{x}_t \in \{0,1\}^l$, the fitness function $f_e(\mathbf{x}_t)$ of an environment, which is periodically changed every τ generations, is computed by:

$$f_e(\mathbf{x}_t) = f\big(\mathbf{x}_t \oplus \mathbf{m}_e\big), \tag{6}$$

where t is the generation index, $e = \lceil t/\tau \rceil$ is the change cycle index, and $\mathbf{m}_e$ is a binary mask for change cycle e, which is incrementally generated by:

$$\mathbf{m}_e = \mathbf{m}_{e-1} \oplus \mathbf{r}_e, \tag{7}$$

where $\mathbf{r}_e$ is a binary template randomly created for change cycle e containing $\lfloor \rho \times l \rfloor$ ones, and $\{\rho \in \mathbb{R} \mid 0.0 \leq \rho \leq 1.0\}$ controls the degree of change for the DOP. If $\rho = 0.0$, the problem stays stationary, while if $\rho = 1.0$, the extreme fitness landscape change in the sense of Hamming distance occurs. For the first change cycle, $\mathbf{m}_1$ is equal to the zero vector.

The main characteristic of the XOR DOP Generator is that each individual of the current population is moved to a new location in the fitness landscape before being evaluated [10]. Instead of evaluating the fitness of the individual at $\mathbf{x}_t$, the fitness is evaluated at $\mathbf{x}_t \oplus \mathbf{m}_e$. It can be observed that the XOR DOP Generator produces DOPs with changes in the fitness landscape. Based on the XOR DOP Generator properties, the following theorem is proposed.

Theorem 1. *The XOR DOP Generator produces DOPs with permutation.*

Proof : It can be observed that only the fitness vector is modified by Eq. (6). The fitness vector in change cycle $e > 1$ for a DOP generated by the XOR DOP Generator from a stationary problem with fitness function $f(\mathbf{x}_t)$ is given by:

$$\mathbf{f}_e = \begin{bmatrix} \mathbf{f}_{e(0)} \\ \mathbf{f}_{e(1)} \\ \vdots \\ \mathbf{f}_{e(n-1)} \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_{(0)} \oplus \mathbf{m}_e) \\ f(\mathbf{x}_{(1)} \oplus \mathbf{m}_e) \\ \vdots \\ f(\mathbf{x}_{(n-1)} \oplus \mathbf{m}_e) \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_{(0)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \\ f(\mathbf{x}_{(1)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \\ \vdots \\ f(\mathbf{x}_{(n-1)} \oplus \mathbf{m}_{e-1} \oplus \mathbf{r}_e) \end{bmatrix}, \tag{8}$$

where $\mathbf{x}_{(i)}$ is the i-th possible solution in the n-dimensional discrete space χ and $\mathbf{f}_{e(i)}$ is its respective fitness.

Defining the i-th solution in change cycle $e-1$ as $\mathbf{x}_{e-1(i)} = \mathbf{x}_{(i)} \oplus \mathbf{m}_{e-1}$, Eq. (8) can be written as:

$$\mathbf{f}_e = \begin{bmatrix} f(\mathbf{x}_{e-1(0)} \oplus \mathbf{r}_e) \\ f(\mathbf{x}_{e-1(1)} \oplus \mathbf{r}_e) \\ \vdots \\ f(\mathbf{x}_{e-1(n-1)} \oplus \mathbf{r}_e) \end{bmatrix} = \sigma_{\mathbf{r}_e} \begin{bmatrix} \mathbf{f}_{e-1(0)} \\ \mathbf{f}_{e-1(1)} \\ \vdots \\ \mathbf{f}_{e-1(n-1)} \end{bmatrix} = \sigma_{\mathbf{r}_e} \mathbf{f}_{e-1}, \tag{9}$$

where $\sigma_{\mathbf{r}_e}$ is a permutation matrix mapping the element at position $\mathbf{j}$ of the vector $\mathbf{f}_{e-1}$ to the element at position $\mathbf{j} \oplus \mathbf{r}_e$ of the vector $\mathbf{f}_e$. Equation (9) indicates that the fitness of the i-th solution in change cycle e is equal to the fitness of the i-th solution in change cycle $e-1$ moved according to the permutation $\mathbf{r}_e$. That is, the XOR DOP Generator produces DOPs with permutation. □

One can still observe that the XOR DOP Generator produces stationary environments for $\rho = 0.0$ and DOPs with regular changes for $\rho = 1.0$. In the latter case, the DOP switches between two environments. For $0.0 < \rho < 1.0$, the changes are not regular because the template $\mathbf{r}_e$ is randomly generated, and, as a consequence, the metastable points for the stationary environments generated for each template $\mathbf{r}_e$ are generally different.

As the DOP is viewed as a sequence of stationary environments, the analysis of how the fixed points and metastable states for each stationary environment are related can provide insights in understanding GA's behavior on the DOP generated by the XOR DOP Generator. Here, the state of the DOP in a change cycle corresponding to the fixed point in the respective stationary environment is called main metastable state. It is important to observe that the main metastable states are not fixed points of the DOP, as the problem changes and the population generally does not converge to a fixed point. However, the metastable states control the trajectory of the population during each change cycle.

In a DOP with permutation, the points of the search space in change cycle $e > 1$ are obtained by the permutation, according to $\sigma_{\mathbf{k}_e}$, of the points of the search space in change cycle $e-1$. As a consequense, the i-th eigenvector $\mathbf{p}_{e(i)}$ of $U_e F_e$ in change cycle e can be obtained by the permutation, according to $\sigma_{\mathbf{k}_e}$, of the respective eigenvector for the environment in change cycle $e-1$, i.e.,

$$\mathbf{p}_{e(i)} = \sigma_{\mathbf{k}_e} \mathbf{p}_{e-1(i)}. \tag{10}$$

Besides, the eigenvalues of $U_e F_e$ for two environments defined by change cycles $e-1$ and e are equal. As the metastable states of $\mathcal{G}_e$ are given by the eigenvectors of $U_e F_e$, the metastable states of $\mathcal{G}_e$ and $\mathcal{G}_{e-1}$ in a DOP with permutation are related by the permutation matrix $\sigma_{\mathbf{k}_e}$ (or $\sigma_{\mathbf{r}_e}$ for environments created by the XOR DOP Generator). Besides, the average fitness (eigenvalue) at the main metastable remains the same.

Theorem 2. *Consider the standard GA with mutation and fitness proportional selection is applied in: i) a DOP with permutation (Definition 1), where the* *duration and permutation matrix of change cycle e are, respectively, d_e and $\sigma_{\mathbf{k}_e}$; ii) in a stationary environment where the population is permuted according to the permutation matrix $\sigma_{\mathbf{k}_e}$ after every cycle $e = 1, 2, \ldots$ with duration d_e. If both evolutionary processes have the same initial population and parameters, and the fitness function in the first change cycle for the first process is equal to the fitness function in the second process, then the evolution of the population in the two processes is identical, i.e., the two evolutionary processes are equivalent.*

Proof: According to Eqs. (3) and (2), the population for the infinite population case in generation $t > 1$ for a DOP with fitness landscape changes is given by:

$$\mathbf{p}_t = \frac{UF_e \; \mathbf{p}_{t-1}}{\mathbf{f}_e^{\mathrm{T}} \mathbf{p}_{t-1}}. \tag{11}$$

For a change cycle $e > 1$ in a DOP with permutation (Definition 1), $\mathbf{f}_e = \sigma_{\mathbf{k}_e} \mathbf{f}_{e-1}$, and, as a consequense, $F_e = \sigma_{\mathbf{k}_e} F_{e-1} \sigma_{\mathbf{k}_e}$. Then, we can write Eq. 11 as:

$$\mathbf{p}_t = \frac{U\sigma_{\mathbf{k}_e} F_{e-1} \sigma_{\mathbf{k}_e} \; \mathbf{p}_{t-1}}{\mathbf{f}_{e-1}^{\mathrm{T}} \sigma_{\mathbf{k}_e} \mathbf{p}_{t-1}}. \tag{12}$$

Defining $\mathbf{q}_t = \sigma_{\mathbf{k}_e} \mathbf{p}_t$ and considering that U and $\sigma_{\mathbf{k}_e}$ commute, then:

$$\mathbf{q}_t = \frac{UF_{e-1} \; \mathbf{q}_{t-1}}{\mathbf{f}_{e-1}^{\mathrm{T}} \mathbf{q}_{t-1}}. \tag{13}$$

It can be observed that Eqs. (11) and (13) are similar. If, after the change e, we transform the final population at the last generation of change cycle $e - 1$ according to $\mathbf{q}_t = \sigma_{\mathbf{k}_e} \mathbf{p}_t$, then, we can use Eq. (13) with the same fitness landscape of change cycle $e - 1$ to reproduce the dynamics of the population in the infinite population case for change cycle e. □

As a consequence of Theorem 2, the XOR DOP Generator can be simplified. Instead of computing the fitness of each individual $\mathbf{x}_t$ of the population at the position $\mathbf{x}_t \oplus \mathbf{m}_e$ in every generation, each individual of the initial population in change cycle e is moved to $\mathbf{x}_t = \mathbf{x}_t \oplus \mathbf{r}_e$, i.e., the population is moved only one time, and the fitness is computed as $f(\mathbf{x}_t)$, like in the stationary environment. In this way, the complexity of the procedure is reduced from $O(lNd_e)$ to $O(lN)$.

5 Experimental Study

In this section, we present simulations for the evolution of the standard GA with mutation and fitness proportional selection in a DOP created by the XOR DOP Generator from a deceptive fitness function defined by:

$$f(\mathbf{x}) = \begin{cases} l, & \text{if } u(\mathbf{x}) = l \\ (l-1) - u(\mathbf{x}), & \text{otherwise,} \end{cases} \tag{14}$$

where $u(\mathbf{x})$ is the unitation function of a binary vector $\mathbf{x}$ of length l. This function has one global optimum and one local optimum. In the simulations presented

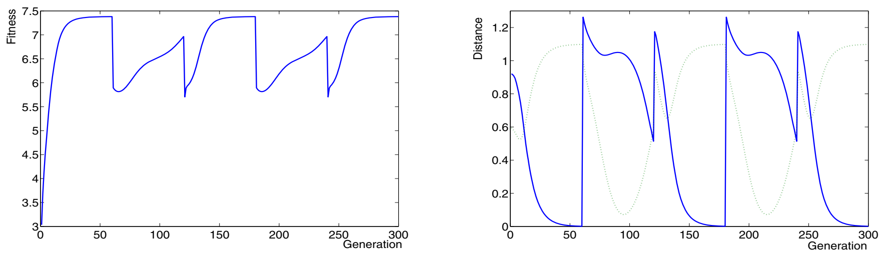
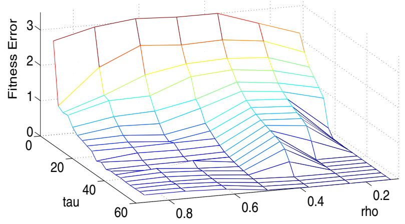


Fig. 1. Mean fitness of the population (left) and distance to the current first (solid) and second metastable states (dotted) for five change cycles with $\tau = 60$ and $\rho = 0.875$

in this section, $l = 8$, the mutation rate is 0.01, and the initial population ($\mathbf{p}_0$) is uniformly distributed. For all simulations, Eq. 11 is employed to generate the population vector $\mathbf{p}(t)$, i.e., the exact model with infinite population is employed in the simulations in order to generate the expected next population during the evolutionary process. In the simulations, the problem changes, according to $\mathbf{f}_e = \sigma_{\mathbf{r}_e} \mathbf{f}_{e-1}$, every τ generations with change degree ρ. Twenty values of τ (from $\tau = 3$ to $\tau = 60$ with a step size 3) and seven values of ρ (from $\rho = 0.125$ to $\rho = 0.875$ with a step size 0.125) are considered. In this way, 140 simulations were executed, one for each pair of τ and ρ. Each evolutionary process is simulated for 30 change cycles of the infinite population model.

Figure 1 shows a simulation with $\rho = 0.875$ and $\tau = 60$, where the mean fitness of the population during the evolution and the Euclidean distance between the population in the current generation and the two eigenvectors with the largest eigenvalues are presented. The first eigenvector corresponds to the current main metastable state (where the number of individuals of the population at the global optimum is larger than the number of individuals at any other place), while the second eigenvector is the metastable state with the second largest eigenvalue (where the number of individuals of the population at the local optimum is larger than the individuals at any other place). It can be observed that, in some change cycles, the population goes to the neighborhood of the second metastable state and, after some generations, goes to the main metastable state. When Eq. (13) is used, i.e., evolution in a stationary environment where the population is permuted according to the permutation matrix $\sigma_{\mathbf{k}_e}$ after every cycle e with duration $d_e = 60$, the same graphics presented in Fig. 1 are obtained if the same parameters are employed (those graphics are not shown here), as stated by Theorem 2.

Figure 2 presents the results for all simulations. The first graph shows the value of $f_{opt} - f(\mathbf{p}_e)$ averaged over all change cycles e, where f_{opt} is the current mean value of fitness in the main metastable state and $f(\mathbf{p}_e)$ is the mean fitness of the population $\mathbf{p}_e$ in the end of change cycle e. The second graph presents the respective mean distance between the current main metastable state and the state $\mathbf{p}_e$ in the end of change cycle e. From Fig. 2, some observations can be made. When τ is close to 60 generations, i.e., in slow changing environments, the



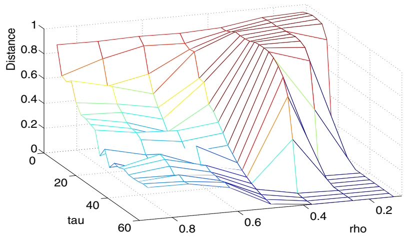


Fig. 2. Fitness error (left) and distance to the current main mestastable (right) in the simulations for different τ and ρ. The values are relative to the average (over 30 change cycles) obtained by the population vector in the generation before the change

population always reaches the main metastable state after changes with small degree of change ρ. When τ is large, there is enough time for the population to go from the neighborhood of the main metastable state (where most of the population is at the global optimum) in change cycle $e-1$ to the neighborhood of the main metastable state in change cycle e. As a consequence, the fitness error in the end of each change cycle is zero when τ is large and ρ is small (see Fig. 1). In the XOR DOP Generator, the parameter ρ controls the degree of change. As ρ controls the percentage of changed bits from template $\mathbf{r}_{e-1}$ to template $\mathbf{r}_e$, the hamming distance between $\mathbf{r}_{e-1}$ and $\mathbf{r}_e$ is $h(\mathbf{r}_e, \mathbf{r}_{e-1}) = \lfloor \rho \times l \rfloor$. In this way, larger ρ imply larger hamming distance between the optima in two consecutive change cycles and in longer trajectories of the population in the simplex, and, thus, more time to reach the neighborhood of the main metastable point.

However, it can be observed that a higher degree of modification in the fitness landscape (larger ρ) does not necessarily imply a worse performance of the GA in the DOP for medium and small τ. One can observe that for medium and small τ, the simulations with $\rho = 0.375$ presented worse performance than those for larger ρ. This behavior can be found in experiments with the XOR DOP Generator for different algorithms (for example, see [13]). The performance of the GA is related to trajectories of the population in the simplex, and the trajectories are related to the fitness vector and the transformation operators. In a medium velocity or fast changing environment, generally, when the population reaches the neighborhood of the main metastable point in change cycle $e-2$, the population after the change is closer to the second metastable state in the next change cycle when ρ is large. In this case, the population does not have enough time to be closer to the new main metastable state neighborhood in change cycle $e-1$ than to the old main metastable neighborhood. However, when the problem changes again, the population is close to the neighborhood of the main metastable state in change cycle e for ρ close to 1. The mean Hamming distance of template $\mathbf{r}_e$ between two change cycles, which is given by $\bar{h}(\mathbf{r}_e, \mathbf{r}_{e-2}) = 2l(\rho - \rho^2)$, explains the behavior of the GA in this case. It can be observed that the mean fitness generally alternates between two different values for larger ρ and medium or small τ (see, for example, Fig. 1). One can observe that the values of distance in Fig. 2 are higher for ρ close to one than for ρ close to zero, as in part of the change cycles, the population remains in the neighborhood of the second metastable state for larger ρ.

Two observations can be made for the previous analysis. First, a higher degree of modification (ρ) in the templates $\mathbf{r}_e$ does not necessarily imply a worse performance of the GA. This result has been observed in several experiments with the XOR DOP Generator (for example, see [13]). The performance of the GA is related to the trajectories of the population in the simplex, which makes more complex the analysis of the performance of the algorithms. Second, the metrics used to compare the algorithms in DOPs cannot be adequate for some problems. For example, in the problem investigated here, an algorithm that keeps the population close to the second metastable neighborhood for a high degree of change in a fast changing environment can have higher mean fitness than an algorithm that allows the population escaping from the local optima, but does not have enough time to reach the main metastable state neighborhood.

6 Conclusion and Future Work

In this paper, DOPs are defined based on the dynamical system (or exact model) [11] and the class of DOPs with permutation is defined. Such definition, and others that can be defined based on the same approach, can be useful to classify real world DOPs and, hence, allow a systematic analysis of such problems based on the properties of each class. Here, the XOR DOP Generator, which allows creating DOPs from any binary encoded stationary problem, is analyzed based on the dynamical system approach and the definition presented in Section 3. In this paper, the optimization process of the GA on the DOP is viewed as a sequence of evolutionary processes, each one described as a stationary optimization problem, where the initial population in a change cycle with duration $d_e \geq 1$ is given by the last population in the previous change cycle. In the problems generated by the XOR DOP Generator, the duration of all change cycles is equal and the fitness vector of the problem in change cycle $e > 1$ is related to the fitness vector in change cycle $e-1$ by a random template $\mathbf{r}_e$. Thus, a problem created by the XOR DOP Generator is identified as a DOP with permutation (Definition 1), where the fitness vector changes according to $\mathbf{f}_e = \sigma_{\mathbf{r}_e}\mathbf{f}_{e-1}$.

When the standard GA with proportional fitness selection and mutation is applied to a DOP with permutation, the eigenvectors of the fixed point equation between two consecutive change cycles are related by the permutation matrix $\sigma_{\mathbf{k}_e}$ (or $\sigma_{\mathbf{r}_e}$ in DOPs created by the XOR DOP Generator). This way, the metastable points in change cycle e can be obtained by the permutation (according to $\sigma_{\mathbf{k}_e}$) of the same points in change cycle $e-1$, and the evolution in a DOP with permutation is equivalent to that in a stationary environment where the population is permuted by the permutation matrix $\sigma_{\mathbf{k}_e}$ after each cycle $e = 1, 2, \ldots$ with duration d_e. Hence, the XOR DOP Generator can be simplified by moving the initial population of a change cycle instead of computing the fitness function of each individual in each generation in a new position. In this paper, the influence of the parameter ρ in the XOR DOP Generator is also analyzed, and the results obtained in experiments related in the literature, where the worst performance for some algorithms are obtained for medium ρ, is explained.

It can be observed that algorithms exploring the properties described on the analysis of the XOR DOP Generator can be proposed. However, it is not clear if such algorithms are useful in real world DOPs. To answer this question, the real world DOPs identified as permutation DOPs should be described, which should allow the use of the XOR DOP Generator to reproduce such problems. In this way, a very relevant future investigation is to analyze real world DOPs, to classify them according to their properties, and to develop DOPs generators based on the identified class of DOPs. Another relevant future work is to analyze algorithms proposed for DOPs, e.g., GA with hypermutation and GA with random immigrants, according to the dynamical system approach.

References

1. Arnold, D.V., Beyer, H.-G.: Random dynamics optimum tracking with 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. 3–12. Springer, Heidelberg (2002)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer, Dordrecht (2001)
3. Droste, S.: Analysis of the (1+1) EA for a dynamically changing onemax-variant. In: 2002 Congr. on Evol. Comput., pp. 55–60 (2002)
4. Van Nimwegen, E., Crutchfield, J.P., Mitchell, M.: Finite populations induce metastability in evolutionary search. Physics Letters A 229(3), 144–150 (1997)
5. Reeves, C.R., Rowe, J.E.: Genetic algorithms - principles and perspectives: a guide to GA theory. Kluwer Academic Publishers, Dordrecht (2003)
6. Rohlfshagen, P., Lehre, P.K., Yao, X.: Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In: 2009 Genetic and Evol. Comp. Conf., pp. 1713–1720 (2009)
7. Ronnewinkel, C., Wilke, C.O., Martinetz, T.: Genetic algorithms in time-dependent environments. In: Kallel, L., Naudts, B., Rogers, A. (eds.) Theor. Aspects of Evol. Comp., pp. 263–288. Springer, Heidelberg (2001)
8. Rowe, J.E.: Finding attractors for periodic fitness functions. In: 1999 Genetic and Evol. Comp. Conf., pp. 557–563 (1999)
9. Rowe, J.E.: Cyclic attractors and quasispecies adaptability. In: Kallel, L., Naudts, B., Rogers, A. (eds.) Theor. Aspects of Evol. Comp., pp. 251–259. Springer, Heidelberg (2001)
10. Tinós, R., Yang, S.: Continuous dynamic problem generators for evolutionary algorithms. In: 2007 Congr. on Evol. Comput., pp. 236–243 (2007)
11. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. The MIT Press, Cambridge (1999)
12. Yang, S.: Non-stationary problem optimization using the primal-dual genetic algorithm. In: 2003 Congr. on Evol. Comput., pp. 2246–2253 (2003)
13. Yang, S., Tinós, R.: A hybrid immigrants scheme for genetic algorithms in dynamic environments. Int. J. of Autom. and Comput. 4(3), 243–254 (2007)
14. Yang, S., Yao, X.: Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput. 9(11), 815–834 (2005)

The Role of Degenerate Robustness in the Evolvability of Multi-agent Systems in Dynamic Environments

James M. Whitacre[1], Philipp Rohlfshagen[1], Axel Bender[2], and Xin Yao[1]

[1] CERCIA, School of Computer Science, University of Birmingham
Birmingham B15 2TT, United Kingdom
{j.m.whitacre,p.rohlfshagen,x.yao}@cs.bham.ac.uk

[2] Land Operations Division, Defence Science and Technology Organisation
Edinburgh, Australia
axel.bender@dsto.defence.gov.au

Abstract. It has been proposed that degeneracy plays a fundamental role in biological evolution by facilitating robustness and adaptation within heterogeneous and time-variant environments. Degeneracy occurs whenever structurally distinct agents display similar functions within some contexts but unique functions in others. In order to test the broader applicability of this hypothesis, especially to the field of evolutionary dynamic optimisation, we evolve multi-agent systems (MAS) in time-variant environments and investigate how degeneracy amongst agents influences the system's robustness and evolvability. We find that degeneracy freely emerges within our framework, leading to MAS architectures that are robust towards a set of similar environments and quickly adaptable to large environmental changes. Detailed supplementary experiments, aimed particularly at the scaling behaviour of these results, demonstrate a broad range of validity for our findings and suggest that important general distinctions may exist between evolution in degenerate and non-degenerate agent-based systems.

1 Introduction

The field of evolutionary dynamic optimisation (e.g., [3,8]) is concerned with the application of evolutionary algorithms (EA) to dynamic optimisation problems (DOP). In DOP, conditions vary frequently, and optimisation methods need to adapt their proposed solutions to time-dependent contexts (tracking of the optimum). EA are believed to be excellent candidates to tackle this particular class of problems, partially because of their correspondence with natural systems – the archetypal systems exposed to inherently dynamic environments.

Here we examine the properties that are believed to facilitate the positive relationship between *mutational robustness* and *evolvability* that takes place in natural evolution. In computational intelligence, these issues relate directly to concepts of fitness landscape neutrality and the search for high-quality solutions. Fitness landscapes are used extensively in the field of combinatorial optimisation to describe the structural properties of the problem to be optimised. The fitness landscape results directly from the choice of representation as well as the choice of search operators. Subsequently, different representations lead to different fitness landscapes and hence to problems of different difficulty (see [9] for an overview). Much research has focused on developing and analysing different problem representations. Inspired by earlier developments in theoretical biology, neutrality – the concept of mutations that do not affect system fitness – has been integrated into problem representations using various approaches such as polyploidy (see [1,18,10,7,6]). However, there are theoretical reasons as well as some experimental evidence to suggest that only particular representations of neutrality will support the discovery of novel adaptations. Edelman and Gally have proposed that degeneracy, a common source of stability against genetic mutations and environmental changes, creates particular types of neutrality that increase access to distinct heritable phenotypes and support a system's propensity to adapt [5]. Before describing Edelman and Gally's hypothesis on the mechanics of evolution, we first define some biological concepts – evolvability, robustness, redundancy and degeneracy – with special emphasis on their meaning to optimisation.

Evolvability in biology is concerned with the inheritance of new and selectively beneficial phenotypes. It requires 1) phenotypic variety (PV), i.e. an ability to generate distinct heritable phenotypes, and 2) that some of this phenotypic novelty can be transformed into positive adaptations [16,17,14]. Similarly, evolvability in optimisation describes an algorithm's ability to sample solutions of increasing quality.

Robustness has several meanings in optimisation that mostly relate to the maintenance of adequate fitness values. In robust optimisation, robustness refers to the insensitivity of a solution's fitness to minor alterations of its decision variables. In dynamic optimisation, robustness is often defined as the insensitivity of a solution's fitness to perturbations in the objective function's parameters over time.

In biology, redundancy and degeneracy often contribute to the robustness of traits [5]. Redundancy means 'redundancy of parts' and refers to identical components (e.g. proteins, people, vehicles, mechanical tools) with identical functionality (see Fig. 1b). Redundant components can often substitute for one another and thus contribute towards a 'fail-safe' system. In contrast, degeneracy arises when similarities in the functions of components are only observed for certain conditions. In particular, while diverse components sometimes can be observed performing similar functions (many-to-one mapping), components are also functionally versatile (one-to-many mapping) with the actual function performed at a given time being dependent on the context. For degeneracy to arise, a component must have multiple context-induced functions of which some (but not all) are also observed in another component type.

In a landmark paper [5], Edelman and Gally present numerous examples where degeneracy contributes to the stability of biological traits. They hypothesize that degeneracy may also fundamentally underpin evolvability by supporting the generation of PV. In particular, degenerate components stabilize conditions where they are functionally compensatory, however they also retain unique structural characteristics that lead to a multiplicity of distinct functional responses outside of those conditions. These differential responses can occasionally have distinct phenotypic consequences [16] that may emerge as selectively relevant adaptations when presented with the right environment, cf [5,16,14,15]. Edelman and Gally's hypothesis describes degeneracy as a mechanistic facilitator of both robustness and adaptation that, in principle, could be applied outside biological contexts [16]. As described in [5,17], degeneracy is ubiquitous throughout natural systems that undergo parallel problem-solving. Yet until recently, it has not informed the design and development of nature-inspired algorithms. Here we present evidence that degeneracy may provide a new (representational) approach to improve evolvability throughout EA execution in both static and dynamic environments. This approach could be applicable for many problems that are naturally modeled by systems with autonomous and functionally versatile agents that must survive within a heterogeneous environment.

2 The Role of Degeneracy in Evolution

When considering discrete local changes (mutations) in the decision variables of a single solution, the number of distinct accessible solutions is trivially constrained by the dimensionality of the solution space. Under these conditions, any increase in fitness neutrality – i.e. *mutational robustness* – will reduce PV. While more explorative/disruptive variation operators can increase PV, nature almost always takes a different approach. In gene regulatory networks and other biological systems, mutational robustness often creates a neutral network that improves access to PV over long periods of time, e.g. by drifting over neutral regions in a fitness landscape [4]. With PV being a prerequisite of evolutionary adaptability, a strong case has been made that this positive correlation of mutational robustness and PV is important to the evolvability of biological systems [4,16,14].

Inspired by these developments, some computational intelligence studies have investigated whether increasing neutrality (e.g. designing a many-to-one mapping between genotypes and phenotypes) influences the evolvability of a search process [1,18,10,7,6]. A common approach is to introduce genetic redundancy so that more than one copy of a gene performs the same function [1,18]. Although some researchers have indicated that redundant forms of neutrality improve evolvability, others have questioned the utility of fitness landscape neutrality generated through redundant encodings [7,15,16].

In the next section we describe, in detail, the computational study used to evaluate Edelman and Gally's hypothesis, including the details for the experimental setup. The proposed model provides the basis for simulating the evolution of a population of multi-agent systems (MAS) and depends on a minimal set of parameters that provide sufficient degrees of freedom to study the system properties – redundancy, degeneracy, robustness and evolvability – that we are interested in. The model (including the fitness function) is formally the same as the one developed in [15]. The study in [15] investigated degeneracy's relationship to genetic neutrality and evolvability and found that degenerate forms of genetic neutrality increase PV while neutrality from redundancy does not. In [16] we expanded on these results and found evidence that neither mutational robustness nor the size of the neutral network in a fitness landscape guarantees high PV, unless degenerate neutrality is present.

The studies in [15,16] investigated PV only within the local vicinity of a static neutral network. While this allowed for comparisons with recent biologically-inspired models (e.g. [4]), it was not within their scope to assign a selective relevance to heritable phenotypic variations. Thus, while previous studies were promising for Edelman and Gally's hypothesis, there has yet to be direct evidence that PV facilitated by degeneracy leads to higher rates of adaptive improvement. In the following we outline a set of experimental conditions that allow us, for the first time, to evaluate Edelman and Gally's claim that degeneracy facilitates evolvability (and not just PV).

3 Computational Study and Experimental Setup

Each MAS $\mathcal{M} = (\boldsymbol{a_1}, \ldots, \boldsymbol{a_n})$ consists of $n = 30$ agents and each agent is able to perform two types of tasks $\boldsymbol{a_i} = (a_{i1}, a_{i2})$ where $0 < a_{i1} < a_{i2} \leq m$. We have chosen a value of $m = 20$. This simple model is sufficient to allow for measurable degrees of redundancy and degeneracy: Any two agents $\boldsymbol{a_i}$ and $\boldsymbol{a_j}$, $i \neq j$, are considered unique with respect to one another if $\forall a_{ik} \in \boldsymbol{a_i} \Rightarrow a_{ik} \notin \boldsymbol{a_j}$. Redundancy with respect to two agents, on the other hand, is defined as $\forall a_{ik} \in \boldsymbol{a_i} \Rightarrow a_{ik} \in \boldsymbol{a_j}$. If a pair of agents is neither unique nor redundant, it is considered degenerate. A system-wide measure of degeneracy (redundancy) of $\mathcal{M}$ then corresponds to the fraction of all unique pair-wise comparisons of all agent pairings that are degenerate (redundant).

Each agent may devote its resources (e.g., time or energy) to the two tasks it is able to carry out. For instance, if agent $\boldsymbol{a_i}$ is able to carry out tasks 1 and 2, it could devote 30% of its resources to task 1 and 70% to task 2. We subsequently define a global resource allocation vector $\mathcal{R} = (\boldsymbol{r_1}, \ldots, \boldsymbol{r_n})$, where each resource allocation $\boldsymbol{r_i}$ is a pair (r_{i1}, r_{i2}) with $0 \leq r_{ij} \leq 1$ and $r_{i1} + r_{i2} = 1$; the number r_{ij} denotes the fraction of resources that agent $\boldsymbol{a_i}$ devotes to its task a_{ij}.

The available resources may be allocated dynamically using a *local decision-making process* without global control. In order to do so efficiently, we discretise the continuous range of each element r_{ij} into 11 segments $\{0, \frac{1}{10}, \ldots, 1\}$. For each iteration of this procedure, we consider every element $\boldsymbol{r_i}$ (without replacement) and perform a local search that systematically increases or decreases the value r_{i1} by $\frac{1}{10}$, doing the opposite for r_{i2} (such that $r_{i1}+r_{i2} = 1$). We do this as long as the fitness of the MAS (see below) improves (or the bounds of r_{ij} have been reached). This step is repeated until no further improvements may be made across all elements of $\mathcal{R}$.

Each MAS is exposed to $s = 10$ distinct scenarios at any one time: each scenario $\boldsymbol{s_i}$ specifies a set of demands for each of the m task types, $\boldsymbol{s_i} = (s_{i1}, \ldots, s_{im})$ where $0 \leq s_{ij} \leq n$. We also impose that the sum of all demands equals the size of the MAS: $\sum_{j=1}^{n} s_{ij} = n$. In order to generate the s scenarios, a *seed scenario* $\boldsymbol{s_0}$ is generated randomly and the remaining $s - 1$ scenarios are then generated by means of a random walk of length 10 (volatility) that always starts from $\boldsymbol{s_0}$. For each step of the random walk, a pair of task-types is chosen uniformly at random and the demand for one of the chosen task-types is increased by a value of 1, the other is decreased by a value of 1 (subject to staying within bounds; if this operation should be unsuccessful, a new pairing of task-types would be chosen). It follows that the total demand of the scenario remains constant but its distribution changes. The set of environments changes every 200 generations (of the genetic algorithm; see below) either *moderately* or *drastically*. For moderate changes, the seed for the new set of scenarios is randomly selected from the previous set (excluding the original $\boldsymbol{s_0}$). For drastic changes, on the other hand, a new seed scenario is generated uniformly at random. The remaining scenarios are generated as before.

The distribution of resources within a MAS (as described above) occurs as a direct response to the environmental conditions (i.e., demands) experienced by the system. We denote the output of a MAS by the vector $\mathcal{O} = (o_1, \ldots, o_n)$ where o_i is the sum of resources dedicated to task-type i: $o_i = \sum_{j=1}^{n} \sum_{k=1}^{2} r_{jk} \cdot [a_{jk} = i]$ where $[\cdot]$ returns 1 if the containing statement is true. The *fitness* of a MAS under environment $\boldsymbol{s}_i$ is then the difference between its output $\mathcal{O}$ and the demand imposed by the environment $\boldsymbol{s_i}$: $F(\mathcal{M}, \boldsymbol{s_i}) = \sum_{j=1}^{m} \max\{0, s_{ij} - o_j\}^2$ where $o_j \in \mathcal{O}$ approximates an optimal allocation of resources under $\boldsymbol{s_i}$ given the capabilities of $\mathcal{M}$. The *robustness* of the MAS is subsequently defined as the average fitness across all scenarios, $R(\mathcal{M}, \{\boldsymbol{s_1}, \ldots, \boldsymbol{s_s}\}) = \frac{1}{s} \sum_{j=1}^{s} F(\mathcal{M}, \boldsymbol{s_j})$. This measure was chosen for simplicity, although we found that robustness measurements that incorporated fitness thresholds did not appear to alter our basic findings.

The vector $\mathcal{O}$ is obtained on-the-fly with respect to each $\boldsymbol{s_i}$ encountered. However, the optimality of the resource allocation is strictly dependent on the task-types contained within the MAS. We thus use a genetic algorithm (GA) based on deterministic crowding to evolve a population of MAS (i.e., $\mathcal{M}$) towards a specific set of scenarios. Prior to the algorithm's execution, $\frac{m}{2}$ unique agent-types (i.e., pairing of task-types) are constructed from the $m = 20$ task-types and stored in a set T. The initial population P, of size $N = 20$, is then created by sampling (with replacement) from T to obtain a MAS that consist exclusively of pairwise unique or redundant agent-types.

During evolution, two parents are randomly selected from the population (without replacement) and subjected to uniform crossover (element-wise probability of 0.5) with probability 1. Each resulting offspring has exactly one element (agent-type) mutated and then replaces the genotypically more similar parent if its fitness is at least as good. Mutation changes the functional capabilities of a single agent and thereby determines whether degeneracy may arise during evolution. The mutation operator has been designed with the following considerations in mind: (a) the search space is to be of the same size in all experiments; (b) in some experiments both redundancy and degeneracy can be selected for during the evolutionary process.

Each position in $\mathcal{M}$ is occupied by a specific agent-type and the mutation operator replaces exactly one such agent-type with a new one. The agent-types available at each position are determined *a priori* and remain constant throughout the execution of the algorithm. In the fully restricted case (no degeneracy), the options at *each* position are given by the set T (which was also used to initialise the population). It follows that a purely redundant MAS remains redundant after mutation. For experiments in which the MAS can evolve degenerate architectures, each position i has a *unique* set of options T'_i which closely resembles the set T but allows for a partial overlap in functions: Each T'_i contains the same task-types as T but half its members (chosen randomly) have exactly one element per task-type pairing altered randomly. The mutation operator is illustrated in Fig. 1b: agents from both system classes have access to the same number of task type pairings (mutation options are shown as faded task type pairings), hence the search space sizes are identical. In the redundant case, mutation options are defined in order to prevent degeneracy. In the degenerate case, it is evident that the agents' capabilities may be unique, redundant, or may partially overlap due to slightly altered task type pairings for each agent.

4 Experimental Results

In our experiments, a MAS architecture (i.e. the specification of agent task capabilities) evolves to maximise robustness within a set of environmental scenarios. To evaluate Edelman and Gally's hypothesis, we place different restrictions on the architectural properties that can evolve in a MAS (see mutation operator in Section 3), preventing degeneracy from arising in some cases. We then evaluate if degeneracy improves adaptation properties during static and dynamic environmental conditions.

4.1 Robustness, Evolvability in Static (Heterogeneous) Environments

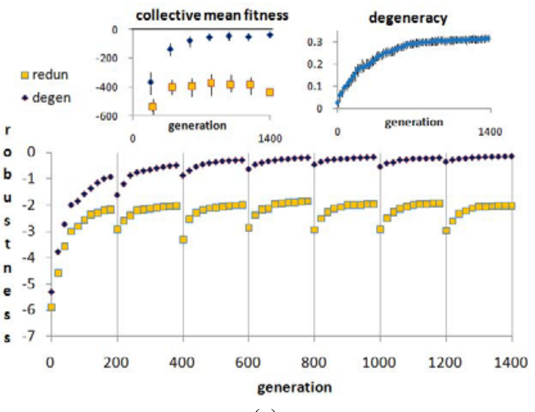
In Fig. 1a, for the 200 generations before the environment changes we see that, when degeneracy is allowed to emerge, the MAS evolves higher robustness towards the set of environmental scenarios. This finding is not intuitively expected considering that: systems are the same size (and solution spaces are constrained to identical sizes), MAS are presented with the same scenarios, agents have access to the same task types and, within a noiseless environment, all MAS evolve within a unimodal fitness landscape that contains the same optimal fitness value. In our view, there are two factors that primarily contribute to these observed differences in evolved robustness: 1) evolvability within the static noisy environment (discussed below) and 2) differences in the robustness potential of a system (discussed in the networked buffering hypothesis in [17]).

Conceptually, evolvability is about discovering heritable phenotypic improvements. In Fig. 1d, we record the probability distribution for the time it takes the MAS population to find a better solution. As can be seen, degenerate architectures are finding adaptive improvements more quickly. An analysis of improvement size vs fitness finds this relationship is similar for the two types of MAS, thus suggesting the faster adaptation rate is largely responsible for the divergence in fitness from generation 0 to 200.

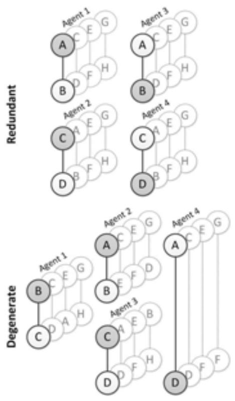
4.2 Evolvability in Dynamic Environments

In a dynamic environment, evolvability is no longer merely about a propensity for discovering improvements but is also about sustaining high fitness throughout and after the environment changes. As can be seen from Fig. 1a and c, in both redundant and degenerate MAS, robustness drops every 200 generations when environmental change is imposed. This drop reflects declines in fitness across the population. However, we can make the following noteworthy observations.

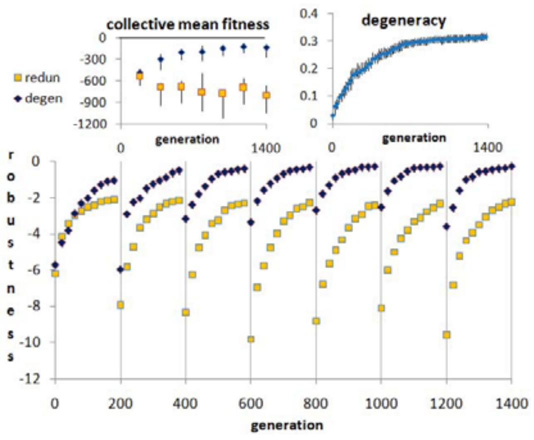
When evolution of degeneracy is enabled, MAS populations can adapt better to change than MAS with purely redundant architectures. Except for the decline of fitness at generation 200, all subsequent drops (at generations 400, 600, etc.) are smaller in the 'degen' experiments than in the 'redun' experiments, irrespective of whether the scenario changes are moderate or drastic. With every change in the set of scenarios, MAS that cannot evolve degenerate architectures appear to drop in performance by similar amounts. The only exception is the first adaptation after a change of environmental conditions (i.e. the time period from generation 201 to 400) where there is some overall improvement when the environmental change is moderate (Fig. 1a), and some overall deterioration when the change is drastic (Fig. 1c). MAS that can evolve degeneracy, on the other hand, have some capacity to adapt to the nature of change. From



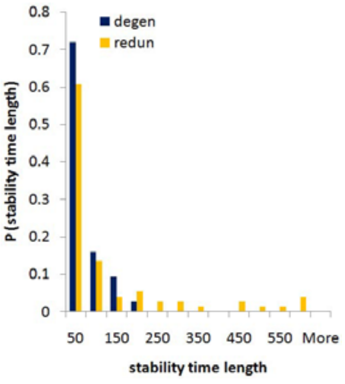
(a)



(b)



(c)



(d)

Fig. 1. Figure 1 (a): When degeneracy is/is not permitted in the MAS architecture, we label these as ‘degen’/‘redun’. The main graph plots robustness evolved over time with smaller graphs of collective mean fitness (top-left) and degeneracy (for the MAS where it is allowed to emerge, top-right). Environmental changes (every 200 gen.) are moderate (see Experimental Setup). (b): Degenerate and redundant MAS. Agents (depicted as pairs of connected nodes) can perform 2 different task types. Each MAS (top: redundant; bottom: degenerate) consists of 4 agents and the faded pairings indicate the predetermined set of options the mutation operator may choose from. (c): MAS evolve in conditions where environmental changes are dramatic. (d): histogram for the number of offspring sampled before an improvement is found (stability time length). Conditions are the same as (a) except environmental changes occur every 400 generations.

environmental change to environmental change, the drop in fitness/robustness becomes smaller.

When plotting the *collective mean fitness* (i.e. the area under the fitness/robustness curve between two consecutive environmental changes [8]), we do not only observe this adaptation in experiments with moderately changing environments (top-left graph in Fig. 1a) but we also see overall adaptation levels improve over time even when the environmental changes are drastically different (top-left graph in Fig 1c). Comparing this with the amount of degeneracy integrated within the MAS architecture (top-right graphs in Fig. 1a and c), we see that the collective mean fitness improves as degeneracy is integrated into the system. Furthermore, the degeneracy-enabled capacity to adapt is better when changes in the environment are moderate or correlated; a proposed precondition for continuous adaptation in DOP (see [2]). It is admittedly difficult however to directly evaluate changes in the rate of adaptation (e.g. as we did for a static environment in Fig. 1d) in the dynamic case because fitness differences at the beginning of each epoch act to confound such an analysis. We note however that in somewhat similar MAS models, experimental conditions were established that can more clearly demonstrate an acceleration in adaptation rates during degenerate MAS evolution within a dynamic environment [12].

When we make the scale of our model larger (i.e. by increasing MAS size, T, and random walk size by the same proportion), the differences between degenerate and redundant MAS in robustness, evolvability and collective mean fitness become accentuated. Future studies guided by selected MAS application domains will aim to further investigate the generality and limitations of these findings by considering: restrictions in functional capability combinations in each agent, different classes of environment correlation, the speed of agent behavior modification, costs in agent behavior modification, and agent-agent functional interdependencies.

5 Discussion and Conclusions

In this paper, we investigated the potential for designing dynamic optimisation problem (DOP) representations that are robust to environmental conditions experienced during a solution's lifecycle and, at the same time, have the capacity to adapt to changing environments. Our investigation was motivated by a hypothesis formulated in the context of biological evolution – namely that degeneracy facilitates robustness and adaptation in time-variant environments. In simulation experiments we evolved populations of multi-agent systems (MAS) and compared the robustness and adaptation potentials of systems that could evolve degenerate architectures with those that could evolve redundant structures only. We found evidence that incorporating degeneracy into a problem's representation can improve robustness and adaptiveness of dynamic optimisation in ways that are not seen in purely redundant problem representations.

While our investigation was quite abstract, we can identify several features that make degeneracy suitable for dynamic optimisation. First, degenerate systems appear to exhibit a greater propensity to adapt. While we have not reported an analysis of fitness landscape neutrality here, previous studies on the ensemble properties of similar models have shown degeneracy creates neutral regions in fitness landscapes with high access to phenotypic variety [15,16]. In light of these earlier studies, the results presented here demonstrate that the discovery of adaptations in static neutral landscapes created by degeneracy can be surprisingly rapid. Theoretical arguments have suggested that long periods of time may be needed to discover a single adaptive phenotype from a neutral network [11], however the rapid adaptation in Fig. 1a,d suggests that little neutrality is ever traversed in these experiments before an improvement is discovered. As believed to also take place in biology, this fast pace of adaptation likely reflects the existence of many alternative paths to adaptive change within neutral networks created by degeneracy. This means that little of the neutral network needs to be searched before new improvements are found, thus fitness barriers are not being replaced with large "entropic barriers" during evolution, cf [11]. While optimal solutions are not guaranteed, the propensity to adapt in evolved degenerate systems appears to allow such a strategy to quickly find highly fit and highly robust solutions – as needed when tackling DOP.

A second desirable feature of degenerate systems is their enhanced capacity to deal with novel conditions. Compared with redundant architectures, degenerate systems have a greater potential to evolve innovative solution responses that account for small variations in environmental conditions. In a supplemental analysis of these systems we have found this robustness potential can extend to moderate degrees of environmental novelty, thus helping to explain the differences between system classes immediately after a change in the environment (Fig. 1a,c). However, a further reason that degenerate MAS exhibited highly effective responses to immediate environmental change was the emergence of population properties known in evolutionary biology as cryptic genetic variation (CGV).

Many EA-based dynamic optimisation techniques aim to artificially control population convergence based on a general understanding that low genetic diversity limits a population's adaptability when it encounters a changed fitness landscape. The resulting genetic and phenotypic properties of EA populations differ significantly however from that observed in natural populations. Genetic diversity within natural populations is maintained in a static environment by being phenotypically and selectively hidden. Trait differences across the population are mostly exposed only after an environment changes; a phenomena known as *cryptic genetic variation* (CGV). The present study focuses on how Edelman and Gally's hypothesis is relevant when applying neutral evolution theories to the topic of evolvable problem representations. However in [13] we also analyze the population properties from these experiments and report evidence that degeneracy generates hide and release mechanisms for genetic diversity that are analogous to the natural CGV phenomena just described. This evidence of CGV is presented as a separate supplemental report in [13] due to space limitations as well as its distinctive theoretical relevance.

Acknowledgements

This work was partially supported by a DSTO grant on "Fleet Designs for Robustness and Adaptiveness" and an EPSRC grant (No. EP/E058884/1) on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications."

References

1. Banzhaf, W.: Genotype-phenotype-mapping and neutral variation-a case study in genetic programming. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 322–332. Springer, Heidelberg (1994)

2. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Congress on Evolutionary Computation, vol. 3, pp. 1875–1882. IEEE Computer Society, Washington (1999)
3. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer, Dordrecht (2002)
4. Ciliberti, S., Martin, O.C., Wagner, A.: Innovation and robustness in complex regulatory gene networks. Proceedings of the National Academy of Sciences, USA 104(34), 13591–13596 (2007)
5. Edelman, G.M., Gally, J.A.: Degeneracy and complexity in biological systems. Proceedings of the National Academy of Sciences, USA 98(24), 13763–13768 (2001)
6. Jin, Y., Gruna, R., Paenke, I., Sendhoff, B.: Evolutionary multi-objective optimization of robustness and innovation in redundant genetic representations. In: IEEE Symposium on Computational Intelligence in Multi-Criteria Decision Making, Nashville, USA, pp. 38–42 (2009)
7. Knowles, J.D., Watson, R.A.: On the utility of redundant encodings in mutation-based evolutionary search. LNCS, pp. 88–98. Springer, Heidelberg (2003)
8. Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Springer, Berlin (2004)
9. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Springer, Heidelberg (2006)
10. Rothlauf, F., Goldberg, D.E.: Redundant representations in evolutionary computation. Evolutionary Computation 11(4), 381–415 (2003)
11. van Nimwegen, E., Crutchfield, J.P.: Metastable evolutionary dynamics: Crossing fitness barriers or escaping via neutral paths? Bulletin of Mathematical Biology 62(5), 799–848 (2000)
12. Whitacre, J.M.: Evolution-inspired approaches for engineering emergent robustness in an uncertain dynamic world. In: Proceedings of the Conference on Artificial Life XII (to appear)
13. Whitacre, J.M.: Genetic and environment-induced innovation: complementary pathways to adaptive change that are facilitated by degeneracy in multi-agent systems. In: Proceedings of the Conference on Artificial Life XII (to appear)
14. Whitacre, J.M.: Degeneracy: a link between evolvability, robustness and complexity in biological systems. Theoretical Biology and Medical Modelling 7(1), 6 (2010)
15. Whitacre, J.M., Bender, A.: Degenerate neutrality creates evolvable fitness landscapes. In: WorldComp 2009, July 13-16 (2009)
16. Whitacre, J.M., Bender, A.: Degeneracy: a design principle for achieving robustness and evolvability. Journal of Theoretical Biology 263(1), 143–153 (2010)
17. Whitacre, J.M., Bender, A.: Networked buffering: a basic mechanism for distributed robustness in complex adaptive systems. Theoretical Biology and Medical Modelling 7(20) (to appear 2010)
18. Yu, T., Miller, J.F.: Neutrality and the evolvability of boolean function landscape. In: Proceedings of the 4th European Conference on Genetic Programming, pp. 204–217. Springer, London (2001)

Evolutionary Learning of Technical Trading Rules without Data-Mining Bias

Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon

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

Abstract. In this paper we investigate the profitability of evolved technical trading rules when controlling for data-mining bias. For the first time in the evolutionary computation literature, a comprehensive test for a rule's statistical significance using Hansen's Superior Predictive Ability is explicitly taken into account in the fitness function, and multi-objective evolutionary optimisation is employed to drive the search towards individual rules with better generalisation abilities. Empirical results on a spot foreign-exchange market index suggest that increased out-of-sample performance can be obtained after accounting for data-mining bias effects in a multi-objective fitness function, as compared to a single-criterion fitness measure that considers solely the average return.

1 Introduction

The goal of objective technical analysis is the discovery of rules that will be profitable in the future. The research method is back-testing, which produces an observable measure of performance. On the basis of this test statistic an inference is made about a population parameter, the rule's expected performance out of sample. In evolutionary rule data-mining, many rules are back-tested in each generation, and the selection of rules that form the basis of the subsequent rules to-be-sampled is stochastically based on their observed performance. This refinement cycle eventually results in a rule with the best performance being designated as the output of the run. That is to say, this form of data mining involves a performance competition that leads to a winning rule being picked. The problem is that the winning rule's observed performance that allowed it to be picked over all other rules systematically overstates how well the rule is likely to perform in the future. This systematic error is the *data-mining bias*.

Out-of-sample rule performance deterioration is a well-known problem [1,2]. A dominant explanation of the out-of-sample performance break-down is data-mining bias. This has two constituents: (1) randomness, which is a relatively large component of observed performance. It is reasoned that a portion of a rule's back-tested performance was merely luck - a coincidental correspondence between the rule's signals and the market's non-recurring noise. Because this random component is a non-recurring phenomenon that will manifest differently in each sample of data, the rule's expected performance always falls below observed performance on the training data. (2) the logic of data mining, in which the best-performing rule is selected after a repertoire of candidate rules have been evaluated (the observed average return of the best-performing rule is a positively-biased statistic [1]).

Previous research on the induction of data-driven models by means of evolutionary computation accounted for the problem of data-mining bias using out-of-sample testing [3,4], methods to restrict the model complexity [5], and ensemble learning [6]. The most prevalent of these, *out-of-sample* testing, is based on the valid notion that the performance of a data-mined rule, on out-of-sample data, provides an unbiased estimate of the rule's expected performance. However, out-of-sample testing suffers from several deficiencies. First and foremost, the unused status of the data reserved for out-of-sample testing has a short life-span. Once it has been used, it is no longer able to provide unbiased estimates of rule performance. Secondly, it eliminates certain portions of the data from mining operations, thus, it reduces the amount of data available to detect patterns. When noise is high and information is low, the bigger the amount of training data the better the chance of mining something useful. Third, the decision about how to apportion the data between in- and out-of-sample subsets is arbitrary, and hence lacks desired objectivity.

In this paper we propose a new approach to account for the data-mining bias inherent in an iterative modelling technique, such as grammar-based Genetic Programming (GP) [7], where a single dataset is used more than once in the model construction process. Our approach is based on multi-objective evolutionary learning of technical trading rules, where the optimisation criterion is a weighted amalgamation of the rule's observed daily return and the statistical significance (p-value) of this test statistic. This permits data-mining bias to be undertaken with some degree of confidence, and incorporated in the objective function that drives the evolutionary search. Our expectation is that such a fitness measure will create an evolutionary pressure towards parts of the search space that contain individuals with true predictive abilities, and thus lead to better generalisation to unseen data. The rest of the paper is organised as follows. First we introduce two dominant methods for testing the null hypothesis that the best rule encountered during data mining has no predictive superiority over a benchmark model devoid of predictive power. The grammar-based GP system is then described, giving details of the grammar employed, the technical indicators serving as the building blocks for constructing trading rules, the evolutionary algorithm, the trading methodology, and the multi-objective fitness function. The description of the experimental approach comes next, followed by an analysis of the experimental results. Finally, a concluding section summarises our findings, and sketches future directions of this research.

2 Data-Mining-Adjusted Statistical Hypothesis Tests

In the context of evaluating the data mining of technical analysis rules, it is conceivable that by repeatedly examining different trading rules against the same dataset, some rules would appear to be profitable simply due to chance (data mining/data snooping bias). White's (2000) Reality Check (WRC) test [8] and Hansen's (2005) Superior Predictive Ability (SPA) test [9] provide comprehensive tests across all trading rules considered, and directly quantify the effect of data-mining bias by testing the null hypothesis that the performance of the best trading rule is no better than the performance of the benchmark set to a trading model totally devoid of predictive power according to a performance statistic (i.e. observed average return in a back-test). The best rule is identified by applying the performance measure to the full universe of trading rules, and a desired p-value is obtained by comparing the best rule's sample statistic to approximations of the sampling distribution of the test statistic. In both methods bootstrapping is used to approximate the sampling distribution of the test statistic.

Given M models (trading rules), let $\varphi_{k,t}$ ($k = 1, 2, \ldots, M$ and $t = 1, 2, \ldots, N$) denote their performance measures relative to the benchmark model over time t. The null hypothesis is that there does not exist a superior rule in the universe of M rules (joint test).

$$H_0 : \underset{k = 1, \ldots, M}{max} \varphi_k \leq 0. \tag{1}$$

Rejecting H_0 implies that there exist at least one rule that outperforms the benchmark. Setting the performance measure to the rule's mean return obtained by back-testing it in a historical sample of data, the benchmark becomes a rule that has an expected return of zero or less, thus, $\varphi_k = E(f_k)$, where f_k is the return of the k-th trading rule. It is then natural to base the test statistic of hypothesis test to the maximum of the normalised average of $f_{k,t}$:

$$\overline{V}_n = \underset{k = 1, \ldots, M}{max} \sqrt{n} \bar{f}_k \tag{2}$$

where $\bar{f}_k = \frac{1}{n} \sum_{t=1}^{n} f_{k.t}$, with $f_{k,t}$ the t-th observation of f_k.

White suggested using the stationary bootstrap method [10] to approximate the p-values of $\overline{V}_n$. In general, a bootstrap method derives the sampling distribution of a test statistic by resampling with replacement from an original sample. The reason that White decided to use a block-bootstrap method is to maintain some of the statistical properties of the bootstrapped time-series such as heteroskedasticity [8]. To describe the bootstrap algorithm, let X_n be a strictly stationary time-series. Suppose μ is a parameter of the whole joint distribution of the sequence (i.e mean). Given data $X_1, \ldots, X_N$ the goal is to make inferences about μ. Suppose $B_{i,b} = \{X_i, X_{i+1}, \ldots X_{i+b-1}\}$ be a block of b observations starting from X_i. In the case where $j > N$, X_j is defined to be X_i, where $i = (j \texttt{ mod } N)$ and $X_0 = X_N$. Let p be a fixed number in $\{0, \ldots, 1\}$. independent of $X_1, \ldots, X_N$, let $L_1, L_2, \ldots$ be a sequence of independent and identically distributed random variables having the geometric distribution, so that the probability of the event $L_i = m$ is $(1-p)^{(m-1)}p$ for $m = 1, 2, \ldots$. independent of X_i and the L_i, let $I_1, I_2, \ldots$ be a sequence of independent and identically distributed variables which have the discrete uniform distribution on $\{1, \ldots, N\}$. A pseudo-time-series $X_1^*, \ldots, X_N^*$ is generated in the following way: Sample a sequence of blocks of random length by the prescription $B_{I_1,L_1}, B_{I_2,L_2}, \ldots$. The first L_1 observations in the pseudo-time-series $X_1^*, \ldots, X_N^*$ are determined by the first block B_{I_1,L_1} of observations $X_{I_1}, \ldots, X_{I_1+L_1-1}$, the next L_2 observations are the observations in the second sampled block B_{I_2,L_2} , namely $X_{I_2}, \ldots, X_{I_2+L_2-1}$. This process is stopped once N observations in the pseudo-time-series have been generated.

Now, back to White's Reality Check, let $f_k^*(\mathrm{b})$ denote the b-th bootstrapped sample of f_k and $\bar{f}_k^*(b) = \frac{1}{n}\sum_{t=1}^{n} f_{k.t}^*(b)$ its sample average. A bootstrapped sampling distribution $\overline{V}_n^*$ is obtained with the realisations:

$$\overline{V}_n^*(b) = \begin{matrix} max \\ k = 1, \ldots, M \end{matrix} \sqrt{n}(\bar{f}_k^*(b) - \bar{f}_k), b = 1, \ldots, B. \tag{3}$$

The WRC p-value is obtained by comparing V_n the quantiles of the sampling distribution of $\overline{V}_n^*$. The null hypothesis is rejected whenever p-value is less than a given significance level.

Hansen pointed out two potential inefficiencies with White's Reality Check. First, the average returns $\bar{f}_k$ are not standardised. Second, despite that H_0 is composite, the sampling distribution of WRC is based on the "least favourable configuration" (the configuration that is least favourable to the alternative), that is all of the back-tested rules have expected returns of zero. Therefore, the WRD test may lose power dramatically when poor rules with very negative $E(f_k)$ are included in the test. The proposed SPA test is based on studentised returns:

$$\overline{V}_n = max\left(\begin{matrix} max \\ k = 1, \ldots, M \end{matrix} \frac{\sqrt{n}\bar{f}_k}{\sigma_k}, 0\right) \tag{4}$$

where σ_k is a consistent estimator of the standard deviation of $\sqrt{n}\bar{f}_k$.

To avoid using the least favourable configuration and increase the power of the test, Hansen suggested a different way to bootstrap the distribution of $\overline{V}_n$. For the k-th rule, let $\bar{Z}_n^*(b)$ denote the sample average of the b-th bootstrapped sample of the centered returns:

$$Z_{k,t}^*(b) = f_{k,t}^*(b) - \bar{f}_k \mathbf{1}_{\{\bar{f}_k \geq -A_k\}} \tag{5}$$

where $\mathbf{1}(\mathrm{G})$ denotes the indicator function of the event G, and $A_k = -\frac{\sigma_k}{4n^{1/4}}$. The p-value is obtained by the bootstrapped sampling distribution whose realisations are:

$$\overline{V}_n^*(b) = max\left(\begin{matrix} max \\ k = 1, \ldots, M \end{matrix} \frac{\sqrt{n}\bar{Z}_k^*(b)}{\sigma_k}, 0\right), b = 1, \ldots, B. \tag{6}$$

3 Grammar-Based Genetic Programming for Rule Induction

We employ a grammar-based GP system to evolve technical trading rules. The method used is outlined in the following sections.

3.1 Grammar

A context-free grammar is employed to type the language used for program representation. It is presented below:

```
<prog> ::= <if>
<if> ::= <predicate> <expr> <expr>
<expr> ::= <if> | <signal>
<signal> ::= golong | goshort
<predicate> ::= <ti> <op> <constant> | <ti> <op> <ti>
<op> ::= < | >
<ti> ::= MACD | RSI | SM | ADX
<constant> ::= -0.5 | -0.49 | ... | 0.49 | 0.5 | 1.0 | 2.0 ... | 100.0
```

Using the grammar above, a technical trading rule is represented as a disjunction of conjunctions of constraints on the values of technical indicators, taking the classical form of *oblique decision tree* learning for approximating discrete-valued target functions, however, here we also allow comparisons between technical indicators. The space of technical trading rules is formed using the following indicators: **Relative Strength Index** (RSI), **Moving Average Convergence Divergence** (MACD), **Stochastics Momentum** (SM), **Average Directional Movement Index** (ADX) [11]. MACD will usually oscillate around zero with unknown upper and lower bounds, whereas RSI, SM, and ADX oscillate in the range of $\{0, \ldots, 100\}$. The constants that form part of the predicates that test a real-valued technical indicator against some value come from the union of two sets, $\{-0.5, -0.49, \ldots, 0.49, 0.5\}$ and $\{1.0, \ldots 100.0\}$. The reason of choice of this representation is to enhance human understandability of the conditions that trigger certain trading signals, and treat the outcome of the evolutionary process as a decision-support system rather than merely as a black-box method for trading.

3.2 Trading Methodology

Each evolved rule outputs two values, 1 and -1, interpreted a long and short position respectively. The average return of a rule is generated as follows. Let r_t be the daily return of the index at time t, calculated using $(v_t - v_{t-1})/v_{t-1}$, where v_t and v_{t-1} are the values of the time-series at time t and $t-1$ respectively. Also, let s_{t-1} be the trading signal generated by the rule at time $t-1$. Then $d_t = s_{t-1} r_t$ is the realised return at time t. Using a back-test period, an average of d_t can be induced. We are not considering trading, slippage or interest costs.

3.3 Evolutionary Algorithm

For our evolutionary algorithm, we used a panmictic, generational, elitist genetic algorithm. The algorithm uses tournament selection with a tournament size of 7. Evolution proceeds for 50 generations, and the population size is set to $1,000$ individuals. Ramped-half-and-half tree creation with a maximum depth of 5 is used to perform a random sampling of rules during run initialisation. Throughout evolution, expression-trees are allowed to grow up to depth of 10. The evolutionary search employs a mutation-based variation scheme, where subtree mutation is combined with point-mutation; a probability governing the application of each, set to 0.6 in favour of subtree mutation. Neither recombination, nor reproduction were used.

3.4 Fitness Function

In the case of a single-objective fitness function, this takes the form of average daily return generated by the rule's trading signals over a back-test period specified by the training set. On the other hand, the multi-objective fitness function is defined as a weighted sum of average daily return and the p-value of this statistic that is generated from SPA test. For the weighting scheme to be effective the two objectives need to be similarly scaled. In the case of p-value, this is naturally defined within the $\{0, \ldots, 1\}$ interval. We employed a simple normalisation technique to make the average daily return of each individual in a population fall into the same $\{0, \ldots, 1\}$ interval, by taking into account the minimum and maximum average daily returns produced by individuals in each population.

3.5 Adapting the Data-Mining Bias Tests to a Population of Rules

SPA tests a composite null hypothesis whose test statistic is defined as the *maximum standardised mean of N means*, where N is the number of rules in our universe. It is therefore analogous to treat each evolving population as a universe of N individual rules. Under this formulation the following sequence of steps are involved in calculating the p-values:

1. Calculate the standardised mean daily return for each rule in a population on de-trended daily returns of a back-test period (note - this is not to be confused with de-trending of the time-series of an index). De-trended daily returns have an average daily return of zero, thus the expected return of a rule with no predictive power will be zero if its returns are computed from de-trended data. De-trending is a simple operation, where the average daily return over a period is subtracted from each daily return. Sort the population in descending order based on standardised mean daily return.
2. Using the the bootstrap method described in Section 2 create a bootstrapped sample of trading dates (days).
3. Using the dates obtained in the bootstrapped sample, a pseudo-track record based on the actual daily returns associated with these dates is created for each rule in the population.

4. For every rule in the population, each daily return in the pseudo-track record is adjusted according to Equation 5 (Section 2). The adjusted pseudo-track record of returns is then averaged and standardised.
5. The larger of these standardised values and zero is designated as the first value to form the sampling distribution of the test statistic of Equation 4 (Section 2).
6. Steps 2 to 5 are repeated many times (i.e B times). In this way, the sampling distribution of the test statistic is approximated from these B values.
7. The p-value of the first rule (in the sorted list of rules of step 1) is calculated as a fraction of B values that exceed the standardised mean daily return of the tested rule.
8. Remove the tested rule from the sorted list, and repeat steps 2 to 7 using the remaining rules.

4 Experimental Approach

This empirical study aims to reveal whether there is an advantage accruing from using the statistical significance of average daily returns acquired in a back-test as an additional objective, in order to drive the evolutionary search towards better-generalising technical trading rules (out-of-sample testing). For this, we are considering an exhaustive set of combinations (with a step of 0.1) for coefficients that weight the average daily return and the p-value in the multi-objective fitness function, in order to manually set the trade-off between the two. An obvious benchmark to compare against our methodology is the single-objective evolution of technical rules, using solely the average daily return as the fitness function.

This study uses daily closing foreign exchange rate of USD/GBP for the period of 01/01/1990 to 31/03/2010. The first $4,000$ trading days are used as the training-set, whereas the remaining $1,229$ as the test-set. The parameters of the technical indicators are set as follows: $n = 21$ for RSI; $n = 28$, $r = 30$, $s = 2$ for SM; $n = 14$ for ADX. For the stationary bootstrap procedure we set the number of bootstrapped samples to $2,000$, and the probability of success in each trial to 0.9 in order to generate the probability mass function of the geometric distribution. We perform 50 independent runs for each experimental setup allowing either for a multi-objective fitness function (with variable weighting coefficients), or a single-objective fitness function.

5 Results

A summary of the experimental results is depicted in Table 1. A statistically significant difference (unpaired t-test, $p < 0.05$, degrees of freedom $df = 98$) is found between the average daily return of single-objective fitness function and multi-objective one (weighting coefficients of 0.2, 0.8) during out-of-sample back-testing, suggesting a better generalisation ability of the trading rules that were encouraged to take account of the data-mining bias during the induction process. Results on the out-of-sample data also show that the best evolved technical rule, using the multi-criterion fitness, outperforms its single-objective-evolution counterpart, obtaining an annualised return of 13.75% (average daily return of 0.055) as opposed to 8% (average daily return of 0.032). In addition, it is interesting to note that most combinations of coefficients for weighting the impact of different objectives yielded similar generalisation performance, suggesting that the (0.2, 0.8) interplay in favour of the *p*-value creates the tradeoff required to drive the evolutionary search towards the discovery of better-generalising trading rules. Nevertheless, optimising in favour of the *p*-value evidently leads to inferior in-sample performance; a statistically significance difference (unpaired t-test, $p < 0.0001$, $df = 98$) is found in the average daily return between single- and multi-objective fitness functions (0.2, 0.8) for the training period. This is intuitive, indicative of the closer fit to the training data that is obtained by a model that is evolved unconstrained for sole profitability maximisation.

Table 1. Performance summary. Average daily return has been abbreviated to AR. Means are based on best-of-run individuals from 50 evolutionary runs. The case where the weighting coefficients for AR and *p*-value are set to 1.0 and 0.0 respectively refers to the single-objective fitness function. Std. deviation in parentheses for mean. Best out-of-sample performance indicated in bold.

AR coeff.	p-value coeff.	Mean Train AR	Min Test AR	Mean Test AR	Max Test AR
1.0	0.0	0.037 (0.002)	−0.027	0.007 (0.013)	0.032
0.9	0.1	0.038 (0.003)	−0.014	0.008 (0.014)	0.042
0.8	0.2	0.037 (0.003)	−0.021	0.009 (0.015)	0.041
0.7	0.3	0.038 (0.003)	−0.015	0.006 (0.013)	0.030
0.6	0.4	0.037 (0.002)	−0.012	0.007 (0.014)	0.045
0.5	0.5	0.037 (0.003)	−0.014	0.007 (0.012)	0.026
0.4	0.6	0.037 (0.003)	−0.015	0.007 (0.015)	0.040
0.3	0.7	0.038 (0.003)	−0.022	0.006 (0.014)	0.031
0.2	0.8	0.033 (0.002)	−0.020	**0.014** (0.018)	**0.055**
0.1	0.9	0.038 (0.003)	−0.013	0.007 (0.013)	0.028

The graphs depicted in Figure 1 show the evolution of best-of-generation average daily return for the cases of single- and multi-objective (0.2, 0.8) fitness functions, for both in- and out-of-sample data-sets. Figures 1(a), 1(b) show that single-objective evolution learns faster, and the trading rules fit more closely to the training data, achieving bigger daily returns compared to the multi-objective case. Figure 1(c) illustrates the learning curve for the out-of-sample data, depicting an inherent difficulty in the generalisation ability of the best-of-generation trading rule; a phenomenon that has been widely documented in previous studies. An interesting result is depicted in Figure 1(d), where a good generalisation to unseen data is observed by relatively random rules in the initial generations, and then followed by a rapid decrease in performance up to approx. generation 7, before learning starts again. This learning behaviour is explained by the nature of the time-series in the training and testing data-sets, which allows relatively random rules to better model the time-series fluctuations in the testing-set. However, as learning is dictated on the information provided by the training-set, the

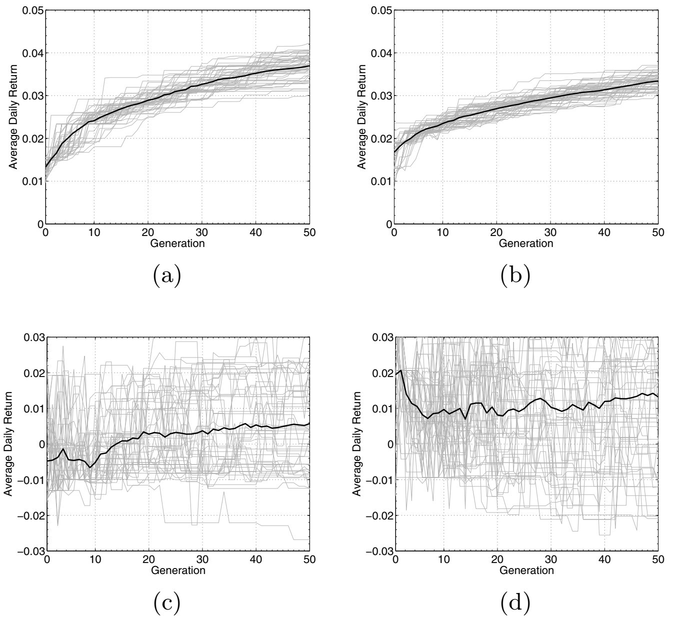


Fig. 1. Evolution of best-of-generation average daily return. (a) and (b) show in-sample evolution using sets of weighting coefficients represented by the tuples (1.0, 0.0) and (0.2, 0.8) respectively; (c) and (d) show the out-of-sample evolution for the same weighting coefficient setups. Each graph presents 50 evolutionary runs; average in bold.

adaptive expression-tree structures are gradually fitting to in-sample data, drifting away from the genotypes that exhibited an initial out-of-sample superiority.

6 Conclusion

Multiple studies on the evolutionary search for profitable technical trading rules have been conducted in the past, suggesting that this form of rule induction technique does have merit. However, the effects of data-mining bias in the generalisation ability of the trading rules have not been accounted for. We proposed a method to encourage the evolution of technical rules with statistically significant returns during a back-testing training period, in an expectation to increase their out-of-sample performance. This relies on a multi-criterion fitness function that in addition to a measure of profitability, takes into account Hansen's Superior Predictive Ability test, which can directly quantify the effect of data-mining bias, by testing the performance of the best mined rule in the context of the full universe of technical trading rules. Initial experiments, using an index from a foreign-exchange market, are encouraging, resulting in human-understandable trading rules with better generalisation to unseen data after accounting for data-mining bias. Future work includes the application of this methodology to a wider range of market indices in order to corroborate its breadth of efficiency, and the employmdent of Pareto-based evolutionary optimisation. The grammar employed in our experiments was deliberately kept as simple as possible in an attempt to minimise exogenous factors affecting performance, in order to objectively assess the potential of the newly introduced method. Subsequent versions will rely on dynamically setting the constraints on technical indicators, as well as a richer repertoire of such primitive constructs.

Acknowledgement

This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grant Number 08/SRC/FM1389.

References

1. Aronson, D.: Evidence-Based Technical Analysis. John Wiley and Sons, Inc., Chichester (2007)
2. Pardo, R.: Design, testing and optimisation of trading systems. John Wiley and Sons, Inc., Chichester (1992)
3. O'Neill, M., Brabazon, A., Ryan, C., Collins, J.J.: Evolving market index trading rules using grammatical evolution. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 343–352. Springer, Heidelberg (2001)
4. Thomas, J.D., Sycara, K.: The importance of simplicity and validation in genetic programming for data mining in financial data. In: Freitas, A.A. (ed.) Data Mining with Evolutionary Algorithms: Research Directions, Orlando, Florida, Technical Report WS-99-06, July 18, pp. 7–11. AAAI Press, Menlo Park (1999)
5. Iba, H., De Garis, H., Sato, T.: Genetic programming using a minimum description length principle. In: Advances in Genetic Programming, pp. 265–284. MIT Press, Cambridge (1994)
6. Breiman, L., Breiman, L.: Bagging predictors. Machine Learning, 123–140 (1996)
7. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in a Arbitrary Language. Genetic programming, vol. 4. Kluwer Academic Publishers, Dordrecht (2003)
8. White, H.: A reality check for data snooping. Econometrica 68(5), 1097–1126 (2000)
9. Hansen, P.R.: A test for superior predictive ability. Journal of Business & Economic Statistics 23, 365–380 (2005)
10. Romano, J., Politis, D.: The stationary bootstrap. Journal of American Statistical Association 89(428), 1303–1313 (1994)
11. Kaufman, P.: New Trading Systems and Methods, 4th edn. Wiley, Chichester (2005)

Using Computational Intelligence to Identify Performance Bottlenecks in a Computer System

Faraz Ahmed, Farrukh Shahzad, and Muddassar Farooq

Next Generation Intelligent Network Research Center (nexGIN RC)
National University of Computer & Emerging Sciences (FAST-NUCES)
Islamabad, Pakistan
{faraz.ahmed,farrukh.shahzad,muddassar.farooq}@nexginrc.org

Abstract. System administrators have to analyze a number of system parameters to identify performance bottlenecks in a system. The major contribution of this paper is a utility – EvoPerf – which has the ability to autonomously monitor different system-wide parameters, requiring no user intervention, to accurately identify performance based anomalies (or bottlenecks). EvoPerf uses Windows `Perfmon` utility to collect a number of performance counters from the kernel of Windows OS. Subsequently, we show that artificial intelligence based techniques – using performance counters – can be used successfully to design an accurate and efficient performance monitoring utility. We evaluate feasibility of six classifiers – UCS, GAssist-ADI, GAssist-Int, NN-MLP, NN-RBF and J48 – and conclude that all classifiers provide more than 99% classification accuracy with less than 1% false positives. However, the processing overhead of J48 and neural networks based classifiers is significantly smaller compared with evolutionary classifiers.

1 Introduction

The pervasive penetration of Internet and associated next generation intelligent networks has resulted in great demand for e-commerce, gaming and e-health applications that must provide ubiquitous and instant access to its potential customers in a reliable and efficient manner. Currently, in most of the cases, system administrators themselves analyze and correlate a number of parameters – CPU usage, memory usage, network utilization etc. – to identify bottlenecks in a computer system [8]. A performance bottleneck can seriously compromise or undermine the functionality of a given business: unavailability of service due to a denial of service attack or crashing of a server process on account of low memory.

System administrators need diagnostic tools that can automatically identify bottlenecks on different server machines; as a result, they can efficiently invoke countermeasure strategies to gradually remove the bottleneck in the system. Therefore, in this paper, we propose a tool that can automatically monitor the

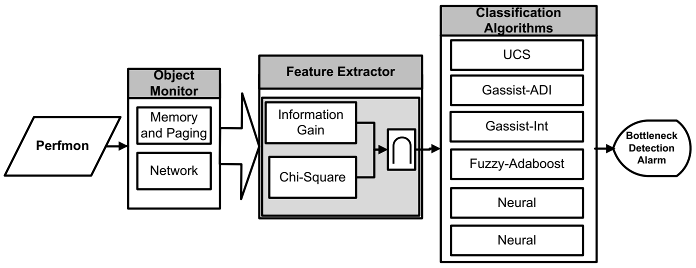


Fig. 1. Architecture of EvoPerf

system-wide performance of a computer and raise an alarm if the system is experiencing a bottleneck.[1] Our monitoring system consists of three sub-modules: (1) object monitor, (2) feature selector, and (3) classifier. An object monitor uses Windows `Perfmon` utility to collect a number of performance counters from the kernel of Windows OS. The task of feature selector is to use feature selection techniques to reduce the dimensionality of input space. Finally, the reduced features' set is given as an input to a number of classifiers that raise the final alarm.

2 Related Work

Monitoring the performance of a system in an automatic fashion is an active area of research. In [8], `SysProf` is presented that can monitor performance parameters of network applications at different granularity of time period. Moreover `SysProf`, also requires active user feedback to tune its different parameters and identify network related bottlenecks. Other tools for example `Paradyn` [11] exist but it only analyzes the performance of application level programs. In comparison, our system is capable of identifying bottlenecks in a relatively large number of memory and network performance counters.

In [4] S. Duan et al have presented a comparative study in which machine learning techniques (clustering, classification and regression trees and Bayesian networks) are empirically compared for identification of different system states, states comparison and short-listing the attributes for system failures. The authors propose some important challenges in the identification of system failure when performing classification: use of high dimensional dataset without compromising accuracy is one of the main issues. Our proposed scheme is composed of a step-wise identification methodology which efficiently eradicates all the three problems. We resolve high dimensionality of our dataset by monitoring and classifying performance parameters of the system independently.

[1] It is important to note that we collect the logs of selected performance parameters because maintaining logs of each process significantly increases the processing overhead of the logging process.

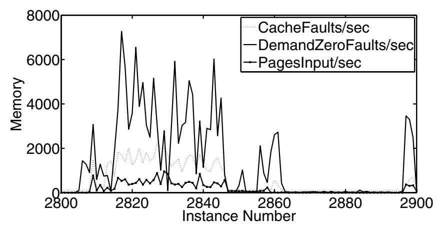
3 EvoPerf: Architecture and Functionality

We present the architecture of our `EvoPerf` utility in Figure 1 that consists of three sub-modules: (1) object monitor, (2) feature extractor, and (3) classifier. We now describe the functionality of each module.

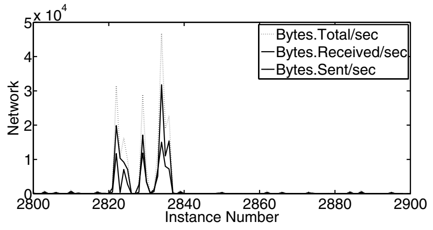
3.1 Object Monitor

As mentioned before, this module captures logs of different *objects* of the operating system. Each object, provides one or more *counters* that represent a particular performance indicator of a given computer system. The values of the counters are updated after periodic intervals. One can select any `object monitor` and log its associated counters with the help of `Perfmon` utility. In EvoPerf, we use two types of objects: (1) Memory and paging, and (2) Network.

Memory and Paging. The memory and paging objects depict the behavior of physical and virtual memory of a computer system. We know that physical memory is fast random access memory (RAM) on a computer; while virtual memory consists of RAM and secondary storage on the hard disk. A number of counters in the paging object monitor the information transfer – in the unit of fixed size memory chunks (pages) – between RAM and virtual memory. Thrashing is a special scenario in which a processor spends all its time in moving pages from the main memory to the virtual memory and vice versa. In this scenario, the response of a system might become significantly degraded that might eventually result in a denial of service [2]. We used 33 counters associated with the memory and paging objects. Some of these counters are described in Table 1. Just to substantiate the thesis that counters of memory objects can be used to monitor performance, we show a time series plot of three counters: CacheFaults/sec, DemandZeroFaults/sec and PagesInput/sec, in Figure 2(a) and it is obvious that the value of these counters are perturbed during the bottleneck period in between instance number 2810 and 2850.



(a) `Memory`



(b) `Network`

Fig. 2. Performance bottleneck plot

Table 1. Memory Counters

Pool.Nonpaged.Allocs:	is the number of calls to allocate space in the nonpaged pool.
Pool.Paged.Allocs:	is the number of calls to allocate space in the paged pool.
Pages.Input/sec:	is the rate at which pages are read from disk to resolve hard page faults.
Pages/sec:	is the rate at which pages are read from or written to disk to resolve hard page faults.
Committed.Bytes:	is the amount of committed virtual memory, in bytes.
Committed.Bytes.In.Use:	is the ratio of Committed Bytes to the Commit Limit.
Cache.Faults/sec:	is the rate at which faults occur when a page sought in the file system cache is not found.
Page.Reads/sec:	is the rate at which the disk was read to resolve hard page faults.
System Cache Resident Bytes:	gives the size of pageable operating system code present in the cache of file system.
System Code Resident Bytes:	the size of operating system code in memory available to be written to physical disk.
System Driver Resident Bytes:	gives the size of the pageable physical memory being used by device drivers.
Pool Paged Resident Bytes:	is the sampled size of paged pool in bytes. Paged pool describes the area of physical memory under use of operating system for writing available objects to the disk.
pagefile.sys	The amount of the Page File instance in use in percent
System.Driver.Total.Bytes	bytes of the pageable virtual memory currently being used by device drivers.
Free.System.Page.Table.Entries	is the number of page table entries not currently in used by the system.
Cache.Bytes.Peak	gives the maximum number of bytes used by the file system cache since the last system restart.
Pool.Nonpaged.Bytes	is the size, in bytes, of the nonpaged pool.
Cache.Bytes	is the sum of the System Cache Resident Bytes, System Driver Resident Bytes System Code Resident Bytes and Pool Paged Resident Bytes counters.
Available.MBytes	is the physical memory available to processes running on the computer, in Megabytes
Available.Bytes	is the physical memory, in bytes, available to processes running on the computer.
Available.KBytes	is the physical memory available to processes running on the computer, in Kilobytes

Network. Network activity is a key element in identifying bottlenecks in computers that are connected on the network. A computer system typically consists of multiple wired and wireless interfaces. The counters of network interface object mostly consist of volumetric traffic statistics and connection errors. The majority of network activity consists of TCP or UDP traffic (in case of TCP/IP network); therefore, we log counters of TCP and UDP objects together with the network interface objects[2]. TCP activity mostly results because of internet browsing. We use 48 counters which are associated with the network interface, TCP and UDP objects. The description of selected Network counters is in Table 2. Figure 2(b) shows the plots of three important network counters – BytesTotal/sec, BytesReceived/sec and BytesSent/sec. We can see the values of these counters change significantly from instance number 2820 to 2840 – an interval in which the bottleneck was created. We can see that bottleneck activity occur at the same instances in both memory and network objects. This is because heavy network activity has a direct effect on the memory of the system, so a bottleneck at the network interface causes a bottleneck on the memory.

3.2 Feature Extractor

One can appreciate the fact that our initial list of features' set consists of 33 memory and 48 network counters. It means that we need to keep track of 81 counters in our system which will not only increase the logging overhead but also increase the dimensionality space of our feature set. Therefore, it becomes relevant to use well know feature selection techniques to reduce the number of counters in our features' set.

We utilize two well known schemes for feature selection: (1) information gain [19], and (2) chi-square method [20]. We provide our raw features' set – obtained from the object module – to these feature ranking schemes. Both schemes rank features separately on the basis of a feature's ability or role in enhancing the

Table 2. Network Counters

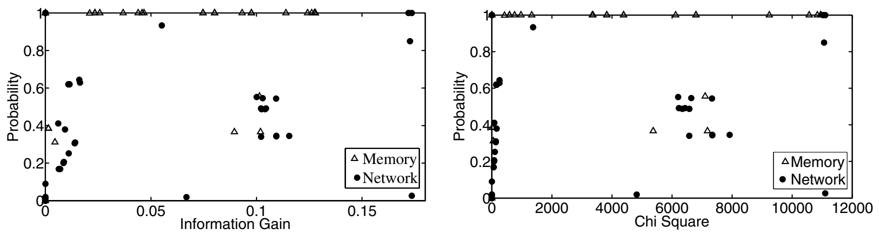
TCP Segments Sent/sec:	gives the rate at which TCP segments are sent.
TCP Segments Received/sec:	gives the rate at which TCP segments are received, this includes segments received in error.
Packets Received/sec:	gives the rate at which packets are received on the network interface.
Packets Received Unicast/sec:	gives the rate of (subnet) unicast packet delivery to a higher-layer protocol.
Bytes Total/sec:	gives the rate of sending/receiving bytes over the network adapter.
Bytes Received/sec:	is the rate at which bytes are received over each network adapter.
Packets/sec:	is the rate at which packets are sent and received on the network interface.
TCP Segments/sec:	is the rate at which TCP segments are sent or received using the TCP protocol.
OutputQueueLength	is the length of the output packet queue (in packets).
TCPConnectionsEstablished	gives the number of TCP connections whose current states are either ESTABLISHED or CLOSE-WAIT.
TCPConnections.Active	is the number of TCP connection transition from the CLOSED state to the SYN-SENT state.
TCPConnections.Reset	is the number of direct TCP connection transition to the CLOSED state.
TCPConnections.Passive	is the number of direct TCP connection transition to the SYN-RCVD state from the LISTEN state.
Packets.Outbound.Errors	is the number of outbound packets that were not transmitted because of errors.
UDPDatagrams.Received.Errors	is the number of received UDP datagrams that were not delivered due to reasons excluding failure of application at the destination port.
TCPConnection.Failures	is the number of times TCP connections have made a direct transition to the CLOSED state from the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition to the LISTEN state from the SYN-RCVD state.
Bytes.Sent/sec	is the rate at which bytes are sent over each each network adapter.
Packets.Sent.Unicast/sec	is the rate at which packets are requested to be transmitted to subnet-unicast addresses by higher-level protocols.
Packets.Sent/sec	is the rate at which packets are sent on the network interface.

Table 3. Feature Selection

Objects	**Memory**	**Network**
Threshold IG	0.089	0.067
Feature IG	15	25
Threshold CS	5374.5	4823.9
Feature CS	15	25

classification accuracy. After applying both schemes, the number of features – indicated in Table 3 – reduces from 81 to 40 (15 for memory and 25 for network). We have selected only those top ranked features which are common in both schemes. We now provide a brief description of each scheme to make the paper self contained.

Information Gain. Information gain is an entropy based information theoretic measure. A feature with higher information gain will have higher classification power and vice versa. For a given attribute X and a class attribute Y, the



(a) Information Gain (b) Chi-squared

Fig. 3. Normal Probability Plot

uncertainty is given by their respective entropies $H(X)$ and $H(Y)$. Then the information gain of X with respect to Y is given by $I(Y;X)$, where

$$I(Y;X) = H(Y) - H(Y|X)$$

Table 3 shows the threshold values of information gain for memory and network objects. Figure 3(a) shows the normal probability distribution plot of information gain of all features [19].

Chi-Squared Statistics. The χ^2 method performs its feature selection by the use of chi-squared statistics of each feature with respect to its class. Initially χ^2 value of all features is calculated. The χ^2 is calculated as:

$$\chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{(O_{ij} - E_{ij})}{E_{ij}} \tag{1}$$

where n is the number of intervals, k is the number of classes, O is the number of samples and E is the expected frequency. Table 3 shows the threshold χ^2 values. The features with χ^2 values greater then the threshold are eventually selected. Figure 3(b) plots the normal probability distribution of Chi-Square method of all features.

A closer look at Figure 3(a) and Figure 3(b) reveals that the memory and network objects follow approximately the same distribution pattern with minor differences. Therefore, a correlation of the top ranked features only, removes any discrepancy in the selected features' set. This analysis shows that `Available Bytes` – a counter of memory object – is an important feature even though it is a middle ranking feature. The reason is that a reduction in the available memory counter depicts a serious bottleneck because it could lead to an eventual denial of service. So the integrity of the selected feature set is an important issue [20].

3.3 Classifier

We have selected a number of well known classifiers in order to evaluate their feasibility for our EvoPerf. The choice of six classifiers is as following:(1) UCS is a state-of-the-art Michigan-style classifier [13], (2) two state-of-the-art Pittsburgh-style classifiers – GAssist-ADI [16] and GAssist-Intervalar [15], (3) two state-of-the-art neural network based classifiers – MLP [21] and RBF [22] and (4) a decision tree J48 [23]. The purpose of using classifiers from diverse learning paradigms is to select a classifier that provides the best accuracy with minimum processing overheads. (Remember in our case real time deployment of the system is very important.) We have used implementations of evolutionary classifiers – UCS [13], GAssist-Int [15], GAssist-ADI [16]– provided in toolkit `KEEL` [12]; while for neural networks – RBF and MLP – we have used `WEKA` [5]. We have used implementation of J48 in `WEKA`. We empirically determined the best configurations for different parameters of each classifier.[2]

[2] Parameters values of RBF: clustering seed = 1, minStdDev = 0.1, numClusters = 2, ridge = 1.0E-8.

4 Experiments

We have collected the performance logs on a computer system in our networks lab. The hardware specifications of the system are: Intel(R) Core2Duo 1.8 GHz CPU, 2GB of RAM and 160GB of physical drive. We utilized Windows `Perfmon` utility for monitoring performance counters. We have collected two sets of performance counter logs: normal and artificially created bottleneck. We have selected a sampling rate of 12 samples per minute. We have monitored a user's activity on the system for more than 15 hours over a period of 3 days to get a better idea about the normal usage behavior. Later, we have created a number of performance bottlenecks by maximizing network and memory usage of the system for a period of 15 minutes. The results of our experiments show that the six classifiers – UCS, GAssist-ADI, GAssist-Int, NN-MLP, NN-RBF and J48 – provide approximately the same accuracy. However, Neural Networks (NN) based classifiers have significantly smaller processing overhead, but when compared to machine learning algorithms, J48 outperforms the rest of the classifiers. This makes machine learning algorithms suitable for real world deployment.

We now report the results of our experiments. We follow a 10-fold cross validation strategy in all experiments. The dataset of each object is divided into 10 folds, 9 folds are used for training and the rest is used for testing. The process is reported for all folds and we report an average value of all folds.

Table 4. Results of Experiments on Raw Features' Set

	Features		UCS	Gassist-Int	Gassist-ADI	RBF	MLP	J48	Average
Memory	33	Acc	0.994	0.999	0.999	0.999	0.999	0.995	0.998
		TP	140	194	195	195	195	195	
		TN	10805	10811	10812	10812	10812	10812	
		FP	56	2	1	1	1	1	
		FN	7	1	0	0	0	0	
Network	48	Acc	0.999	1	1	1	0.997	1	0.999
		TP	272	288	288	288	160	288	
		TN	10813	10813	10813	10813	10796	10813	
		FP	16	0	0	0	17	0	
		FN	0	0	0	0	16	0	

The results are tabulated in Table 4. It is obvious from the results of our evaluation that all classifiers provide approximately the same accuracy. Therefore, we now need to focus on our next objective: to reduce the processing overhead of the classifiers. Towards this end, we evaluate the impact of feature selection module.

5 Results and Discussions

In this section, we discuss the results obtained once we apply features' selection techniques (see Table 5). Once we compare the results reported in Table 4 with those of Table 5, it is clear that the accuracy of the classifiers remain (almost) unaffected even with a reduced features' set. Now we analyze the impact of features' reduction on training and testing times of different classifiers.

Table 5. Results of Experiments with Selected Features' Set

	Features		UCS	GAssist-Int	GAssist-ADI	MLP	RBF	J48	Average
Memory	15	Acc	0.998	0.999	0.999	0.999	0.999	0.995	**0.998**
		TP	183	192	195	195	195	195	
		TN	10811	10812	10811	10812	10811	10812	
		FP	1	0	1	1	1	1	
		FN	13	4	1	0	1	0	
Network	25	Acc	0.993	1	1	0.997	1	1	0.998
		TP	207	288	288	288	288	288	
		TN	10813	10813	10813	10813	10813	10813	
		FP	0	0	0	0	0	0	
		FN	81	0	0	0	0	0	

5.1 Timing Analysis

We run two sets of experiments: (1) measure training and testing times once classifiers are using raw features' set, and (2) repeat the same experiments as in (1) but with reduced features' set. The obtained results for the first case are tabulated in Table 6. It is obvious from the table that J48 has the smallest training times while other classifiers take significantly large amount of time – GAssist-ADI is the worst – making them infeasible for realtime deployment on a computer system. Similarly J48 takes almost the same time for testing as the neural networks based classifiers.

Table 6. Testing and Training times (seconds) of classifiers using all features

Parameter	**Memory**		**Network**	
	Training Time	**Testing Time**	**Training Time**	**Testing Time**
UCS	608.1766	31.0547	523.4749	42.2907
GAssist-ADI	2307.7	114.34	2187.12	81.6
GAssist-Int	2254.98	107	2096.54	78
NN-MLP	977.47	0.16	860.35	0.13
NN-RBF	6.33	0.04	6.35	0.08
J48	1.52	0.07	1.47	0.1

Table 7. Testing and Training times (seconds) of classifiers using selected features

Parameter	**Memory**		**Network**	
	Training Time	**Testing Time**	**Training Time**	**Testing Time**
UCS	261.9124	15.2766	236.2954	14.7342
GAssist-ADI	1199.85	62.4	1018	27.4
GAssist-Int	1050.6	53.5	996.52	24
NN-MLP	137.95	0.02	113.21	0.02
NN-RBF	4.51	0.03	2.2	0.03
J48	0.57	0.04	0.66	0.04

Table 7 shows the training and testing times of different classifiers once we have reduced our features' set. The results prove our thesis: the training and testing times of all classifiers are reduced more than 50% due to features' selection. Again, J48 is having the smallest training and testing time without compromising on the detection accuracy.

6 Conclusions

The major contribution of the paper is an online real time autonomous performance monitoring system that has the capability to detect bottlenecks without user intervention. In a large network of interconnected systems, the proposed system can significantly reduce the workload of a system administrator by relieving him of man-in-the-loop analysis; as a result, he can focus his attention on countermeasure strategies. Our research shows that using performance counters of memory and network objects, classifiers can identify bottlenecks with a high accuracy. Our future work involves using other objects and analyzing the robustness of the system to evasion strategies.

Acknowledgments

The research presented in this paper is supported by the grant # ICTRDF/TR&D /2007/13 for the project titled ISKMCD, by the National ICT R&D Fund, Ministry of Information Technology, Government of Pakistan. The information, data, comments, and views detailed herein may not necessarily reflect the endorsements of views of the National ICT R&D fund.

References

1. Perfmon, Microsoft Technet, Microsoft Corporation, http://technet.microsoft.com/en-us/library/bb490957.aspx
2. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts, 7th edn. John Wiley & Sons Inc., Chichester (2004)
3. Ji, Z., Dasgupta, D.: Revisiting Negative Selection Algorithms. Evolutionary Computation Journal 15(2), 223–251 (2007)
4. Duan, S., Babu, S.: Empirical Comparison of Techniques for Automated Failure Diagnosis. USENIX (2009)
5. Witten, I.H., Frank, E.: Data mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, USA (2005)
6. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computer Survey 31(3), 264–323 (1999)
7. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: International Conference on Machine Learning (ICML), pp. 727–734. Morgan Kaufmann, USA (2000)
8. Agarwala, S., Schwan, K.: SysProf: Online Distributed Behavior Diagnosis through Fine-grain System Monitoring. In: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, p. 8 (2006)
9. Agarwala, S., Poellabauer, C., Kong, J., Schwan, K., Wolf, M.: Resource-Aware Stream Management with the Customizable dproc Distributed Monitoring Mechanisms. In: Proc. of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12), Seattle, WA (June 2003)
10. Massie, M., Chun, B., Culler, D.: The Ganglia Distributed Monitoring System: Design, Implementation, and Experience (2003)

11. Miller, B.P., Callaghan, M.D., Cargille, J.M., Hollingsworth, J.K., Irvin, R.B., Karavanic, K.L., Kunchithapadam, K., Newhall, T.: The Paradyn parallel performance measurement tool. IEEE Computer 28(11), 37–46 (1995); G.K. Williams
12. Alcala-Fdez, J., Garcia, S., Berlanga, F.J., Fernandez, A., Sanchez, L., del Jesus, M.J., Herrera, F.: KEEL: A data mining software tool integrating genetic fuzzy systems, Genetic and Evolving Systems (GEFS). In: 3rd International Workshop, vol. 4(3), pp. 83–88 (2008)
13. Mansilla, E.B., Guiu, J.M.G.: Accuracy-Based Learning Classifier Systems: Models, Analysis and Applications to Classification Tasks Ester. Evolutionary Computation 11(3), 209–238 (2006)
14. Rivest, R.: Learning Decision Trees. Machine Learning 2, 229–246 (1987)
15. Bacardit, J., Garrell, J.M.: Bloat control and generalization pressure using the minimum description length principle for a Pittsburgh approach Learning Classifier System. In: Kovacs, T., Llorà, X., Takadama, K., Lanzi, P.L., Stolzmann, W., Wilson, S.W. (eds.) IWLCS 2003. LNCS (LNAI), vol. 4399, pp. 59–79. Springer, Heidelberg (2007)
16. Bacardit, J., Garrell, J.M.: Evolving Multiple Discretizations with Adaptive Intervals for a Pittsburgh Rule-Based Learning Classifier System. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1818–1831. Springer, Heidelberg (2003)
17. Zheng, Z., Lan, Z., Park, B.-H., Geist, A.: System Log Pre-processing to Improve Failure Prediction. In: Proc. of DSN 2009 (2009)
18. Salfner, F., Tschirpke, S.: Error Log Processing for Accurate Failure Prediction. In: Proc. of WASL 2008, in conjunction with OSDI (2008)
19. Liu, H., Li, J., Wong, L.: A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns. Genome Inform. 13, 51–60 (2002)
20. Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes. In: Proc. IEEE 7th International Conference on Tools with Artificial Intelligence, pp. 338–391 (1995)
21. Moller, F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, 525–533 (1990)
22. Broomhead, D.S., Lowe, D.: Multivariable Functional Interpolation and Adaptative Networks. Complex Systems, 321–355 (1988)
23. Quinlan, J.R.: C4. 5: programs for machine learning. Morgan Kaufmann, San Francisco (1993)

Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation

Bernd Bischl[2], Igor Vatolkin[1], and Mike Preuss[1]

[1] Chair of Algorithm Engineering, TU Dortmund, Germany
[2] Chair of Computational Statistics, TU Dortmund, Germany

Abstract. Classification of audio recordings is often based on audio-signal features. The number of available variables is usually very large. For successful categorization in e.g. genres, substyles or personal preferences small, but very predictive feature sets are sought. A further challenge is to solve this feature selection problem at least approximately with short run lengths to reduce the high computational load. We pursue this goal by applying asymmetric mutation operators in simple evolutionary strategies, which are further enhanced by mixing in greedy search operators. The resulting algorithm is reliably better than any of these approaches alone and in most cases clearly better than a deterministic greedy strategy.

Keywords: Evolutionary Strategies, Asymmetric Mutation, Feature Selection, Music Information Retrieval, Machine Learning.

1 Introduction: Motivation and Music Classification as Optimization Problem

Personal digital music collections have been growing rapidly during the recent years. Approaches for smart navigation through large audio libraries or recommendation techniques provide obviously needed remedies for music management. Supervised audio classification methods build models from labeled music examples and extracted features.

A large number of parameters have an impact on the classification results, e.g. for feature extraction, different feature source time frames can be selected. Larger frames allow more precise frequency resolution; however, if they are too large, several notes can be mixed and it will be harder to learn anything at all from the spectrum distribution. Feature processing optimization may select the optimal feature set or choose different preprocessing methods. On the other hand, hyperparameters or model classes of classification methods can be optimized over for achieving better performance.

Because of the nonlinear interactions between different parameters, search for optimal or sufficient solutions can profit from heuristics. Continuing our previous work [15], [16], we concentrate here on the experiments for feature selection by evolutionary strategies (ES). Since very different optimal feature sets may exist for different music categories (as shown e.g. in [11]), the search for the best features must be done for each category separately. Several facts motivate the additional search for rather small feature sets: Firstly, the storage of more features requires large indexing disc space. Secondly, the algorithms for extraction, processing and classification need more computing time. Moreover, classification models built from larger feature sets increase the danger to overfit the currently used song sets, leading to a much weaker performance in general. Introducing a bias towards less features can be seen as a regularization method to produce more stable solutions. Also, smaller sets provide a better possibility to interpret the less complex classification model.

A simple (1+1)-ES [1] already obtains a good result, however it often selects relatively large feature sets, and is sometimes beaten by a simple greedy strategy. Obviously, we have a two-criteria problem which could be approached by a multi-criterial evolutionary algorithm (EA). We abstain from doing so because the reasonably available amount of function evaluations is very low (in the order of a few hundred). In this work, we suggest to encode the need for small datasets directly into the mutation operator via asymmetric mutation. Moreover, we show that importing parts of the greedy strategy into the ES further improves performance, leading to a reliable and well performing method that by design choses only small feature sets.

2 Music Genre Classification and Feature Selection

2.1 Genre Classification

In music classification, generally a set of raw features are extracted from segments of a song, which describe different characteristics like timbre, harmony, melody, rhythm or time and structural properties (for details see [16]). Using these covariates and some given labels, which specify the genre of the song, a model is built by using one of the many classification algorithms from machine learning. Models are evaluated by applying them to new data, which were not used during training and are assumed to be i.i.d. drawn from the same data generating process, and their predictions are measured by applying an appropriate loss function. Often this is zero-one loss for classification, but due to the segmentation of songs into multiple parts we use a mean squared error on song level

$$E^2 = \frac{1}{L} \sum_{i=1}^{L} (\hat{s_i} - s_i)^2 \,. \tag{1}$$

Here, L is the number of songs and the s_i are their true labels. As we only consider binary categories, these will always be from $\{0, 1\}$.

$$\hat{s_i} = \frac{1}{P} \sum_{j=1}^{P} \hat{g}(x_{ij}) \tag{2}$$

is a “voting score”, obtained by applying the fitted learning machine $\hat{g}$ to all partitions of a song and averaging the predictions. The x_{ij} are the feature vectors corresponding to the P partitions of song s_i.

It is usually unknown which learning algorithm performs best for a task at hand, therefore extensive model selection (e.g. by cross-validation) among the different classes of inducers has to be performed. As we want to focus on the feature selection algorithms and maintain comparability to our previous work, we will only consider decision trees though. Further, the CART methodology [3] is very fast, does not require tuning of many hyperparameters, and performs well in comparison to many other data mining methods.

2.2 Feature Selection

There are at least two (somewhat contradictory) viewpoints of feature selection in machine learning: On the one hand, it is well known that in the setting of many noisy, highly correlated and possibly irrelevant covariates the predictive power of a model fitted on the whole feature set will be suboptimal [4]. Thus, feature selection might be employed mainly as a method to improve generalization performance. On the other hand, one could select variables to construct smaller, and therefore more interpretable models, if one is mainly interested in understanding the data. In this case one might even accept a substantial loss in predictive performance to achieve a smaller model. We will follow the former perspective here.

One of the most popular class of feature selection algorithms - because of their general applicability and strength to build very predictive models - are wrappers [7]. These algorithms internally use a learning algorithm as a black-box and search for an optimal set of input features w.r.t. the learner by repeatedly adjusting the variable set, fitting a model and evaluating it. This basically reduces the feature selection problem to a discrete, binary optimization problem, with a possibly noisy target function, as the target value can vary because of either the stochastic nature of the resampled training and test sets or because of a stochastic model fitting scheme. For an overview of competing feature selection techniques,like filters and embedded methods, see [4].

2.3 Categories and Feature Sets

For the following music classification experiments we created an mp3 database with 120 commercial albums, whose songs are labeled as Classic, Pop/Rock, Rap, Electronic, R&B, ClubDance and Heavy Metal by AllMusicGuide[1]. The relations between songs and categories were manually created by the AMG music experts. As demonstrated in [11], different features might be relevant to identify each category, so we transformed the multi-class problem into seven binary ones by “one-vs-rest”.

For the training of classification models we employ two audio feature sets: An older one with 198 features (most of them described in [14]) and a newer one consisting of 572, which were extended by MIR Toolbox functions (see [8]) like tonal centroids, fluctuation patterns, etc. It is possible to incorporate other feature groups (playlists and tags from the web community, metadata etc.), however they are not always available or can be erroneous - only audio signal based features guarantee their availability for the automatic extraction for each music piece from any personal music collection.

In contrast to the experiments in [16], the dimensions of the multidimensional features, like twelve chroma vector values, are treated as independent variables, so that it is e.g. possible to use only the first and third chroma characteristics, discarding all

[1] www.allmusic.com, visited in April 2010.

others. This allows more flexible feature selection while increasing the complexity of the optimization problem.

3 Search Strategies

For large feature spaces the wrapper approach is computationally expensive and its performance obviously depends on the efficiency of the used search algorithm. It should be mentioned though, that more complex search methods as well as very long runs or even exhaustive searches do not necessarily equal superior results in generalization performance, as overfitting might occur on this level as it possibly does on the lower level of model fitting itself [9].

As we are mainly interested in the properties of the optimization techniques itself, we don't employ a more appropriate, but even more time-consuming setup like nested cross-validation. Instead use 20 songs in the training set (each with roughly 100 segments) as prototypes for the categories to be learned, and a rather large optimization set of 120 songs for feature set evaluation to avoid overfitting through extensive search. We also regularize the search by only allowing a rather low number of 500 function evaluations [10].

3.1 Monte Carlo Search (MC / Random)

As a baseline comparison we use a random search which simply draws bit vectors from a binomial distribution, evaluates the corresponding feature sets w.r.t. the optimization set and selects the one with the minimal E^2. Therefore, feature f_i occurs with probability 0.5 in every random feature set and on average the random feature sets are half as large as the full set.

3.2 Greedy Forward Search with Correlation Heuristic (GFS)

Because the commonly used sequential forward selection cannot be used due to the restricted number of function evaluations, we construct a variant which rapidly moves through the variables in a greedy, heuristically guided order: On the training set we rank the features by considering the absolute empirical correlation $|\rho(\mathbf{f}_i, \mathbf{y})|$ between the feature vector $\mathbf{f}_i$ and the binary label vector $\mathbf{y}$ of the song segments. Now, starting from the empty set, we successively try to add variables to the current set in the order of their ranking. If a variable improves the performance on the optimization set, it is accepted, otherwise not and the following variable in the ranking is tried.

This technique is sometimes called Rank-Search and variants of it are proposed in [5,12].

3.3 Evolutionary Strategy with Local Operators (ES-LO)

We consider the (1+1)-ES with local, hybrid operators to select more uncorrelated feature sets as introduced in [16]. To obtain smaller feature sets, we enhance it by an asymmetric mutation operator as in [6], using different mutation probabilities for set and unset bits. Let the current feature set be a binary vector **m** with length N. The mutation is a bit flip for the ith feature with probability

$$p_m(i) = \frac{\gamma}{N}|m_i - p_{01}| \tag{3}$$

γ impacts the general mutation probability and can be interpreted as a step size. p_{01} controls the balance between the number of 1- and 0-entries in the bit feature vector [6]. If e.g. only five percent of ones are desired, p_{01} is set to 0.05. It can be seen as a probability to switch a bit on, if it is currently not in the solution, and $1 - p_{01}$ as a probability to switch it off, if it is.

After the successful mutations we apply a neighborhood search. Here the mean empirical correlation between a feature vector **f** and the M features currently in the set is calculated:

$$R(\mathbf{f}) = \frac{1}{M}\sum_{j=1}^{M}|\rho(\mathbf{f}, \mathbf{f}_j)| \, . \tag{4}$$

Briefly speaking, the local search operator LO^+ adds a new feature with the smallest $R(\mathbf{f})$, trying to extend the feature set with another covariate which is least correlated to the current. LO^- removes the feature with the highest $R(\mathbf{f})$, removing the covariate with the highest correlation with the current features.

3.4 Evolutionary Strategy (ES)

This is the same search algorithm as the previous ES-LO, simply without the local operators. We consider this algorithm in order to analyze whether the previously mentioned local operators really improve the efficiency of the search.

3.5 Evolutionary Strategy with Success Rule Adaptation (ES-SRA)

One of the early attempts to adapt the mutation strength of an ES was the 1/5 success rule, with increasing mutation strength for more than 1/5 successes, and decreasing mutation strength for less than 1/5 successes [18]. However, this method was envisioned for real-valued search spaces and as Schwefel has shown at the same time, does not work reliably for discretized search spaces [13]. The reason for this failure is that the problem must provide maximal success rates for minimal steps, which is not generally the case already for integer variables, let alone for bit vectors. We assume that it is also not the case for our problem and thus use a different rule that increases the mutation probability $p_m(i)$ by a factor of 1.2 if less than 1/10 successes are recorded and decrease it by the same factor if more than 3/10 successes are recorded. As the number of function evaluations for one run is very limited, we employ a small window size of 10 for measuring success rates. This method may be interpreted as 'controlled restart' to get out of an area where Hamming-1 distance steps are very rarely successful. Interestingly, our scheme has a predecessor in [19], where for minimal success rates the mutation strength is doubled.

3.6 Evolutionary Strategy with Greedy Heuristic (ES-GH)

This is the same search algorithm as the plain ES, but with a modified mutation operator. As section 4 will show, GFS performs very well on some data sets. Therefore we try to encode its greedy heuristic into the mutation operator of the ES:

$$p_m(i) = \frac{\gamma}{N}|m_i - p_{01}| \cdot |m_i - |\rho(\mathbf{f_i}, \mathbf{y})|| \qquad (5)$$

The second factor in $p_m(i)$ forms a correlation-based probability to switch bits on with higher probability, which are highly correlated with the target, and to switch bits off which are not. We simply multiply the two probabilities from the mutation operator of the normal ES and the correlation-based heuristic to form a combined probability and incorporate both advantages. Section 4 also shows, that SRA is essential for the success of the optimization, therefore we use it here as well to adapt the step size γ.

4 Experimental Assessment of Search Strategies

We are most interested in assessing the performance of the local operators (LO), the success-rule adaptation (SRA) and the greedy-heuristic (GH) enhanced ES variants against the simple Monte Carlo search and the Greedy Forward Search (GFS). This is tested in two steps, first LO and SRA, and the most successful variant is then tried with and without GH enhancement. Possible parameter variations and representation issues are either used throughout the following experiments (asymmetry of mutations) or dealt with in the pre-experimental planning phase (mutation strength and base feature set). The overall goal of our experiments is to compose the best achievable ES variant and test if it is reliably better than MC and GFS.

For our experiments we used the statistical programming language *R* [17] and the package *mlr* [2], which allows to select from a wide range of machine learning, variable selection and hyperparameter tuning methods.

Experiment: Do local operators and/or success rule adaptation of mutation strenghts improve ES performance?

Pre-experimental planning. First experimentation showed that for ES, ES-LO and ES-SRA the mutation probability parameter γ can be set to 16 for the old and new feature sets. Comparing the MSE values attained for these two representations leads to the conclusion that the new, larger feature sets most often allow finding better classifiers. In the following, we therefore fix γ at 16 and employ only the new feature sets. Moreover, the run lengths are fixed to 500 function evaluations as a good compromise between achievable performance and consumed real time (around 2h for one run on a modern PC). For the same reason, we perform only 5 repeats.

Task. A method is recognized as better than another, if its average final MSE is better in at least 4 of the 7 categories.

Setup. We test 3 ES variants against each other, ES-NLO-SRA (ES without local operators but with success rule adaptation), ES-LO-NSRA (ES-LO without SRA) and ES-LO-SRA (both switched on), each time with asymmetric bit flip probability set to

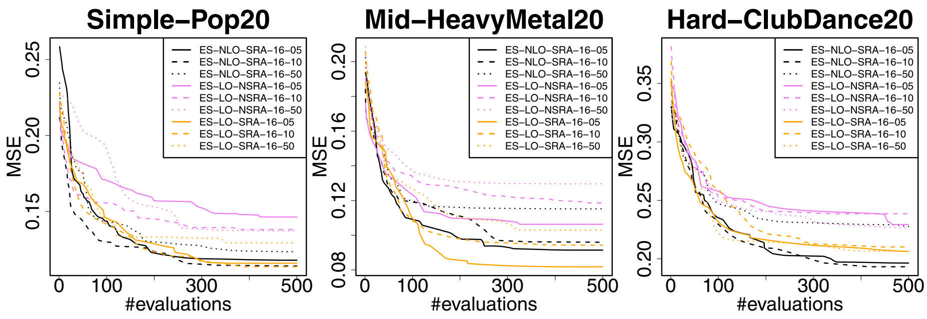


Fig. 1. Comparison of different ES variants. Curves are averaged over 5 runs.

$p_{01} \in \{0.05; 0.10; 0.50\}$. All runs are done over all 7 categories. Note that $p_{01} = 0.5$ means symmetric mutation.

Results/Visualization. The average performance of all variants is plotted in fig. 1 for 3 of the 7 categories[2]. The best recorded variant of our comparison is ES-NLO-SRA-16-05 (no local operators, success rule adaptation, $p_{01} = 0.05$, by winning 4 categories and being placed second, otherwise.

Observations. In general (over all categories), we can state that the local operators do not increase performance but sometimes decrease it. However, the success rule adaptation is obviously necessary to reach good MSE values. Concerning the mutation probabilities, the results get the better, the stronger the deviation from symmetry is.

Discussion. It is no surprise that the asymmetric mutation improves performance as sparsely selecting alternative algorithms (e.g. greedy feature selection) also cope well with the treated problems, meaning that there must be good small feature sets.

The failure of the local operators can be explained by considering the extreme case of completely irrelevant, random features. This will neither be removed by LO^-, as they are completely uncorrelated to all features currently in the set, and they will constantly be proposed to be added by LO^+ for exactly the same reason. We also verified this undesirable effect on synthetic data sets, were the truly relevant features are known.

However, it is more difficult to explain why the success rate adaptation does so well. We therefore visualize the current feature set and its Hamming-1 distance throughout a typical run of ES-NLO-SRA-16-05 in fig. 2. It gets clear that in higher generations, most 1-bit flips are neutral, and there are only very few 1-bit mutations left that lead to an improvement. In such situations, it makes sense to increase the mutation probability to flip several bits at once.

Experiment: Is a greedy-heuristic enhanced ES reliably better than GFS/MC?

Pre-experimental planning. In initial tests, we try several ES-GH parameterizations and set the p_{01} again to 0.05 and $\gamma = 32$ as the flip probabilities are reduced by the multiplication in (5).

[2] Complete results and plots are available as supplementary material from `http://www.statistik.uni-dortmund.de/~bischl/ppsn2010_suppl`

Task. As before, an algorithm is termed better than another one if it is better on average in at least 4 categories.

Setup. We run the best method of experiment 1 (ES-NLO-SRA-16-05) against the full featured classifier, random search, greedy forward search and ES-GH (with SRA), again over all 7 categories.

Results/Visualization. Figure 3 reports the (averaged where applicable) performance of all algorithms on 3 of 7 categories.

Observations. The greedy-enhanced ES-GH performs consistently better than ES-NLO-SRA-16-05, and is able to achieve better MSE values than GFS in 5 of 7 cases. MC and the full featured classifier are much worse in all cases. Over all categories, the ES-GH seems to be much more robust than GFS which sometimes fails dramatically. Even for the 2 lost categories, ES-GH achieves acceptable results.

Discussion. Importing a greedy mechanism into the previously best ES variant obviously leads to the envisioned effect: We can combine the best of both methods by obtaining a reliable algorithm that always performs well and is in most cases better than all others we tried.

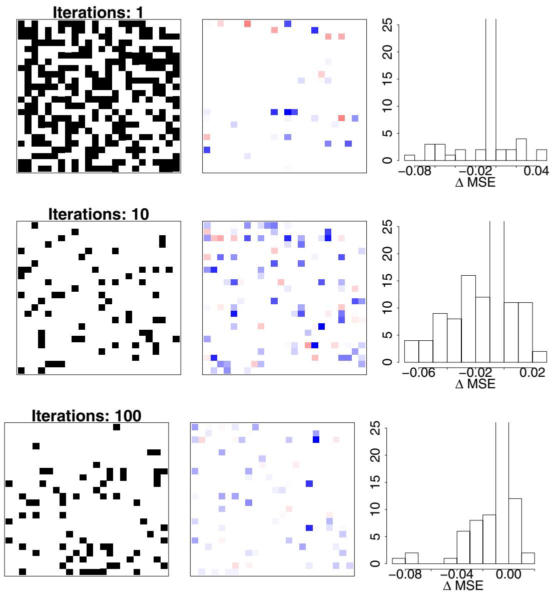


Fig. 2. Analysis of Hamming-1-neighborhood for a run of ES-NLO-SRA-16-05, logged at different numbers of function evaluation during the run. The left pattern shows the variables currently in the feature set, a black square means the bit is switched on. The middle plot is produced by flipping each bit separately and measuring the difference in MSE to the current solution. A positive value means improvement. A color gradient from white to red is used for improvement, and from white to blue if flipping the bit worsens the solution. The right plot simply shows a histogram of the differences in MSE from the middle plot, capped at 25.

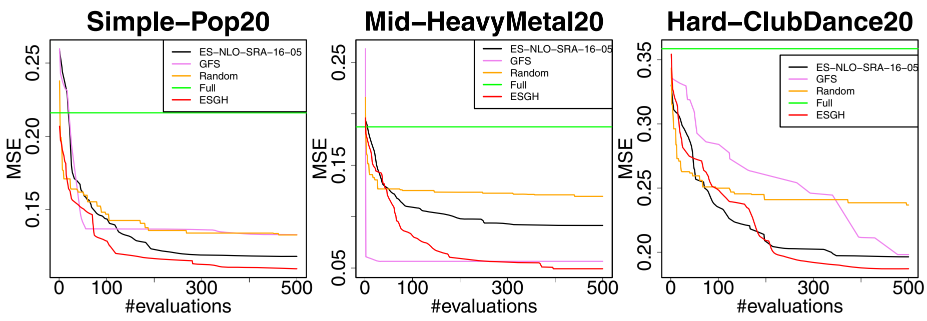


Fig. 3. Comparison of best ES variant of experiment 1 with baseline methods and improved ES. Curves are averaged over 5 runs.

5 Conclusions and Outlook

In this work we continued the design of ES for feature selection in music classification. We have seen that enforcing sparse selection of features by means of an asymmetric mutation operator produces competitive results after the optimization. This leads to a reduced number of features which must be computed and stored and improves the generalization performance.

We also demonstrate by experiment, that previously proposed local operators do not lead to a better algorithm, but instead might deteriorate the performance, and we also provide theoretical aspects to explain this result. A rather simple greedy Rank-Search is presented, which sometimes achieves quite impressive performance results by selecting extremely small feature sets. But this is not a reliable behavior across all data sets, so we show how to combine both advantages of the ES-LO with success rate adaptation and the GFS into a reliably well performing method.

It seems quite clear, that the full potential to include heuristics for feature selection into the stochastic optimization is yet to be explored. If one can allow for only a low or moderate number of function evaluations, guiding the search into the relevant areas of the search space quickly - even by crude measures - will be crucial for success.

Acknowledgments

We thank the Klaus Tschira Foundation for the financial support. Due to the computationally intensive nature of this case study, the majority of the calculations were performed on the LiDO HPC cluster at the TU Dortmund.

References

1. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies – A comprehensive introduction. Natural Computing: an International Journal 1, 3–52 (2002)
2. Bischl, B.: The mlr Package: Machine Learning in R (2010), `http://mlr.r-forge.r-project.org`

3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
4. Guyon, I.: An Introduction to Variable and Feature Selection. The Journal of Machine Learning Research 3, 1157–1182 (2003)
5. Hall, M.A., Holmes, G.: Benchmarking attribute selection techniques for discrete class data mining. IEEE Transactions on Knowledge and Data Engineering 15(6), 1437–1447 (2003)
6. Jelasity, M., Preuß, M., Eiben, A.E.: Operator Learning for a Problem Class in a Distributed Peer-to-peer Environment. 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. 172–183. Springer, Heidelberg (2002)
7. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–324 (1997)
8. Lartillot, O., Toiviainen, P.: MIR in Matlab (II): A Toolbox for Musical Feature Extraction From Audio. In: Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR), pp. 127–130 (2007)
9. Loughrey, J., Cunningham, P.: Overfitting in Wrapper-Based Feature Subset Selection: The Harder You Try the Worse it Gets. In: Research and Development in Intelligent Systems XXI, pp. 33–43 (2005)
10. Loughrey, J., Cunningham, P.: Early-Stopping to Avoid Overfitting in Wrapper-Based Feature Selection Employing Stochastic Search. Computer Science Technical Report, Trinity College Dublin, TCD-CS-2005-37 (2005)
11. Pohle, T., Pampalk, E., Widmer, G.: Evaluation of Frequently Used Audio Features for Classification of Music into Perceptual Categories. In: Fourth International Workshop on Content-Based Multimedia Indexing (2005)
12. Ruiz, R., Riquelme, J.C., Aguilar-Ruiz, J.S.: Incremental wrapper-based gene selection from microarray data for cancer classification. Pattern Recognition 39(12), 2383–2392 (2006)
13. Schwefel, H.-P.: Cybernetic Evolution as Strategy for Experimental Research in Fluid Mechanics. Diploma Thesis, Hermann Föttinger-Institute for Fluid Mechanics, Technical University of Berlin (1965) (in German)
14. Theimer, W., Vatolkin, I., Eronen, A.: Definitions of Audio Features for Music Content Description, Algorithm Engineering Report TR08-2-001, Technical University Dortmund (2008)
15. Vatolkin, I., Theimer, W.: Optimization of Feature Processing Chain in Music Classification by Evolution Strategies. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 1150–1159. Springer, Heidelberg (2008)
16. Vatolkin, I., Theimer, W., Rudolph, G.: Design and Comparison of Different Evolution Strategies for Feature Selection and Consolidation in Music Classification. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009). IEEE Press, Piscataway (2009)
17. R Development Core Team, R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria (2010), ISBN 3-900051-07-0, http://www.R-project.org
18. Rechenberg, I.: Cybernetic Solution Path of an Experimental Problem. In: Fogel, D.B. (ed.) Evolutionary Computation - The Fossil Record. IEEE Press, Los Alamitos (1998)
19. Greenwood, G., Zhu, Q.: Convergence in Evolutionary Programs with Self-Adaptation. Evolutionary Computation 9(2), 147–158 (2001)

Globally Induced Model Trees: An Evolutionary Approach

Marcin Czajkowski and Marek Krętowski

Faculty of Computer Science
Bialystok University of Technology
Wiejska 45a, 15-351 Białystok, Poland
{m.czajkowski,m.kretowski}@pb.edu.pl

Abstract. In the paper we propose a new evolutionary algorithm for induction of univariate regression trees that associate leaves with simple linear regression models. In contrast to typical top-down approaches it globally searches for the best tree structure, tests in internal nodes and models in leaves. The population of initial trees is created with diverse top-down methods on randomly chosen subsamples of the training data. Specialized genetic operators allow the algorithm to efficiently evolve regression trees. Akaike's information criterion (*AIC*) as the fitness function helps to mitigate the overfitting problem. The preliminary experimental validation is promising as the resulting trees can be significantly less complex with at least comparable performance to the classical top-down counterparts.

Keywords: Model trees, evolutionary algorithms, regression trees, *AIC*, simple linear regression.

1 Introduction

Data mining [7] is a process of extracting useful information, relationships and hidden patterns in large databases. One of data mining techniques is predictive modeling also known as supervised prediction or supervised learning. The most common predictive task in data mining applications besides classification is regression. Regression and model trees are now popular alternatives to classical statistical techniques like standard regression or logistic regression [10]. The tree-based approaches are gaining in popularity because of their ease of application, fast operation and effectiveness. Additionally, the hierarchical tree structure closely resembles a human way of decision making which makes regression trees natural and easy to understand even for inexperienced analysts.

Recently many regression and model tree systems have been proposed. One of the first solutions was presented in the seminal book describing the *CART* system [4]. *CART* algorithm finds a split that minimizes the sum of squared residuals of the model when predicting and builds a piecewise constant model with each terminal node fitted by the training sample mean. In the next years multiple authors improve upon the accuracy of regression trees by replacing the single predicted values in the leaves by more advanced models (e.g. linear). *M5* [20], *SECRET* [6], *SMOTI* [15] or *RT* [21] are some of the model tree algorithms that have been proposed.

All aforementioned systems induce regression and model trees in a top-down approach. Starting from the root node they search for the locally optimal split (test) according to the given optimality measure and then the training data is redirected to newly created nodes. This procedure is recursively repeated until the stopping criteria are met. Finally, the post-pruning is applied to improve the generalization power of the predictive model. Such a greedy technique is fast and generally efficient in many practical problem, but obviously does not guarantee the globally optimal solution. It can be expected that a more global induction could be more adequate in certain situations.

In this paper we want to investigate a global approach to model tree induction based on a specialized evolutionary algorithm. Our work covers the induction of univariate regression tree with simple linear models in leaves. The proposed solution may be applied to the problems that are primarily concerned with the regression of an outcome onto a single predictor. As an example the original genetic epidemiology problem required only consideration of simple linear regression models like [19] to locate genes associated with a quantitative trait of interests. There are other systems that associate leaves with simple linear regression like the one described by Alexander and Grimshaw [1] called *Treed Regression*.

Previously performed research showed that evolutionary inducers are capable to efficiently induce various types of classification trees: univariate [11], oblique [12] and mixed [13]. In our last paper we applied a similar approach to obtain accurate and compact regression trees [14]. In this work we would like to extend standard regression trees by replacing single predicted values (means) in leaves by simple linear models. Additionally, the search for an optimal structure was modified and now is driven by the Akaike's information criterion (*AIC*) [2].

The rest of the paper is organized as follows. In the next section a new evolutionary algorithm for global induction of univariate model trees is described. Experimental validation of the proposed approach on artificial and real-life data is presented in section 3. In the last section, the paper is concluded and possible future works are sketched.

2 Evolutionary Induction of Model Trees

In the proposed approach the general structure of the system follows a typical framework of evolutionary algorithms [16] with an unstructured population and a generational selection.

2.1 Representation

Model trees are represented in their actual form as classical univariate trees. Each test in a non-terminal node concerns only one attribute (nominal or continuous valued). Additionally, in every node information about learning vectors associated with the node is stored. This enables the algorithm to perform more efficiently local structure and tests modifications during applications of genetic operators.

In a case of a continuous-valued feature typical inequality tests are applied. As potential splits only precalculated candidate thresholds are considered. A candidate threshold for the given attribute is defined as a midpoint between such a successive pair of examples in the sequence sorted by the increasing value of the attribute, in which the examples are characterized by different predicted values. Such a solution significantly limits the number of possible splits and focuses the search process. For a nominal attribute at least one value is associated with each branch. It means that an inner disjunction is built into the induction algorithm.

A simple linear model is calculated at each terminal node of the model tree using standard regression technique [17]. A dependent variable Y is modeled as a linear function of single variable X:

$$Y = \beta_0 + \beta_1 * X \tag{1}$$

where X is one of the independent variables, β_0 is the intercept and β_1 is the slope of the regression line that minimizes the sum of squared residuals of the model.

2.2 Initialization

Like in *M5* approach [20] we first learn a standard regression tree (with constants in the leaves) and only afterwards we turn it into a model tree. Initial individuals are created by applying the classical top-down algorithm to randomly chosen subsamples of the original training data (10% of data, but not more than 500 examples). Additionally, for every initial tree one of three test search strategies in non-terminal nodes is applied. Two strategies come from the very well-known regression tree systems i.e. *CART* [4] and *M5* [20] and they are based on *Least Squares* or *Least Absolute Deviation.* The last strategy is called *dipolar*, where a pair of feature vectors (dipole) is selected and then a test is constructed which splits this dipole. Selection of the dipole is randomized but longer (with bigger difference between dependent variable values) dipoles are preferred and mechanism similar to the ranking linear selection [16] is applied. The recursive partitioning is finished when all training objects in a node are characterized by the same predicted value (or it vary only slightly [20]), the number of objects in a node is lower than the predefined value (default value: 5) or the maximum tree depth is reached (default value: 10).

One of two search strategies of predicted variable used in linear model at leaves is applied. First one calculates simple linear regression model for each attribute and applies the one that minimizes the sum of squared residuals of the linear regression model. In second strategy the simple linear model is built from training objects in this leaf on the randomly chosen independent variable.

2.3 Termination Condition

When the fitness of the best individual in the population does not improve during the fixed number of generations (default value is equal 1000) the evolution terminates. Additionally maximum number of generations is specified, which allows limiting the computation time in case of a slow convergence (default value: 5000).

2.4 Genetic Operators

In our previous paper [14] we have presented two specialized genetic operators corresponding to the classical mutation and cross-over. We have extended them due to simple linear model in leaves.

Application of both operators can result in changes of the tree structure, tests in non-terminal and models in terminal nodes. After applying any operator it is usually necessary to relocate learning vectors between parts of the tree rooted in the altered node. This can cause that certain parts of the tree does not contain any learning vectors and has to be pruned.

Mutation operator. A mutation-like operator is applied with a given probability to a tree (default value is 0.8) and it guarantees that at least one node of the selected individual is mutated. Firstly, the type of the node (leaf or internal node) is randomly chosen with equal probability and if a mutation of a node of this type is not possible, the other node type is chosen. A ranked list of nodes of the selected type is created and a mechanism analogous to ranking linear selection is applied to decide which node will be affected. While concerning internal nodes, the location (the level) of the node in the tree and the quality of the subtree starting in the considered node are taken into account. It is evident that modification of the test in the root node affects whole tree and has a great impact, whereas mutation of an internal node in lower parts of the tree has only a local impact. In the proposed method, nodes on higher levels of the tree are mutated with lower probability and among nodes on the same level the absolute error calculated on the learning vectors located in the subtree is used to sort them. As for leaves, only absolute error is used to put them in order, but homogenous leaves are not included. As a result, leaves which are worse in terms of accuracy are mutated with higher probability.

Modifications performed by mutation operator depend on the node type (i.e. if the considered node is a leaf node or an internal node). For a non-terminal node a few possibilities exist:

- A completely new test can be found by means of the dipolar method used for the initialization;
- The existing test can be altered by shifting the splitting threshold (continuous-valued feature) or re-grouping feature values (nominal features);
- A test can be replaced by another test or tests can be interchanged;
- One sub-tree can be replaced by another sub-tree from the same node;
- A node can be transformed (pruned) into a leaf.

After performed mutation in internal nodes the models in corresponding leaves are not recalculated for performance reasons. However, adequate linear models can be found while performing the leaves mutations. Modifying a leaf makes sense only if it contains objects with different dependent variable values. For a terminal node two possibilities exists:

– The leaf is transformed into an internal node and a new test is chosen in the aforementioned way;
– Simple linear model is replaced by other one that is calculated on different predictor variable.

Cross-over operator. In the proposed solution there are three variants of recombination. All of them start with selecting of cross-over positions in two affected individuals. One node is chosen randomly in each of two trees. In the most straightforward variant, the subtrees starting in the selected nodes are exchanged. This corresponds to the classical cross-over from genetic programming. In the second variant, which can be applied only when non-internal nodes are randomly chosen and the numbers of outcomes are equal, only tests associated with the nodes are exchanged. The third variant is also applicable only when non-internal nodes are drawn and the numbers of descendants are equal. Branches which start from the selected nodes are exchanged in random order.

2.5 Selection

As a selection mechanism the ranking linear selection is applied. Additionally, the individual with the highest value of the fitness function in the iteration is copied to the next population *(elitist strategy).*

2.6 Fitness Function

A fitness function drives the evolutionary search process and is very important and sensitive component of the algorithm. When concerning any prediction task it is well-known that the direct minimization of the prediction error measured on the learning set leads to an overfitting problem. In a typical top-down induction of decision trees, the over-specialization problem is partially mitigated by defining a stopping condition and by applying a post-pruning. In our previous work [14] the search for an optimal structure was embedded into the evolutionary algorithm by incorporating a complexity term into the fitness. This term worked as a penalty for increasing the tree size, however, there were no optimal value of it for all possible datasets.

In presented approach, we decided to use Akaike's information criterion (*AIC*) [2] as the fitness in the search for an optimal structure. This measure of the goodness of fit of an estimated statistical model works as a penalty for increasing the tree size.

The fitness function is minimized and for binary regression tree models has the following form:

$$Fitness_{AIC}(T) = -2 * ln(L(T)) + 2 * k(T) \tag{2}$$

where $L(T)$ is the maximum of the likelihood function of the tree T and $k(T)$ is the number of model parameters in the tree. Log(likelihood) function $L(T)$ is typical for regression models [9] and can be expressed as

$$ln(L(T)) = -0.5n * [ln(2\pi) + ln(SS_e(T)/n) + 1] \tag{3}$$

where $SS_e(T)$ is the sum of squared residuals of the tree T and n is the number of observations. The term, $2 * k(T)$ can also be viewed as a penalty for over-parametrization. In our approach we set $k(T) = Q(T) + 1$ in *AIC* criterion where $Q(T)$ is equal a number of terminal nodes in model tree T. Then, the fitness function is:

$$Fitness_{AIC}(T) = n(ln(2\pi) + ln(SS_e(T)/n(T)) + 1) + 2(Q(T) + 1) \tag{4}$$

In [10] authors suggested that the effective number of parameters estimated is actually much higher than $Q(T) + 1$ due to the split rule selections that were made during the T tree construction process. However, higher $k(T)$ value in *AIC* criterion leads to the smaller trees with less predictive accuracy. Further research to determine the appropriate value of complexity penalty term in the *AIC* criterion for proposed solution is required and other commonly used measures such as Bayesian information criterion (*BIC*) [18] or structural risk minimization (*SRM*) [5] should be considered.

3 Experimental Validation

Validation of the global approach to induction of model trees (denoted in tables as *GMT*) was performed on synthetical and real-life datasets. Obtained results are compared with two model trees and two regression trees. For the purpose of comparison, we have implemented the top-down regression model with simple linear regression in each leaf alike to *Treed Regression* [1] algorithm (denoted as *TR*). However, for better performance we have improved pruning to the *AIC* cost-complexity pruning proposed in [22]. We also present the results for more advanced model tree *M5* [20] proposed by Quinlan.

For real-life datasets results obtained by the classical top-down inducer *REP-Tree*, which is publicly available in the *Weka* system [8], are also presented. *REP-Tree* builds a regression tree using variance and prunes it using reduced-error pruning (with backfitting). Finally, we enclose the results of global induction of regression tree approach (denoted as *GRT*) from our previous work [14].

Each system run with default values of parameters. All results presented in the table correspond to averages of 10 runs and were obtained by using test sets (when available) or by 10-fold cross-validation. The average number of nodes is given as a complexity measure of regression and model trees.

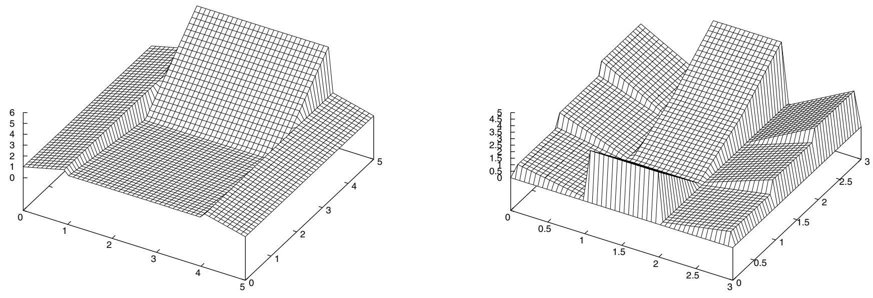


Fig. 1. Examples of artificial datasets (*armchair2* - left, *ski jump* - right)

Synthetical datasets. First group of experiments was performed on two simple artificially generated datasets with analytically defined decision borders. Both datasets contain a dependent feature that is linearly dependent with one of two independent features. One thousand observations for each dataset were divided into a training set (33.3% of observations) and testing set (66.7%).

Illustrated in figure 1 *armchair2* function is defined as:

$$g(x,y) = \begin{cases} x+1 & x \in [0,1] \\ -x-6 & x \in [4,5] \\ -0.5y+1.5 & x \in [1,4],\ y \in [0,3] \\ 3y-9 & x \in [1,4],\ y \in [3,5] \end{cases} \tag{5}$$

and the *ski jump* function:

$$g(x,y) = \begin{cases} -x+1 & x \in [0,1],\ y \in [0,1] \\ -2x+2 & x \in [0,1],\ y \in [1,2] \\ -3x+3 & x \in [0,1],\ y \in [2,3] \\ -4y+4 & x \in [1,2],\ y \in [0,1] \\ 2y-2 & x \in [1,2],\ y \in [1,2] \\ 3y-4 & x \in [1,2],\ y \in [2,3] \\ x-2 & x \in [2,3],\ y \in [0,1] \\ 2x-4 & x \in [2,3],\ y \in [1,2] \\ 3x-6 & x \in [2,3],\ y \in [2,3] \end{cases} \tag{6}$$

Table 1. Results obtained on the synthetical datasets *armchair2* and *ski jump*. Root mean squared error (RMSE) is given as the error measure and number of nodes as the tree size.

	GMT		*M5*		*TR*	
Dataset	RMSE	Tree size	RMSE	Tree size	RMSE	Tree size
armchair2	0.12	6.5	0.37	11.0	0.35	24.0
ski jump	0.30	10.9	0.61	26.0	0.54	25.0

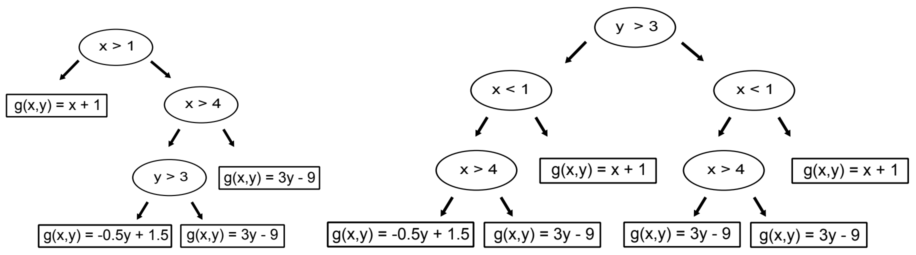


Fig. 2. Examples of model trees for *armchair2* (*GMT* - left, *TR* - right) from the experiment

Table 1 presents obtained results only for model trees as the regression trees on those synthetic datasets are not competitive due to the training sample mean in terminal nodes.

It should be noticed that in both cases trees obtained by *GMT* have optimal or almost optimal structure and gain very small error. For the top-down inducers both problems were too difficult. *M5* and *TR* generated overgrown trees and as a result the testing error is higher. Additionally, the reason why *M5* model tree performed lower than *TR* is that the *M5* tried to use both independent features in linear model at leaves.

The advantage of the global approach can be observed in the in figure 2 where the optimal model trees for *GMT* and *TR* on the first dataset *armchair2* are illustrated. We can see that the first split which minimizes the sum of squared residuals ($y < 3$) is not optimal as it leads to overgrown tree.

Real-life datasets. In the second series of experiments, several datasets taken from UCI Machine Learning Repository [3] or provided by L. Torgo on his website are analyzed to assess the performance of the proposed system in solving real-life problems. Table 2 presents characteristics of investigated datasets and obtained results. More complex model trees like *M5* or *SMOTI* were not included in these experiments as in this paper we are focussing on comparing the improvement of global induction for regression trees and model trees that associate leaves with simple linear regression. We plan to extend evolving model trees with multivariate linear regression so it will be possible to compare with more advanced modeling trees.

It can be observed that the prediction accuracy of model trees that associate leaves with simple linear regression models is comparable to regression trees that have training sample mean in terminal nodes. It is not surprising that size of the trees are smaller in favor to model trees however, it should be noticed that globally induced trees are less complex. Proposed solution *GMT* performed better on 8 out of 10 datasets comparing to *TR* and 9 out of 10 comparing to *REPTree* in term of accuracy. Tree size for *GMT* was lower on almost all datasets.

Table 2. Characteristics of the real-life datasets (number of objects/number of numeric features/number of nominal features) and obtained results. Root mean squared error (RMSE) is given as the error measure and number of nodes as the tree size.

		GMT		*TR*		*GRT*		*REPTree*	
Dataset	Properties	RMSE	size	RMSE	size	RMSE	size	RMSE	size
Abalone	4177/7/1	2.297	7.7	2.636	10.9	2.314	51.8	2.358	201
Auto-Mpg	392/4/3	3.434	9.9	3.670	73.5	3.572	45.4	3.646	94
Auto-Price	159/17/10	2507.6	3.7	2433.9	9.9	2618.9	13.8	2760.5	32
Delta Ailerons	7129/6/0	0.000178	11.1	0.000185	7.2	0.000179	82.6	0.000175	291
Delta Elevators	9517/6/0	0.00150	9.3	0.00157	16.2	0.00148	78.3	0.00150	319
Housing	506/14/0	4.322	9.1	4.495	11.8	4.126	32.3	4.84	41
Machine CPU	209/7/0	67.53	3.8	78.18	11.3	63.99	14.8	92.34	15
Pyrimidines	74/28/0	0.1090	4.52	0.0987	5.8	0.1011	10.7	0.1355	1.0
Triazines	186/61/0	0.1405	4.7	0.1564	3.9	0.1387	13.7	0.1517	7.0
Wisconsin Cancer	194/32/0	34.33	3.1	35.13	1.9	39.22	16.3	35.88	9.0

4 Conclusion

In the paper a new global approach to model tree learning is presented. In contrast to classical top-down inducers, where locally optimal tests are sequentially chosen, both the tree structure, tests in internal nodes and models in leaves are searched in the same time by specialized evolutionary algorithm. This way the inducer is able to avoid local optima and to generate better predictive model. Even preliminary experimental results show that the globally evolved regression models could be competitive compared to the top-down based counterparts, especially in term of tree size.

The presented approach is constantly improved and currently we are working on introducing oblique tests in the non-terminal nodes. On the other hand, we plan to extend the knowledge representation by evolving model trees with multivariate linear regression. However, the proposed solution may be applied to the problems that are primarily concerned with the regression of an outcome onto a single predictor.

Acknowledgments. This work was supported by the grant W/WI/3/2010 from Bialystok University of Technology.

References

1. Alexander, W.P., Grimshaw, S.D.: Treed Regression. Journal of Computational and Graphical Statistics 5, 156–175 (1996)
2. Akaike, H.: A New Look at Statistical Model Identification. IEEE Transactions on Automatic Control 19, 716–723 (1974)
3. Blake, C., Keogh, E., Merz, C.: UCI Repository of Machine Learning Databases (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`

4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth Int. Group (1984)
5. Cherkassky, V., Mulier, F.: Learning from Data: Concepts, Theory and Methods. Wiley, New York (1998)
6. Dobra, A., Gehrke, J.: SECRET: A Scalable Linear Regression Tree Algorithm. In: Proc. KDD 2002 (2002)
7. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park (1996)
8. Frank, E., et al.: Weka 3 - Data Mining with Open Source Machine Learning Software in Java. University of Waikato (2000), http://www.cs.waikato.ac.nz/~ml/weka
9. Gagne, P., Dayton, C.M.: Best Regression Model Using Information Criteria. Journal of Modern Applied Statistical Methods 1, 479–488 (2002)
10. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. In: Data Mining, Inference, and Prediction, 2nd edn. Springer, Heidelberg (2009)
11. Krętowski, M., Grześ, M.: Global Learning of Decision Trees by an Evolutionary Algorithm. In: Information Processing and Security Systems, pp. 401–410. Springer, Heidelberg (2005)
12. Krętowski, M., Grześ, M.: Evolutionary Learning of Linear Trees with Embedded Feature Selection. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 400–409. Springer, Heidelberg (2006)
13. Krętowski, M., Grześ, M.: Evolutionary Induction of Mixed Decision Trees. International Journal of Data Warehousing and Mining 3(4), 68–82 (2007)
14. Krętowski, M., Czajkowski, M.: An Evolutionary Algorithm for Global Induction of Regression Trees. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010. LNCS (LNAI), vol. 6114, pp. 157–164. Springer, Heidelberg (2010)
15. Malerba, D., Esposito, F., Ceci, M., Appice, A.: Top-down Induction of Model Trees with Regression and Splitting Nodes. IEEE Trans. on PAMI 26(5), 612–625 (2004)
16. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
17. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes in C. Cambridge University Press, Cambridge (1988)
18. Schwarz, G.: Estimating the Dimension of a Model. The Annals of Statistics 6, 461–464 (1978)
19. Shannon, W.D., Province, M.A., Rao, D.C.: Tree-Based Models for Fiting Stratified Linear Regression Models. Journal of Classification 19, 113–130 (2002)
20. Quinlan, J.: Learning with Continuous Classes. In: Proc. AI 1992, pp. 343–348. World Scientific, Singapore (1992)
21. Torgo, L.: Inductive Learning of Tree-based Regression Models. Ph.D. Thesis, University of Porto (1999)
22. Xiaogang, S., Morgan, W., Juanjuan, F.: Maximum Likelihood Regression Trees. Journal of Computational and Graphical Statistics 13(3), 586–598 (2004)

Open-Ended Evolutionary Robotics: An Information Theoretic Approach

Pierre Delarboulas*, Marc Schoenauer, and Michèle Sebag

Project-team TAO – LRI, UMR CNRS 8623 & INRIA Saclay
Université Paris-Sud, F-91405 Orsay

Abstract. This paper is concerned with designing self-driven fitness functions for Embedded Evolutionary Robotics. The proposed approach considers the entropy of the sensori-motor stream generated by the robot controller. This entropy is computed using unsupervised learning; its maximization, achieved by an on-board evolutionary algorithm, implements a "curiosity instinct", favouring controllers visiting many diverse sensori-motor states (sms). Further, the set of sms discovered by an individual can be transmitted to its offspring, making a cultural evolution mode possible. Cumulative entropy (computed from ancestors and current individual visits to the sms) defines another self-driven fitness; its optimization implements a "discovery instinct", as it favours controllers visiting new or rare sensori-motor states. Empirical results on the benchmark problems proposed by Lehman and Stanley (2008) comparatively demonstrate the merits of the approach.

1 Introduction

Evolutionary Robotics (ER) aims at designing robust autonomous robots, and in particular robust robot controllers [15]. The success of ER critically depends on the optimization objective (or fitness function) [14]. For the sake of computational and experimental convenience, the ER literature mostly considers simulation-based approaches, where the controller fitness is computed by simulating the robot behaviour. The price to pay for this convenience is that the controller behaviour suffers from the so-called reality gap, i.e. its performance might dramatically decrease when ported on-board [12]. While Embedded Evolutionary Robotics [2] sidesteps the reality gap as evolutionary computation is achieved on-board, the challenge is to design some fitness either based on environmental cues (e.g. about the target location), or not requiring any ground truth at all, i.e., self-driven fitness.

Developmental Robotics [16] also aims at principled ways of building intelligent agents. The vision, strongly inspired from Brooks' [3], states that (i) the world is its best model and the robot representations must be grounded in its physical perceptions; (ii) robots must be "autonomous, self-sufficient, embodied, and situated" (Complete Agent principle). The search for self-driven objectives,

* Funded by EU FP7, FET IP SYMBRION No. 216342, http://symbrion.eu/

leading the robot to explore its world and gradually learn new skills, thus is at the core of Developmental Robotics [1].

The present work, at the crossroad of Developmental and Embedded ER, focuses on self-driven fitness functions. An original approach, amenable to on-board evolutionary optimization and rooted in Information Theory, is presented as an Intrinsic Motivation Systems [1]. This approach defines a "curiosity" instinct, which enforces the exploration by the robot of its environment. Both approaches exploit the robotic log recording for each time step the sensor and motor values, or Sensori-Motor Stream (SMS). The difference is twofold. On the one hand, the presented approach relies on (computationally frugal) unsupervised learning [6] whereas [1] uses supervised ML to build a forward model. On the other hand, it defines a representation of the robot world: a set of sensori-motor states (sms), built from the SMS, supports the computation of the SMS entropy; the higher the entropy, the richer and the more diversified the world seen by the current controller. Maximizing the SMS entropy thus defines an efficient, self-driven, "curiosity instinct" enforcing the exploration of the world.

The second contribution of the paper is another self-driven fitness function dubbed "discovery instinct". It exploits the fact that sensori-motor states (sms) can be transmitted from parents to offspring, thus enabling some cultural evolution mode [5,8]. Formally, the discovery fitness computes the cumulative entropy defined from the visits of all robots to all sms, until the current individual; it thus rewards individuals that discover sensori-motor states *which have not yet been visited* (or rarely visited) by its ancestors. Discovery fitness might be thought of as a fitness sharing mechanism, except for the fact that it rewards individuals which differ from their ancestors, as opposed to, their peers.

The merits of the two self-driven fitness functions, implemented within an embedded (1+1)-Evolution Strategy [2], are comparatively assessed on the benchmark problems defined by Lehman and Stanley [11], in terms of their patrolling activity (visiting the various places of the arena, and visiting the chambers farthest apart from the starting point). The main limitation of the approach is to require a stimulating interaction between the environment and the robot sensors, conducive to diversified sensory experiences: if located in the middle of nowhere, or endowed with too poor sensors, the robot can only experience a few sensori-motor states and entropy provides no incentive for exploration.

The paper is organized as follows. Related work is briefly reviewed and discussed in section 2. An overview of the curiosity and discovery fitnesses is presented in section 3 and section 4 reports on the experimental results. The paper concludes with a discussion and some perspectives for further research.

2 Related Work

The state of the art in ER and fitness design can be structured in different ways [17,14], depending on the designer's criteria. The perspective presented in [14] focuses on the amount of human effort and prior knowledge required to overcome bootstrap problems. Such problems can be illustrated from the hard

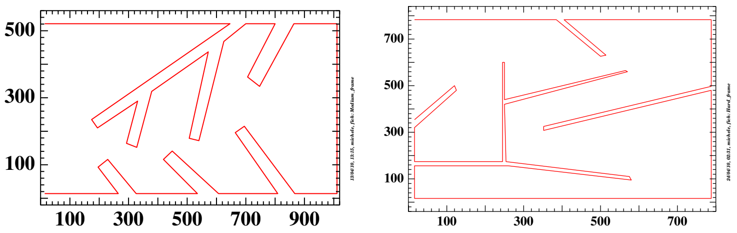


Fig. 1. From (Lehman & Stanley, 08): medium (left) and hard (right) arenas. Starting in the upper left corner, the goal is to reach the farthest apart region.

and medium arenas due to [11] (Fig. 1), where the goal is to reach the chamber farthest apart from the starting point without any ground truth being available; note that providing the robot with its bird eye's distance to the target location will get it trapped in many local optima.

Among the various heuristics investigated to address the bootstrap problem is the staged fitness approach pioneered by [9] (e.g. learning to walk before learning to run). Among the early and still widely used[1] approaches to bootstrap avoidance are fitness-sharing and diversity enforcing [11,7].

Diversity enforcing heuristics in ER mostly rely on the genotypic [10] or phenotypic [7,11] distances between the robot controllers. Some genotypic distances have been defined on structured controller spaces such as neural nets [10]. Phenotypic distances usually rely on prior knowledge. For instance Lehman and Stanley associate to a controller the end point of the robot trajectory; the novelty of a controller w.r.t. the population is then assessed from the average distance of its end point to that of its k-nearest neighbours [11]. A general phenotypic distance between robot trajectories has been proposed by Gomez, relying on the compression-based distance inspired from Kolmogorov complexity [7]. Setting the fitness function to the controller phenotypic diversity leads evolution to construct a fair sample of the robot behavioural space. In this sample, the designer eventually selects the individual best fitting the task at hand, which reportedly gives satisfying solutions. Another way of enforcing diversity is based on multi-objective evolutionary optimization, considering diversity as an additional objective besides the actual robotic objective [13].

Another influential line of research is Embodied Statistical Learning (ESL) [16], aimed at a principled way of addressing the reality gap issue through perceptual and behavioural learning. While perceptual learning is concerned with building a grounded representation of the environment and the robot itself, behavioural learning aims at achieving the target tasks. Regarding perceptual learning, ESL advocates the use of the "information self-structuring" principle, stating that the robot should take advantage of statistical regularities induced

[1] Other possibilities, outside the scope of this paper, are to inject competent individuals in the initial population, or to use co-evolution.

by its interactions with the world, and mentions "adaptive compression" as a possible approach [4].

3 Information Theory-Based Robotic Instincts

This section presents two self-driven fitness functions, referred to as Curiosity and Discovery instincts, simultaneously enforcing the exploration of the environment and the behavioural diversity of the controllers. These functions, also rooted in the information self-structuring principle, are based on unsupervised statistical learning.

3.1 Unsupervised Learning from the Sensori-Motor Stream

Some information the robot gets for free lies in its sensor and motor values. Let the sequence of sensori-motor value vectors along time, referred to as Sensori-Motor Stream (SMS), be denoted $\mathcal{X} = \{x_t; x_t \in \mathbb{R}^d, t = 1, \ldots, T\}$, where vector x_t stores the d values of the sensors plus motors at time t, and T is the number of time steps of the robot lifetime. The SMS $\mathcal{X}$ of course depends on the controller of the robot when it is gathered. Our claim is that *interesting controllers result in a high sensori-motor diversity* (subject to requirements discussed in section 3.3). After Information Theory principles, it thus comes naturally to assess the controller from the entropy of the SMS. Noting $(c_i)_{i=1,\ldots,p}$ the p states (sensori-motor vectors) visited by the robot and n_i the number of times c_i has been visited, it comes:

$$\mathcal{F}(\mathcal{X}) = -\sum_{i=1}^{p} \frac{n_i}{\sum_{j=1}^{p} n_j} \log \frac{n_i}{\sum_{j=1}^{p} n_j} \tag{1}$$

Sensori-motor states should however achieve some abstraction or generalization relatively to the sensori-motor vectors. Otherwise, since x_t is a real-valued vector in a possibly high dimension space, for most visited states $n_i = 1$ and most trajectories over T time steps get the same trivial fitness value $\log(T)$. Unsupervised learning (clustering), is applied to the robotic log to form clusters using Euclidean distance. While many clustering algorithms have been designed in the literature, only the simple, computationally linear k-means and ϵ-means will be considered in the rest of this paper (Fig. 2, left); the interested reader is referred to [6] for a more comprehensive introduction. The k-means algorithm is parametrized from the number k of clusters while ϵ-means is parametrized from the maximal radius ϵ of each cluster.

3.2 Curiosity and Discovery Instincts

Let X be a controller. The rest of the paper does not depend on the actual representation of X (its genotype), and only considers its phenotype $\mathcal{X} = \{x_1 \ldots x_T\}$, defined as its SMS in the environment. Stream $\mathcal{X}$ is processed using k-means or ϵ-means (Fig. 2), yielding the set of p sensori-motor states c_i together with their

```
k-means Algorithm
$\mathcal{C} = c_1 \dots c_k$ random training points
repeat
  for t = 1...T,
    $i(t) = argmin_{j=1\dots k}\{d(x_t, c_j)\}$
  for i = 1 ... k
    $c_i = \frac{\sum_{t/i(t)=i} x_t}{\sum_{t/i(t)=i} 1}$
until $\mathcal{C}$ does not change
```

```
ϵ-means Algorithm
$\mathcal{C} = \emptyset$
for t = 1..T
  $i(t) = argmin_{c_j \in \mathcal{C}}\{d(x_t, c_j)\}$
  if $(d(x_t, c_i) > \epsilon)$    $\mathcal{C} \leftarrow \mathcal{C} \bigcup (x_t, 1)$
  else $n_i + +$
```

Fig. 2. k-means and ϵ-means clustering algorithms.

number n_i of occurrences in the stream. The curiosity fitness $\mathcal{F}_c$ is defined as the entropy of the trajectory (Eq. (1)). An individual gets a high curiosity fitness if it equally shares its time among the visited sms (k-means clustering, $p = k$), or it visits many sms (ϵ-means clustering). The maximization of the curiosity fitness is achieved using a $(1 + 1)$-Evolution Strategy with random restart (Section 4).

As will be discussed in Section 3.3, the number p of sensori-motor states must be kept below a few hundreds for the sake of efficiency. This makes it feasible to store the corresponding set of sms on-board, and to transmit it from the parent to the offspring. Another self-driven fitness function dubbed "discovery instinct" can thus be defined. Informally, the idea is that the offspring will be rewarded for visiting sensori-motor states *which have not been visited* (or rarely visited) by its ancestors.

Formally, the discovery fitness $\mathcal{F}_d$ is defined along the same equation as the curiosity fitness. The only difference is that the ϵ-means algorithm uses the set of sms c_i visited by ancestors, where n_i stands for the total number of visits paid to state c_i along the generations, to initialize $\mathcal{C}$; $\mathcal{C}$ is updated from the current trajectory, incrementing the counters of visited states and possibly adding new sensori-motor states discovered by the current individual. Ultimately, the discovery fitness $\mathcal{F}_d$ is computed from the entropy of $\mathcal{C}$ (Eq. (1)).

The discovery fitness thus implements a dynamic evolution schedule: the worth of any given behaviour depends on its novelty, pushing evolution toward the collective exploration of the sensori-motor space. Typically, an individual controller will get a good discovery fitness iff it discovers new sms, or if it visits sufficiently many rare sms, where novelty and rarity are measured from the current robot-kind experience. As already noted, discovery can thus be viewed in terms of fitness sharing, as the worth of visiting a sensori-motor state depends on how many individuals visited it. The difference compared with standard fitness sharing is that sharing usually takes place among all individuals in a same generation, whereas discovery considers all generations up to the current one[2].

[2] Discovery fitness can thus also be thought of in terms of Cultural Evolution. However, the knowledge gained in the previous generations only modifies the individual assessment in the proposed scheme, contrasting with modifying the individual behaviour in standard Cultural Evolution (see e.g. [5,8]).

3.3 Discussion

The main limitation of the proposed approach is as follows. The entropy of the robot trajectory depends on the richness of both the environment and the robot sensors. If the robot is in the middle of an empty area, there is nothing to be curious about and nothing to be discovered; whatever its sensors, the robot will experiment a single state: nothing in sight[3]. Likewise, if the robot is endowed with a single boolean touch sensor, whatever the richness of the environment the robot can only experience two states: I can touch something, or I can't. The presented entropy-based approach thus only makes sense if the environment mediated by the robot sensors offers sufficient stimulation.

Another critical aspect is the calibration of the clustering algorithm (parameters k or ϵ, Fig. 2). At one extreme (too fine-grained) all sensori-motor vectors belong to different clusters; at the other extreme (too coarse), all belong to the same cluster; in both cases, the entropy is trivial and does not provide any indication to evolution. Along the same lines, rich robotic sensors (e.g. a camera) could hinder the approach due to the curse of dimensionality, and the fact that Euclidean distance in $\mathbb{R}^D$ is not much informative for high D values. A preliminary dimensionality reduction step, mapping $\mathbb{R}^D$ onto $\mathbb{R}^d, d << D$, would thus be required, and could be obtained by careful sensor fusion. It must however be noted that, provided that the dimensionality reduction can be done online, its calibration (e.g. using Principal Component Analysis or non-linear approaches) can be optimized off-line; the approach thus remains tractable in the context of embedded evolution.

The Curiosity fitness can possibly reward some degenerate behaviours, like dancing in front of a wall or in a corner; more generally, a periodic trajectory in a stimulating environment would get a high Curiosity fitness. The Discovery fitness is less prone to degenerate behaviours since it essentially rewards new behaviours.

Conditionally to a stimulating environment and a reasonably calibrated clustering algorithm (in practice, a few hundred sensori-motor states), the Curiosity and Discovery fitnesses display interesting properties. First of all, they meet the on-board evolution requirements (bounded computational and memory resources, no ground truth required). Secondly, they are robust w.r.t sensor and motor noises; introducing outliers in the SMS would result in creating sms that are very rarely visited, with little impact on the entropy. Thirdly, these fitnesses penalize inaction and favour the exploration of the sensori-motor space (not moving leads the robot to experience a single sensori-motor state).

4 Experimental Validation

The main two questions investigated in this section are (i) whether the curiosity and discovery fitnesses are actually compatible with on-board evolution, overcoming the reality gap issue; and (ii) whether they are conducive to the discovery

[3] The implicit assumption done in the paper being that the controller is deterministic.

of "interesting" behaviours, measured from the exploration of sufficiently complex arenas. Other questions of interest, regarding the sensitivity of the approach w.r.t. the clustering parameters, will not be addressed here due to space limitations: only the ϵ-clustering, with $\epsilon = .2$ for Curiosity and $\epsilon = .4$ for Discovery, will be presented below.

The experimental setting is based on the home-made Roborobo 2D simulator, simulating a Cortex-M3 with eight infra-red sensors and two motors: Following [2], the robot supports an on-board (1+1)-Evolution Strategy using the $1/5^{th}$ rule, with restart after 30 fitness evaluations with no improvement; each run stops after 2,000 fitness evaluations. The controller space is that of multi-layer perceptrons with 8 inputs, 2 outputs, and 10 hidden neurons. The 112 weights are initially randomly drawn following a normal distribution with mean 0 and variance 0.1. The isotropic Gaussian mutation has an initial step-size of 0.2. In both settings, the sensori-motor stream is clustered online using ϵ-means, with $\mathcal{C}$ initialized to the empty set for Curiosity, and, for Discovery, to the inherited set, easily stored on the Cortex board. The entropy of $\mathcal{C}$ is computed after $T = 2000$ time steps.

The robot environment is set to one of the arenas defined in [11] (Fig. 1). The performances of Curiosity and Discovery fitnesses are compared, with same experimental setting, to both the baseline fitness for displacement with obstacle avoidance originally proposed by Floreano and Mondada (and described, with references, in [15]) (legend *Displacement*), and the Novelty fitness proposed by Lehman and Stanley [11] (see Section 2, legend *Novelty*). The performance indicators measure the patrolling ability, that is the percentage of $p(\ell)$ of squares in the arena that have been visited at least ℓ times. Another performance indicator is whether the robot can explore the chambers farthest apart (avoiding obstacles) from the starting point. All robots start from the same point, in order to reliably assess the robustness of the algorithm w.r.t. the distance to the starting point. All results are averaged over 11 independent runs.

Table 1. Patrolling performances for 2000 time steps: average (std. dev.) over 11 runs

	Medium Arena			Hard Arena		
	2 visits	5 visits	10 visits	2 visits	5 visits	10 visits
100 best individuals in 2000 generations run						
Curiosity	35.78(9.04)	22.01(7.39)	13.0(4.47)	50.18(6.7)	30.31(5.79)	14.79(3.16)
Discovery	21.99(9.24)	12.82(5.85)	8.12(3.15)	16.28(6.27)	10.11(3.19)	7.26(1.83)
Displacement	25.78(1.77)	22.3(1.96)	18.12(1.86)	44.9(10.51)	28.22(6.2)	18.99(3.79)
Novelty	53.99(2.75)	36.32(2.26)	21.03(1.61)	55.35(4.66)	35.87(3.35)	19.78(2.0)
All individuals in 2000 generations run						
Curiosity	69.67(2.62)	61.56(2.87)	54.95(3.36)	78.29(5.12)	68.32(5.58)	58.09(4.47)
Discovery	62.08(3.47)	53.0(4.59)	45.54(5.81)	65.23(4.98)	52.4(5.59)	42.27(4.97)
Displacement	72.36(2.23)	61.06(3.59)	50.92(1.8)	78.48(3.93)	64.98(5.24)	52.63(3.68)
Novelty	64.82(1.81)	55.68(1.76)	49.23(1.67)	66.37(5.08)	55.31(4.53)	47.24(3.75)

The average patrolling abilities for both arenas and the 4 algorithms are reported in Table 1: for the 100 best individuals that were evolved during the 11 runs (top), and all the individuals that appeared during those runs (bottom), Table 1 gives the percentage of the arena that has been visited at least 2, 5 or 10 times. Fig. 3 displays a typical case: the points that have been visited at least 10 times in the Hard arena setting (the Medium arena shows similar trends).

When considering the 100 best individuals that appeared during the 11 runs, Novelty is clearly outperforming the other approaches in terms of patrolling the Medium arena, while Curiosity (and, to a lesser extend, Displacement) catch up in the co-called Hard arena. Indeed, and almost paradoxically, the Hard arena offers in fact a high sensor diversity, exhibiting zones that look rather different, while the Medium arena somehow repeats the same motif several times, generating less diverse sensor input combinations. Furthermore, looking at the plots of Fig. 3, the Novelty runs tends to also explore the interior regions of the maze, while the Curiosity runs stay along the walls: the empty zones always generates the same sensor values, and hence cannot contribute to increase the entropy. However, as can be seen on Fig. 3, and is confirmed by looking at the maximum distance from the starting position reached by the individuals of different generations (results not shown here), far chambers from the starting points are more densely visited by the Curiosity runs than by the Novelty ones (the first wall from bottom up is more clearly marked in the Curiosity plot).
Finally, the Discovery fitness performs poorly when considering the 100 best individuals. Indeed, those individuals essentially correspond to the 100 last individuals, visiting sensori-motor states which have not been visited by the ancestors, hence basically exploring only the corners of the arena.

When considering all individuals ever produced by evolution during the 11 runs (bottoms half of Table 1 and bottom row of Fig. 3), the picture changes dramatically. The best performing fitness now is Curiosity, and again this is even clearer on the Hard arena. The Displacement fitness now also reaches better performances than Novelty, again more clearly on the Hard arena. Finally, the performance of Discovery is almost as good as the other ones, and the plot of the

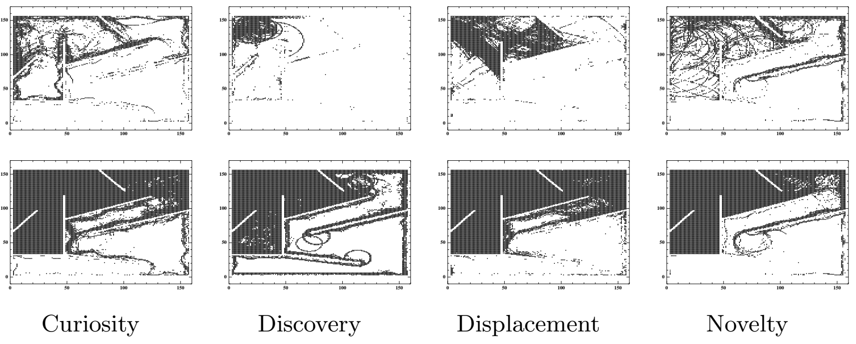


Fig. 3. Hard Arena: points visited 10 times or more by the 100 best individuals (top) or the whole population (bottom) over the 11 runs

10-times visits is even denser toward the far end of the arena with respect to the starting point, i.e., close to the bottom wall: because of the inherited information from parents to offspring, Discovery should indeed only be assessed by looking at complete lineages.

The success of Curiosity compared to Novelty is somewhat unexpected, as Novelty actually relies on a significant amount of prior knowledge, requiring for instance the robot to always know its position, and, from the archive, where all other robots ended up their trajectories. On the opposite, the Curiosity fitness is built up from scratch by each robot – or by the lineage of robot in the case of the Discovery fitness.

5 Discussion and Perspectives

The paper pioneers the use of statistical unsupervised learning to define self-driven fitness functions for on-board ER. Information Theory is then used to push the robot toward unknown parts of the sensori-motor space, hopefully leading to interesting behaviors in the physical space.

The priority here is to enable the robot to merely discover its sensori-motor space, as opposed to maximize the predictive information in the sensori-motor loop (as in [18] and references therein). The rationale for this priority is twofold. Firstly, the presented approach is extremely frugal computationally speaking, compared to e.g. [18], as the target application here concerns swarm robotics. Secondly, recent trends in Machine Learning suggest that changing the representation of the problem domain, as done through unsupervised learning, can dramatically facilitate further supervised learning tasks.

Compared to the standard Evolutionary Robotics framework pioneered by Floreano and Mondada, the robot is rewarded here for what it gets (a rich sensori-motor experience) and not for what it does (going fast and circling infrequently). As mentioned earlier on, such a fitness function is only efficient in "interesting environments"; under-stimulation results in a evolutionary bootstrap problem. Interestingly, over-stimulation is also detrimental to the efficiency of the curiosity and discovery instincts.

Compared to the Novelty approach by Lehman and Stanley [11], no external information is needed here to assess the novelty of a behaviour. Furthermore, almost independently of the context, and the goal (though Novelty search can be totally goal-less, it can also be constrained toward a loosely defined goal), the computational cost of the proposed approaches remains tractable, as only sensori-motor states need to be stored.

Compared to Embodied Statistical Learning and Intrinsic Motivation [1], beside being computationally light learning algorithms amenable to on-board evolution, the proposed approaches rely on the discovery of sensori-motor states, amenable to the direct inspection of the designer. Typically, relating visited sms to some instants of the trajectory would lead to interpreting them, thus allowing the designer to enforce some preferences, e.g., a safety policy in critical situations.

Further study will of course concern experiments in richer behavioral spaces, and also the collective and dynamic adjustment of the clustering granularity ϵ, depending on the current context. Another perspective is related to coupling self-driven fitnesses with interactive optimization, asking the designer's preferences among the available behaviours.

References

1. Baranes, A., Oudeyer, P.-Y.: R-IAC: Robust intrinsically motivated exploration and active learning. IEEE Transactions on Autonomous Mental Development 1(3), 155–169 (2009)
2. Bredeche, N., Haasdijk, E., Eiben, A.E.: On-line, on-board evolution of robot controllers. In: Collet, P., Legrand, P. (eds.) EA 2009. LNCS, vol. 5975, pp. 110–121. Springer, Heidelberg (2009)
3. Brooks, R.A.: Intelligence without reason. In: IJCAI 1991, pp. 569–595. Morgan Kaufmann, San Francisco (1991)
4. Burfoot, D., Lungarella, M., Kuniyoshi, Y.: Toward a theory of embodied statistical learning. In: Asada, M., Hallam, J.C.T., Meyer, J.-A., Tani, J. (eds.) SAB 2008. LNCS (LNAI), vol. 5040, pp. 270–279. Springer, Heidelberg (2008)
5. Curran, D., O'Riordan, C.: Cultural learning in a dynamic environment: an analysis of both fitness and diversity in populations of neural network agents. Journal of Artificial Societies and Social Simulation 10(4), 3 (2007)
6. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, Chichester (2001)
7. Gomez, F.J.: Sustaining diversity using behavioral information distance. In: GECCO 2009, pp. 113–120. ACM, New York (2009)
8. Haasdijk, E., Vogt, P., Eiben, A.E.: Social learning in population-based adaptive systems. In: CEC 2008, pp. 1386–1392. IEEE Press, Los Alamitos (2008)
9. Harvey, I., Husbands, P., Cliff, D.: Artificial evolution; real vision. In: SAB 1994, pp. 392–401. MIT Press, Cambridge (1994)
10. Miikkulainen, R., Stanley, K.O., Bryant, B.D.: Evolving adaptive neural networks with and without adaptive synapses. Evolutionary Computation 4, 2557–2564 (2003)
11. Lehman, J., Stanley, K.O.: Exploiting Open-endedness to solve problems through the search for novelty. In: AILife 2008. MIT Press, Cambridge (2008)
12. Lipson, H., Bongard, J.C., Zykov, V., Malone, E.: Evolutionary robotics for legged machines: From simulation to physical reality. In: IAS, pp. 11–18. IOS Press, Amsterdam (2006)
13. Mouret, J.-B., Doncieux, S.: Using behavioral exploration objectives to solve deceptive problems in neuro-evolution. In: GECCO, pp. 627–634. ACM, New York (2009)
14. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. Robot. Auton. Syst. 57(4), 345–370 (2009)
15. Nolfi, S., Floreano, D.: Evolutionary Robotics. MIT Press, Cambridge (2000)
16. Pfeifer, R., Iida, F., Bongard, J.: New robotics: Design principles for intelligent systems. Artificial Life 11(1-2), 99–120 (2005)
17. Watson, R.A., Ficici, S.G., Pollack, J.B.: Embodied evolution: Distributing an evolutionary algorithm in a population of robots. Robotics and Autonomous Systems 39(1), 1–18 (2002)
18. Zahedi, K., Ay, N., Der., R.: Higher coordination with less control. A result of information maximisation in the sensori-motor loop. Adaptive Behavior (to appear)

A Novel Similarity-Based Crossover for Artificial Neural Network Evolution

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

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

Abstract. This work presents an evolutionary approach for the optimization of neural networks design, based on the joint evolution of the topology and the connection weights, providing a novel similarity-based crossover that aims to overcome one of the major problems of this operator, known as the permutation problem.

The approach has been implemented and applied to two benchmark classification problems in machine learning, and the experimental results, compared to those obtained by other works in the literature, show how it can produce compact neural networks with a satisfactory generalization capability.

1 Introduction

Evolutionary Computation (EC) is the universally accepted term to describe the field of study on computational systems that draw their inspiration from the processes of natural evolution and adaptation. Their advantages over conventional methods [4], like their conceptual and computational simplicity and their applicability to broad classes of problems or self-optimization, make them very suitable for problems with dynamically changing environment and multi-objective optimization requirements.

Evolutionary algorithms have been applied to different problems, including artificial neural network (ANN) design, the so-called Evolutionary Artificial Neural Networks (EANNs) [17]. The success of an ANN application usually requires a high number of experiments. Moreover, several parameters of an ANN can affect, during the design, how easy a solution is to find. Among them, particular attention has been given to those related to the architecture design of the neural network. They correspond to a well-known topic of interest in the literature, since an inadequate network could be unable to learn or will overfit the training data [7]. In this area, the optimization of artificial neural networks (ANNs) with evolutionary algorithms (EAs) has come of age and is now a well-established discipline. EAs are able to overcome an important limitation of traditional neural network learning, namely that they are able to get out from local minima if they get trapped there.

Among the genetic operators that can be applied during the evolutionary process, some authors regard crossover as inefficient, due to the so-called permutation (or competing convention) problem [9]. This problem occurs because the same network can be genetically represented by many and different encodings. This problem is also indicated as a many-to-one mapping from the representation of the solutions (the genotype) to the actual ANNs (the phenotype) [17].

Nevertheless, there are many successful applications of EANNs using crossover: Hancock, for example, conducted studies on structural optimization [9], Garcia-Pedrajas and colleagues have recently investigated a combination of structure and weight evolution [6], and it is worth emphasizing that none of them has found any significant detrimental effects attributable to the permutation problem. Accordingly, there is a need to re-evaluate the traditional theoretical claims with regard to this problem.

In this paper, a novel similarity-based crossover operator is presented, as a new feature of a previous neuro-genetic approach for neural network design [2], based on the conjunction of topology and connection weight optimization.

The paper is organized as follows: Section 2 introduces the features and the main critical aspects of the crossover. The particular neuro-genetic approach considered for demonstrating the proposed crossover operator is then briefly summarized in Section 3, while a more detailed description of the crossover implemented in this work is explained in Section 4. All the experiments are then presented, compared, and discussed in Section 5. Section 6 reports some final remarks.

2 Evolution of Artificial Neural Network Designs

Among the many applications presented in the literature in the area of EANNs, different approaches show interesting combinations of network architecture and weight optimization, carried out simultaneously.

In fact, the choice of an ANN structure has a considerable impact on the processing power and learning capability of the classifier. In ANN evolution, when the recombination process is considered, the crossover operator exchanges the architectures of individuals in the population, identified as parents, in order to search better solutions. In this sense, the *permutation problem* holds that crossover can be seriously disruptive when applied, for examples, to ANNs whose genotypes are incompatible even though their phenotypes might be indistinguishable by the fitness function. One example of the well-known convergence problem is indicated in the literature and it has been cited as a reason for using purely mutation-based approaches [17].

The claim that a standard crossover application, in networks with the many-to-one mapping problem, may produce unfit offsprings, has been identified in the literature as a possible consequence of a disregard of the generally converged nature of standard population-based search [5]. Therefore, the so-called *convergence argument* [5] agrees with some critical aspects reported by some works with respect to crossover, indicating that it is generally very difficult to apply since it tends to destroy feature detectors found by the global evolutionary process while searching for the best individual in the population of networks.

In contrast, it is important to notice that crossover is usually not harmful in practice, because, for most of the generations of an evolutionary run, the population will converge into an area of the genotypic search space which it continues to explore.

The convergence argument was supported by several works presented in the literature. Some authors proposed new crossover operators and representations [8], some others concentrated on the topology or weight evolution in order to apply it, for example by evolving sub-populations of neurons in pre-established topologies, or by applying graph-matching techniques to non-fixed structures [5,10]. Particular attention was given to two empirical studies ([9], [7]) that concentrated their attention on the evolution of the single network unit involved in the crossover operator, the hidden node. Indeed, their idea was to emphasize the equivalence between hidden nodes of ANNs, in order to identify similarly performing units prior to crossover, avoiding all the disruptive effects stated above.

Following this idea, we extend one of the neuro-genetic approaches presented in the literature [1,2], which implements a joint optimization of weights and network structure, by defining a novel crossover operator. This operator allows recombination of individuals that have different topologies, but with hidden nodes that are similarly performing in the cutting point of the hidden layer randomly chosen (indicated in the approach as *local similarity*). The evolutionary process does not consider only a part, but complete multilayer perceptrons (MLPs), achieving satisfactory performances and generalization capabilities, as well as reduced computational costs and networks sizes.

3 The Neuro-genetic Approach

The overall algorithm is based on the joint optimization of structure and weights, here briefly summarized; a more complete and detailed description can be found in the literature [2]. It uses the error back-propagation (BP) algorithm to decode a *genotype* into a *phenotype* NN. Accordingly, it is the genotype which undergoes the genetic operators and which reproduces itself, whereas the phenotype is used *only* for calculating the genotype's fitness. The rationale for this choice is that the alternative of applying BP to the genotype as a kind of 'intelligent' mutation operator, would boost exploitation while impairing exploration, thus making the algorithm too prone to being trapped in local optima.

Training the network weights with the BP algorithm as a way to 'grow' a mature individual (phenotype) from a sort of an initial 'embryonic' network, encoded by the genotype, realizes what is called an *indirect encoding*. Thanks to this encoding, individual ANNs are not constrained to a pre-established topology. In this sense, the population is initialized with different hidden layer sizes and different numbers of neurons for each individual according to two exponential distributions, in order to maintain diversity among all of them in the new population. Such dimensions are not bounded in advance, even though the fitness function may penalize large networks. The number of neurons in each hidden layer is constrained to be greater than or equal to the number of network outputs, in order to avoid hourglass structures, whose performance tends to be poor. Indeed, a layer with fewer neurons than the outputs destroys information which later cannot be recovered.

The evolutionary process adopts the convention that a lower fitness means a better NN, mapping the objective function into an error minimization problem. Therefore, the fitness used for evaluating each individual in the population is proportional to the mean square error and to the computational cost of the considered network. This latter term induces a selective pressure favouring individuals with reduced-dimension topologies.

3.1 Evolutionary Process

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

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

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

Elitism allows the survival of the best individual unchanged into the next generation and the solutions to get better over time. Then, the algorithm mutates the weights and the topology of the offsprings, trains the resulting network, calculates fitness on the test set, and finally saves the best individual and statistics about the entire evolutionary process.

Weights mutation perturbs the weights of the neurons before performing any structural mutation and applying BP to train the network. All the weights and the corresponding biases are updated by using variance matrices and evolutionary strategies applied to the synapses of each NN, in order to allow a control parameter, like mutation variance, to self-adapt rather than changing their values by some deterministic algorithms. Finally, the topology mutation is implemented with four types of mutation by considering neurons and layer addition and elimination. The addition and the elimination of a layer and the insertion of a neuron are applied with three independent probabilities, indicated as p^{+}_{layer}, p^{-}_{layer} and p^{+}_{neuron}, while the elimination of a neuron is carried out only if the contribution of that neuron is negligible with respect to the overall network output.

4 Crossover Operator

We have discussed above some issues relevant to the use of a crossover operator in an EANN context. In particular, two main drawbacks are emphasized [8], that

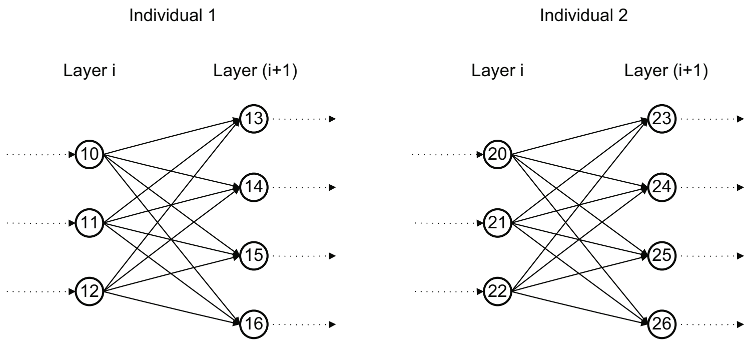


Fig. 1. Phase 1: identification of layers that have the same number of neurons, and that have the same number of neurons also in the layer i

Neuron	Contr.
15	0.95
13	0.84
16	0.52
14	0.37
23	0.87
26	0.69
24	0.55
25	0.40

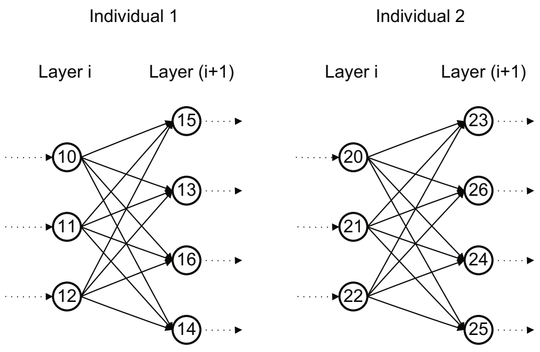


Fig. 2. Phase 2: the contribution of each neuron of the layer selected for the crossover is computed, then the neurons of each layer are swapped with respect to their contribution

correspond to the structural incompatibility between two individuals, due to the different network topologies, and to the parametric incompatibility, that is related to the difference of the weight values associated to the network synapses. The crossover operator proposed in this work aims at overcoming such drawbacks, and it is carried out in four phases.

Phase 1 (Figure 1): two individuals are selected from the population and the algorithm looks for a "local similarity" between the two individuals. We refer to "local similarity" as a situation in which, in both individuals, there are two consecutive layers (i and $[i+1]$) with the same number of neurons. This is a necessary condition for the application of our crossover operator because, this way, we want to overcome the problem related to the structure incompatibility between the individuals. If this condition is satisfied, layer $[i+1]$ is selected for the application of the crossover operator. The condition for the application of the

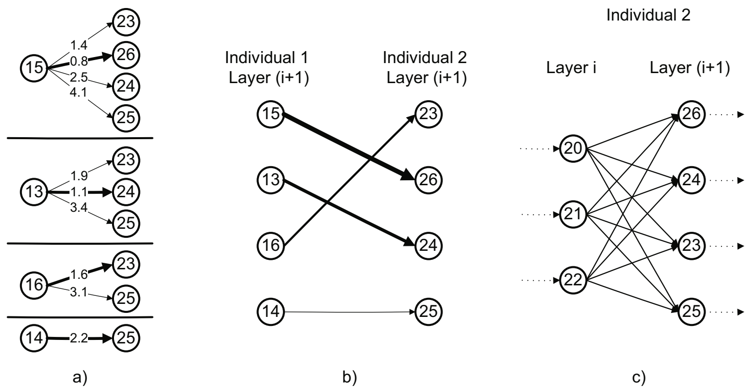


Fig. 3. Phase 3: each neuron of the layer in Individual 1 is associated with the most 'similar' neuron of the layer in Individual 2, the neurons of the layer of Individual 2 are re-ranked by considering the associations with the neurons of the layer in Individual 1

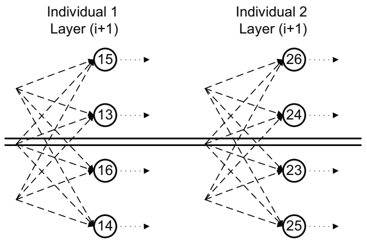


Fig. 4. Phase 4a: a cut-point is randomly selected and the neurons above the cut-point are swapped

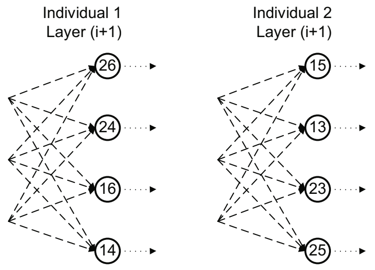


Fig. 5. Phase 4b: the offspring generated by the crossover operator

operator is checked for all hidden layers; therefore, two compatible individuals may be crossed more than once, i.e., at multiple crossover points.

Phase 2 (Figure 2): for each neuron of the selected layers, the algorithm computes the corresponding contribution (i.e., output over the training set), which strongly depends on its input connections. Then, in each individual, the neurons of the selected layers are ranked by considering their contribution. Figure 2 shows an example of the possible contributions, together with the calculated rank.

Phase 3 (Figure 3): we exploit the rank computed in the previous step to create the associations between the neurons of the two individuals. For each instance of the training set we compare the output of each neuron of the layer $[i + 1]$ on Individual 1 with the output of the each neuron of the layer $[i+1]$ of Individual 2 and we compute the overall difference between the neurons (Figure 3a). Starting from the neuron of Individual 1 that has the highest contribution, we associate each neuron of Individual 1 with the neuron of Individual 2 that has the lowest output difference (Figure 3b). When an association is created, the associated neuron of Individual 2 becomes unavailable for the subsequent associations (in the proposed example, neuron 15 is associated to neuron 26; therefore, neuron 26 might not be associated to any other neuron of Individual 1). Finally, the neurons of the selected layer in Individual 2 are re-ranked by considering the associations with the neurons of the selected layer in Individual 1 (Figure 3c).

Phase 4: the last phase of the algorithm generates the offspring. A cut-point is randomly selected between neuron 1 and neuron n (Figure 4), then the algorithm swaps the weights of the neurons that are above the cut-point and it maintains unchanged the others (Figure 5).

5 Experiments and Results

The approach has been applied to two real-world classification problems in the medical domain, namely the Pima Indian diabetes problem and the heart disease problem. The datasets have been obtained from the UCI Machine Learning Repository and, in this work, they have been partitioned into three sets, respectively, a training set (50% of the data), used to train the network; a test set (25% of the data), used to stop the training and avoid overfitting; a validation set (25% of the data), used to test the generalization capabilities of a network. Even if the literature reports some guidelines [14], it is important to stress that, following the commonly accepted practice of machine learning, no standard rule is given to the datasets nomenclature. The input attributes of both diabetes and heart disease data have been rescaled, before being fed as inputs to the population of ANNs, through a Gaussian distribution with zero mean and standard deviation equal to 1.

All the experiments have been carried out by considering different settings for the topology mutation parameters p^+_{layer},p^-_{layer} and p^+_{neuron}, defined in the range $[0.0, 0.5]$, and for the crossover p_{cross} (the 'desired' probability, see Section 3.1) in order to find out the optimal setting able to define the best performance. For each different configuration, we performed 20 runs and the results obtained for the two problems are reported in Table 1, showing the averaged and best accuracy, and the standard deviation. Due to space reasons we only report the results obtained by applying some of the parameter configurations.

We investigated the neuro-genetic approach through two different directions, that refer to the impacts of, respectively, the crossover operator and the different mutation probabilities. The rationale of this choice is to investigate what happens when diversity is injected into the population by the crossover operator, or when it is injected by the mutation operator. Particular attention has been given to the sensitivity of the crossover rate. Indeed, with the Pima dataset, when mutation rates were set to low values, the proposed method was sensitive to a crossover rate with high values, defined in the range $[0.7, 1.0]$. On the other hand, it was possible to notice that, in the second part of the experiments on Pima, when the

Table 1. Results obtained on the PIMA and HEART validation datasets

Settings				PIMA			HEART		
p^+_{layer}	p^-_{layer}	p^+_{neuron}	p_{cross}	Avg. Acc.	Best Acc.	Std. Dev.	Avg. Acc.	Best Acc.	Std. Dev.
0.05	0.05	0.05	0.4	0.7992	0.8229	0.0096	0.8352	0.8529	0.0163
			0.7	0.8059	0.8229	0.0111	0.8389	0.8823	0.0236
			1.0	0.8002	0.8125	0.0106	0.8404	0.8676	0.0174
0.20	0.05	0.10	0.4	0.7989	0.8125	0.0092	0.8279	0.8529	0.0253
			0.7	0.7968	0.8125	0.0122	0.8367	0.8823	0.0282
			1.0	0.8010	0.8281	0.0155	0.8441	0.8676	0.0130
0.30	0.05	0.30	0.4	0.8057	0.8333	0.0125	0.8375	0.8823	0.0221
			0.7	0.8033	0.8281	0.0150	0.8323	0.8676	0.0168
			1.0	0.8002	0.8281	0.0198	0.8411	0.8970	0.0222
0.50	0.05	0.50	0.4	0.8039	0.8177	0.0131	0.8433	0.8823	0.0235
			0.7	0.7986	0.8177	0.0116	0.8448	0.8970	0.0226
			1.0	0.8013	0.8125	0.0121	0.8316	0.8970	0.0383

mutation probabilities of adding layers and neurons increased, better averaged solutions were obtained with a lower crossover probability. In general, a probability of applying crossover equal to 0.7 (i.e., 70%) gave the best average accuracy.

A slightly different situation holds for the Heart dataset, where all the mutation settings had better results with high crossover probability. This indicates that, at least in the case with low mutation, the effect of crossover might be to help evolutionary search. Moreover, when we adopted a high mutation probability, the further diversity introduced by the crossover operator was helpful in preventing the algorithm from converging to a locally optimal solution. In this case, the best average accuracy was reached with high values for both mutation and crossover parameters. The same considerations could also be valid for the best accuracy. By observing the standard deviation, we can notice that the joint evolution of network weights and topologies together with the use of the crossover has considerably low variance values for the accuracy on the testing error, showing the robustness of the evolutionary process. A crucial observation is that the computational load of the crossover operator is negligible with respect to the one of fitness evaluation, since the data needed to perform the neuron associations in the crossover are the same computed during fitness evaluation. Computing the neuron associations thus comes almost for free, provided that those data are stored.

Although a direct comparison with other related approaches is difficult because the algorithms and methods of obtaining the generalization capabilities of the models are different, it is interesting to compare the results of this neuro-genetic approach with other works already presented in the literature, shown in Table 2. The approaches marked with a star use a crossover operator for evolving ANNs, while the others refer to different machine learning solutions in which the crossover is not used for such classification problems. Our neuro-genetic approach obtains higher accuracies than all the others listed in the table, except one, with satisfactory results. The overall performances are also improved by the reduced computational costs of this approach, thanks to the capability of the algorithm of obtaining neural networks with reduced sizes, up to two hidden layers and four hidden nodes in each hidden layer. Even if in one case of the Heart disease problem, that refers to COVNET [6], the accuracy is slightly better than that of our approach, we are able to obtain more compact networks, outperforming the average network size with only three hidden nodes in one hidden layer, thus reducing the overall computational cost. It is important to notice that in Table 2 a recent work that used a combinatorial crossover for ANN design optimization [7] has not been considered for comparison, since it used a twofold dataset subdivision (i.e. train and test sets only) instead of the usual three, in its experimental campaign. This makes its reported results uncomparable to those shown in Table 2.

Table 2. Comparison among the accuracies of the proposed approach and other works already presented in the literature

Model	PIMA Accuracy	Heart Accuracy
Our neuro-genetic approach*	0.8059	0.8448
Graph Matching Rec. (A. Mahmood et al [10])*	0.7542	0.7924
COVNET (N. Garcia-Pedrajas et al [6])*	0.8010	0.8574
aSEPA (P.P. Palmes [12])*	0.7400	0.8000
EPNet (X. Yao [17])	0.7763	0.8323
acasper (N.K. Treadgold [16])	0.7686	0.8079
acascor (N.K. Treadgold [16])	0.7547	0.8011
BP (L. Prechelt [14])	0.7563	–
LogDisc (D. Michie [11])	0.7770	–
MPyramid (R. Parekh [13])	0.7680	–
MTiling (R. Parekh [13])	0.7710	–
MSM1 (K.P. Bennet [3])	–	0.8347
RBF GM (A. Roy [15])	–	0.8182

*) Approaches using a crossover operator.

6 Conclusion and Future Work

In this paper, we have presented a similarity-based crossover operator for the evolution of artificial neural networks. This operator considers the similarity between the neuron outputs in order to choose which neurons may be swapped between the networks. The experiments showed that the application of the crossover operator to a well-tested neuro-genetic approach achieved promising results, also compared to other already published approaches, demonstrating the viability of such an operator. The overall accuracy then confirms satisfactory performances with reduced computational costs, also thanks to the capability of the algorithm to evolve small network topologies.

Increasing the population size could be a good way to further improve the performance, but it requires much computational effort, that might impair the attractiveness of the approach.

References

1. Azzini, A., Tettamanzi, A.: Evolving neural networks for static single-position automated trading. Journal of Artificial Evolution and Applications 2008(Article ID 184286), 1–17 (2008)
2. Azzini, A., Tettamanzi, A.: A new genetic approach for neural network design. In: Engineering Evolutionary Intelligent Systems. SCI, vol. 82. Springer, Heidelberg (2008)

3. Bennett, K.P., Mangasarian, O.L.: Robust linear programming discrimination of two linearly inseparable sets. Optimization Methods Software 1, 23–34 (1992)
4. Fogel, D.: The advantages of evolutionary computation. In: Proc. of the Int. Conf. on Biocomputing and Emergent Computation, BCEC 1997, pp. 1–11. World Scientific, Singapore (1997)
5. Froese, T., Spier, E.: Convergence and crossover: the permutation problem revisited. Cognitive Science Research Papers, CSRP 596 (2008)
6. Garcia-Pedrajas, N., Hervas-Martinez, C., Munoz-Perez, J.: COVNET: A cooperative coevolutionary model for evolving artificial neural networks. IEEE Transactions on Neural Networks 14(3), 575–596 (2003)
7. Garcia-Pedrajas, N., Ortiz-Boyer, D., Hervas-Martinez, C.: An alternative approach for neural network evolution with a genetic algorithm: Crossover by combinatorial optimization. Neural Networks 19, 514–528 (2006)
8. Haflidason, S., Neville, R.: On the significance of the permutation problem in neuroevolution. In: Proc. of Genetic Evolutionary Computational Conference, GECCO 2009, pp. 787–794. ACM, New York (2009)
9. Hancock, P.: Genetic algorithms and permutation problems: a comparison of recombination operators for neural net structure specification. In: Proc. of IEEE Int. Workshop on Combinations of Genetic Algorithms and Neural Networks, COGANN 1992, pp. 108–122. IEEE Press, Los Alamitos (1992)
10. Mahmood, A., Sharmin, S., Barua, D., Islam, M.: Graph matching recombination for evolving neural networks. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007. LNCS, vol. 4492, pp. 562–568. Springer, Heidelberg (2007)
11. Michie, D., Spiegelhalter, D., Taylor, C.: Machine Learning, Neural and Statistical Classification. Ellis Horwood (1994)
12. Palmes, P., Usui, S.: Robustness, evolvability, and optimality of evolutionary neural networks. BioSystems 82, 168–188 (2005)
13. Parekh, R., Yang, J., Honavar, V.: Constructive neural network learning algorithms for pattern classification. IEEE Transactions on Neural Networks 11, 436–450 (1998)
14. Prechelt, L.: PROBEN1 — a set of neural network benchmark problems and benchmarking rules. Technical report, Fakultät für Informatik, Universität Karlsruhe (September 1994)
15. Roy, A., Govil, S., Miranda, R.: An algorithm to generate radial basis function (RBF)-like nets for classification problems. Neural Networks 8(2), 179–201 (1995)
16. Treadgold, N., Gedeon, T.: Exploring constructive cascade networks. IEEE Transactions on Neural Networks 10(6), 1335–1350 (1999)
17. Yao, X., Ieee, S., Liu, Y.: A new evolutionary system for evolving artificial neural networks. IEEE Transactions on Neural Networks 8, 694–713 (1997)

Indirect Encoding of Neural Networks for Scalable Go

Jason Gauci and Kenneth O. Stanley

School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
jgauci@eecs.ucf.edu, kstanley@eecs.ucf.edu

Abstract. The game of Go has attracted much attention from the artificial intelligence community. A key feature of Go is that humans begin to learn on a small board, and then incrementally learn advanced strategies on larger boards. While some machine learning methods can also scale the board, they generally only focus on a subset of the board at one time. Neuroevolution algorithms particularly struggle with scalable Go because they are often directly encoded (i.e. a single gene maps to a single connection in the network). Thus this paper applies an *indirect encoding* to the problem of scalable Go that can evolve a solution to 5×5 Go and then *extrapolate* that solution to 7×7 Go and continue evolution. The scalable method is demonstrated to learn faster and ultimately discover better strategies than the same method trained on 7×7 Go directly from the start.

1 Introduction

The game of Go has proven challenging for artificial intelligence because the branching factor and state space in Go render traditional approaches intractable [1]. Go demands new search techniques to reduce the branching factor, and abstract representations that can consolidate the state space. One promising such approach is machine learning, wherein techniques such as temporal difference learning or neuroevolution learn a value function from an abstract representation [2, 3, 4].

Yet even with such innovations, experienced human Go players can still consistently defeat the strongest of computer players without a handicap [5]. One notable difference between human players and most machine learning-based approaches to Go is that the human player begins to learn Go on a small board [6]. Humans can then *extrapolate* information learned on the smaller board to a larger board, thereby bootstrapping from it. Such extrapolation is challenging for machine learning algorithms, which often cannot transfer knowledge from one board size to another.

However, several notable exceptions exist that typically fall into one of two categories: (1) The first convert the Go board into a set of local features that are independent of the board size [2]; (2) the second class of methods scan sections of the board and *remember* notable positions and information [4, 3]. In both cases, the key is to view a small section of the Go board at one time. As a result, it is potentially difficult to learn tactics (e.g. ladders) that depend on a holistic view of the board.

In this paper, a new method of scaling is presented that breaks from the aforementioned techniques, yet can still scale the board to new sizes and continue learning.

The method is based on Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT), which evolves artificial neural networks (ANNs) that are *aware of* and *parametrized by* the geometry of the board. As a result, these ANNs are able to make holistic decisions based on seeing the entire Go board at once. HyperNEAT encodes ANNs through an indirect representation that has the ability to scale the Go board to new sizes *without changing the representation* and continue evolution. The result is that candidates evolved on 5×5 Go and then scaled and evolved further at 7×7 Go outperform candidates evolved solely on 7×7 Go without scaling. Thus the main contribution is to show that indirect encoding is a viable foundation for training scalable learners, and offers the unique potential to represent holistic solutions at variable sizes.

2 Background

In Go, two players take turns placing stones on an $n \times n$ grid. The standard board size is 19×19; however, common board sizes also include 5×5 and 9×9. The objective is to possess more stones on the board than the opponent at the end of the game. If a player is able to form a complete border around a group of the opponent's stones, the surrounded stones are removed from the board. The player with the most stones at the end is declared the winner. A complete description of Go can be found in [5] and [6].

Go is designed for play at several board sizes. However, few machine learning methods can modify the board size in the middle of training and continue learning. This section discusses several exceptions and reviews the NEAT and HyperNEAT methods.

2.1 Reinforcement Learning and Scalable Go

Because the strategies for 19×19 boards are very different than those for e.g. 9×9, players transitioning from small to large boards must continue to learn and refine their strategy and tactics [6]. Ideally, machine learning algorithms should also learn to play Go at varying board sizes without discarding tactics learned on smaller boards and starting from scratch.

Reinforcement learning has been applied to scalable Go through several approaches [3, 2, 4]. [2] introduce the idea of assigning a weight to each shape in a *shape set*. The key idea is that all shapes learned on a smaller board are analogous on a larger one. New shapes that exist only at the higher scale are introduced after scaling by initializing them with a weight of 0. [7], [8], and [9] follow a similar approach.

In a different approach, [4] evolved a neural network that controls a robot eye that has a small field of vision. The robot is able to move across the board and place pieces. Because the field of vision for the robot is smaller than the size of the Go board, the robot can learn local concepts independently of location. As a result, the roving eye can learn to play Go at any resolution.

[3] introduced a neuroevolution-based action-value approximator for Go that evolves a Multi-Dimensional Recurrent Neural Network (MDRNN) [10]. The MDRNN performs swipes across the Go board. To perform a swipe, the same neural network is evaluated at every position of the Go board. In this way, information is carried across the board through the output values. MDRNNs are inherently scalable because the network is only concerned with relative information.

While these methods have learned effective Go players, each of them relies on integrating a set of small, local views that are processed independently over time or space. The danger is that less holistic heuristics that are significantly simpler become attractive local optima. In general, an interesting question is whether it is possible to scale the Go board to new resolutions while also processing the entire Go board without relying on subsquares. HyperNEAT, reviewed next, creates such a capability.

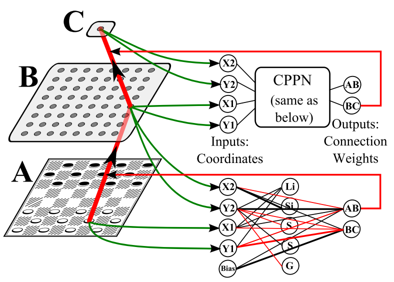
2.2 Indirect Encodings and HyperNEAT

The first methods to evolve both network structure and connection weights encoded networks *directly*, which means that a single gene in the genotype maps to a single connection in the phenotype [11]. NeuroEvolution of Augmenting Topologies (NEAT) is one such method [12]. In addition to evolving weights of connections, NEAT can build structure and add complexity. NEAT is a leading neuroevolution approach that has shown promise in board games and other challenging control and decision making tasks [12, 13, 4]. While this approach is straightforward, it requires learning each connection weight individually. Human engineering is one approach to overcoming this limitation. For example, [14] applies ANNs to checkers by dividing the board into subsquares and architecting the ANN to process them at different resolutions. However, ideally, evolution would capture patterns and regularities on its own.

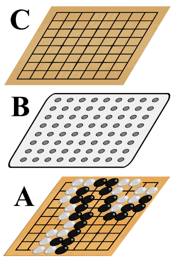
Indirect encodings give evolution the opportunity to explore patterns and regularities by encoding the genotype as a *description* that maps indirectly to the phenotype [15, 16, 17, 18]. That way, the genotype can be much smaller than the phenotype, which results in fewer variables to optimize for the evolutionary algorithm. *Compositional pattern producing networks* (CPPNs) are one such indirect encoding that draws inspiration from biology [19]. The idea behind CPPNs is that patterns such as those seen in nature can be described at a high level as a composition of functions that are chosen to represent several common motifs in patterns. The appeal of this encoding is that it allows patterns with regularities such as symmetry (e.g. with Gaussians), repetition (e.g. with periodic functions such as sine), and repetition with variation (e.g. by summing periodic and aperiodic functions) to be represented as networks of simple functions, which means that NEAT can evolve CPPNs just as it evolves ANNs.

Hypercube-based NEAT (HyperNEAT) is an algorithm that extends CPPNs, which encode two-dimensional spatial patterns, to also represent connectivity patterns [15, 20, 21]. That way, NEAT can evolve CPPNs that encode ANNs with symmetries and regularities that are computed directly from the geometry of the task inputs. The key insight is that $2n$-dimensional spatial patterns are isomorphic to connectivity patterns in n dimensions, i.e. in which the coordinate of each endpoint is specified by n parameters. To apply HyperNEAT to checkers, for example, the substrate (which is the name for the set of ANN nodes and their geometry in HyperNEAT) input layer is arranged in two dimensions to match the geometry of the checkers board (figure 1a). To compute the weight of a connection, the CPPN encoding works by inputting the coordinates of its endpoints (i.e. x_1, y_1, x_2, and y_2) and outputting the connection weight. All connections are computed in this way, in effect painting a pattern across the network connectivity.

[21, 20] originally introduced the type of representation in figure 1a for applying HyperNEAT to the game of Checkers. To distinguish the flow of information through



(a) Checkers Evaluation Function



(b) Go Action Selector

Fig. 1. Substrates for Board Games. Substrate (a) contains a two-dimensional input layer labelled *A* that corresponds to the geometry of a game board, an analogous two-dimensional hidden layer *B*, and a single-node output layer *C* that returns a board evaluation. The two CPPNs to the right of the board are depictions of the *same* CPPN being queried to determine the weights of two different substrate connections. In this way, a four-input CPPN can specify the connection weights of a two-layer network structure as a function of the positions, and hence the geometry, of each node. An action selector substrate (utilized in this paper) with an output for every possible move is shown in (b).

the policy network from the geometry of the game, a third dimension in the substrate represents information flow from one layer to the next. Along this third dimension, the two-dimensional input layer connects to an analogous two-dimensional hidden layer so that the hidden layer can learn to process localized geometric features. The hidden layer then connects to a single output node, whose role is to evaluate board positions. The CPPN distinguishes the set of connections between the inputs and the hidden layer from those between the hidden layer and the output node by querying the weights of each set of connections from a separate output on the CPPN (note the two outputs in the CPPN depiction in figure 1a). That way, the *x* and *y* positions of each node are sufficient to identify the queried connection and the outputs differentiate one connection layer from the next. Because the CPPN can effectively compute connection weights as a function of the *difference* in positions of two nodes, it can easily map a repeating concept across the whole board.

This approach allows HyperNEAT to discover geometric regularities on the board by expressing connection weights as a function of geometry. For a full description of HyperNEAT see [15] or [21].

3 Approach: HyperNEAT in Go

Because of the large branching factor in Go [1], board evaluation functions such as the HyperNEAT approach to checkers discussed above may not be tractable in practice. In the case of Go, there can be hundreds of boards to evaluate in a single move, even at the lowest ply. Thus an appealing alternative would be an *action selector* that evaluates the current state and suggests where to move, rather than a board evaluation function that must view many boards in the future to decide on a move. The next section explores this idea in more detail.

3.1 Evolving an Action Selector

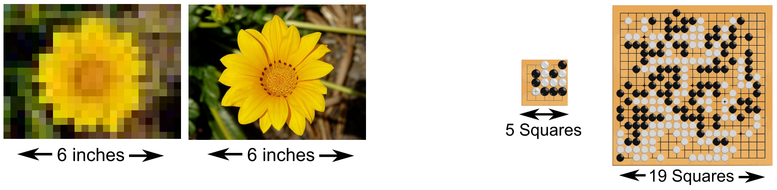
Because HyperNEAT can evolve high-dimensional structure as an indirect encoding, it opens up the possibility to evolve an action selector. This type of ANN contains an output for *each possible action* (figure 1b). In this case, an output exists for each square on the Go board. By activating the substrate, HyperNEAT populates each output with a value indicating the desirability of putting a piece in that position on the Go board. Thus no forward search through the game tree is needed, thereby saving significant computation. Once the substrate has been activated, the output with the highest activation is chosen and the corresponding square on the Go board undergoes a sanity check that prevents the network from making invalid moves in the game. As a result of this new architecture, the output, hidden, and input layers of the Go substrate all contain $n \times n$ nodes, where n denotes the size of the board. Given a board size of 7×7, the substrate thus contains 147 nodes and 4,802 connections. Indirect encoding can produce the smooth patterns of weights necessary to begin evolution with so many connections and still learn effectively. The next section explores the *substrate extrapolation* method that allows solutions to scale in this paper.

3.2 Substrate Extrapolation

A major problem for traditional neuroevolution is that the number of evaluations to solve a problem is related to the number of connections in the network being evolved [12]. Training a network with ten million connections can require significantly more evaluations than training one with one hundred. However, [15] showed that it is possible to query the *same* CPPN at varying substrate resolutions to create larger ANNs. Thus a promising potential approach to expanding the action selector size is to learn basic concepts on a small substrate, increase the substrate resolution, and then continue learning more advanced concepts at the higher resolution. This approach is designed to allow early, rapid learning of fundamental concepts.

There are two ways in HyperNEAT to scale a substrate input layer that represents a geometric space. The first is to sample the inputs at a higher resolution. This form of scaling, called *continuous substrate extrapolation*, preserves the geometric relationships between locations on the input signal (figure 2a). The two images, while different resolutions, exist within the same geometric area. That is, a specific location in the image does not change its *meaning* even if the resolution of the image changes. Thus the scaling changes *only* the distance between two adjacent pixels. Because CPPN inputs are by convention limited to a domain of $[-1,1]$, the CPPN effectively normalizes the width and height of the image regardless of resolution, and can thereby extrapolate the ANN to handle this form of scaling naturally. [15] demonstrated such continuous substrate extrapolation with HyperNEAT in a simple visual recognition domain.

While this method can be effective in visual tasks, some domains do not lend themselves to this form of scaling. For example, if the resolution of the Go board in figure 2b is increased, the size of the domain *itself* increases, as opposed to in the prior example, wherein it simply becomes more detailed. In such *discrete substrate extrapolation*, the size of a meaningful unit of information does not change as the resolution increases. As a result, a new method must be designed to handle this form of scaling.



(a) Continuous Extrapolation (b) Discrete Extrapolation

Fig. 2. Continuous Versus Discrete Extrapolation. In continuous substrate extrapolation (a), the bounds of the geometry do not change as the scale increases. In discrete extrapolation (b), the relative area of a single square stays the same, but the overall geometry is expanded outward. In this case, special care is needed to ensure that the network scales appropriately with the domain.

3.3 Discrete Substrate Extrapolation Implementation

The problem in discrete extrapolation is that the range of the input domain changes as the scale increases. To address this phenomenon, it is necessary to first decide on the maximum resolution of the system. In Go, this maximum resolution is 19×19, the size of the largest tournament Go board. The next step is to calculate the distance between two adjacent cells at this resolution. Because each input to the CPPN ranges from -1 to 1, the Go board must be rescaled to fit this new range. Thus the Go board position at index 0 maps to -1 and the position at index 18 maps to 1, and the distance between two adjacent cells in the Go board is therefore $\frac{2}{18}$. Interestingly, if the system is trained first at a lower resolution, e.g. 5×5, the smaller domain can be situated in the very same coordinate system (figure 2b). Increasing the resolution of each substrate layer during evolution is then an effective method to allow holistic complexification.

4 Experiment

The experiment in this paper aims to determine the effects of scaling HyperNEAT substrates on evolved Go action selectors. The player begins by playing ten games of Go against a fixed policy on a 5×5 board for 500 generations. The fixed policy player is *Liberty Player* from the SimplePlayers package of Fuego [22], who "tries to capture and escape with low liberty stones." A *liberty stone* is surrounded on three of the four sides with stones, and only has one empty adjacent space (i.e. one liberty). Liberty Player can be applied to boards of any size. Because Liberty Player places stones adjacent to stones with few liberties, it escapes captures and also quickly captures given the opportunity. When two or more potential moves are equally viable, Liberty Player picks one at random. These factors make Liberty Player a nontrivial opponent that provides sufficient challenge to demonstrate the utility of scaling. After training on a 5×5 Go board, the domain switches to playing Go against the same policy on a 7×7 board. Like the evolved player, Liberty Player is an action selector, that is, it only evaluates the current board and returns a location on which to place a stone.

During evolution, each candidate plays ten games of Go against the Liberty Player. After each game has ended, the candidate receives a reward based on the final score and the size of the board.

$$\text{fitness} = \begin{cases} 8b^2 & \text{if the evolved player wins} \\ \max\left(0, s + 2b^2\right) & \text{if the evolved player loses,} \end{cases} \tag{1}$$

where s denotes the final score and b denotes the size (i.e. length) of the board. This fitness function guarantees that all individuals will receive a positive fitness (as HyperNEAT requires), and that negative Go scores will still result in a positive reward. This convention puts additional emphasis on winning and also avoids rewarding individuals who win by a large margin in a single game, but lose the remaining games.

4.1 Experimental Parameters

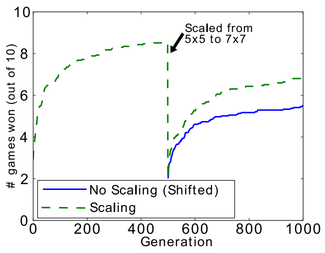
Parameter settings in the experiment follow precedent in applying HyperNEAT to checkers [20, 21]. The population size was 100 and each run lasted 500 generations. The disjoint and excess node coefficients were both 2.0 and the weight difference coefficient was 1.0. The compatibility threshold was 6.0 and the compatibility modifier was 0.3. The target number of species was eight and the drop-off age was 15. The survival threshold within a species was 20%. Offspring had a 3% chance of adding a node and a 5% chance of adding a link, and every link of a new offspring had an 80% chance of being mutated. Available CPPN activation functions were sigmoid, Gaussian, sine, and linear functions. Recurrent connections within the CPPN were not enabled. Signed activation was used in the CPPN and substrate, resulting in a node output range of $[-1, 1]$. By convention, a connection is not expressed if the magnitude of its weight is below a minimal threshold of 0.2 [15]; otherwise, it is scaled proportionally to the CPPN output. These parameters were found to be robust to variation in preliminary experimentation.

5 Results

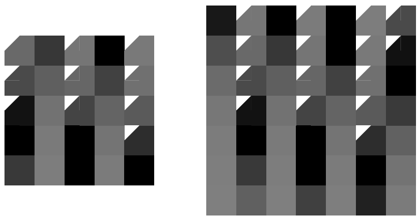
To determine the effect of scaling, substrate extrapolation is compared to an unscaled approach that plays only 7×7 Go. Although fitness drives evolution, fitness cannot be a benchmark for scaling performance because it is derived from the Go score, which varies with the size of the board. Therefore, the win rate is recorded during evolution and determines the effective skill of the player for the purpose of comparing the scaled to non-scaled methods.

Figure 3a compares the performance of the non-scaled 7×7 method against the scaled substrate, averaged over 25 runs. Note that the non-scaled results are shifted to the right so that the reader can easily compare the effects of scaling to not scaling. The scaling approach won significantly more games than the non-scaling approach in all generations after 524 (i.e. 24 generations after scaling) ($p < 0.05$).

To give an idea how scaling works, figure 3b shows a single *receptive field* connecting to the center output from the hidden layer of a scalable substrate at the two resolutions. Each grayscale box represents a link weight from a node in the hidden layer at that location to the center node of the output layer. White triangles in the corner



(a) Scaled versus non-scaled performance



(b) Receptive field at 5×5 and 7×7 scales

Fig. 3. Scaling Comparison and Visualization. The average performance of the generation champions over 25 runs of each variant is shown in (a). The performance is measured as the number of games won out of a possible 10 against Liberty Player. The scaled method wins significantly more than the non-scaled method in every generation beyond 524. A receptive field for the center output node on the substrate is shown in (b). Note that when the substrate is scaled to 7×7, the pattern is extrapolated outwards.

of an box denote negative weights. The individual from which this receptive map was extracted is from generation 500, at which the domain is scaled to 7×7. Note that the pattern of weights is extrapolated outward as the substrate is scaled from 5×5 to 7×7. To understand this result, recall that the substrate is scaled with the discrete substrate extrapolation method. As a result, when the substrate is created at 5×5, the CPPN is queried with all possible combinations of the numbers $-\frac{2}{3}, -\frac{1}{3}, -0, \frac{1}{3}, \frac{2}{3}$ as inputs x_1, x_2, y_1, y_2. The choice of inputs to the CPPN explicitly defines the particular connection weight that the CPPN will output. The substrate is scaled to 7×7 by expanding the inputs to include all possible combinations of the numbers -1, $-\frac{2}{3}$, $-\frac{1}{3}$, -0, $\frac{1}{3}$, $\frac{2}{3}$, 1. This expansion adds the additional cells shown in 3b. This new pattern is thereby an effective bootstrap for learning more advanced concepts at the higher scale.

6 Discussion and Future Work

The key contribution of this paper is to show that indirect encoding makes possible a new kind of holistic, scalable Go player. Interestingly, an evaluation at 7×7 takes *ten times longer* than the same evaluation at 5×5 because the network size is larger and the games take more turns to complete. A method that can learn fundamental concepts at a low board size can thus more quickly progress to more advanced concepts at higher sizes, and thereby learn them with less computational overhead.

The CPPN encoding allows the HyperNEAT substrate to input and output an entire board of neurons. This method thus differs from other scalable approaches that either divide the board into local segments [3] or local features [2]. Constructing a function from the holistic board geometry is important for several reasons. First, it removes the need for a human or external process to divide the search space into local features or segments. Second, constructing functions directly from geometry allow long-distance geometric relationships to be taken into account. For example, the decision to place a piece in Go often hinges not only on the position in the local area, but also on the state of conflicts elsewhere on the board and the geometric relationship of those conflicts with the local positions.

Future work will focus on incrementing to higher board sizes, evolving general Go players with HyperNEAT, and comparing them to other Go players. In addition, it is possible to bootstrap a Monte Carlo Tree Search (MCTS) algorithm with an action-evaluation function evolved by HyperNEAT. For example, the Upper Confidence Bounds Applied to Trees (UCT) algorithm is enhanced by adding a default policy [23]; however, the authors note that, "in many domains it is difficult to construct a good default policy." It is possible that HyperNEAT can evolve an effective default policy for UCT or any search algorithm.

7 Conclusion

This paper focused on the effects of scaling and demonstrated that players evolved incrementally through a scalable representation learn faster and more effectively than players evolved solely at the large scale. This result implies that fundamental concepts learned at a lower resolution facilitated further learning at the higher scale. The substrate extrapolation method scaled the information learned on the 5×5 Go board to the 7×7 board and the HyperNEAT algorithm was able to continue evolution at this new resolution. The main contribution is a step towards holistic neural strategies through indirect encoding that can be scaled to higher resolution or size.

References

[1] Burmeister, J., Wiles, J.: The challenge of Go as a domain for AI research: A comparison between go and chess. In: Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems. IEEE Western Australia Section, pp. 181–186 (1995)

[2] Silver, D., Sutton, R., Müller, M.: Reinforcement learning of local shape in the game of go. In: 20th International Joint Conference on Artificial Intelligence, pp. 1053–1058 (2007)

[3] Schaul, T., Schmidhuber, J.: Scalable neural networks for board games. In: Proceedings of the International Conference on Artificial Neural Networks (ICANN). Springer, Heidelberg (2008)

[4] Stanley, K.O., Miikkulainen, R.: Evolving a roving eye for Go. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 1226–1238. Springer, Heidelberg (2004)

[5] Botermans, J.: The Book of Games: Strategy, Tactics, and History. Sterling Publishing Co. (2008)

[6] Shotwell, P.: Go! More Than a Game. Turtle Publishing (2003)

[7] Silver, D., Sutton, R.S., Müller, M.: Sample-based learning and search with permanent and transient memories. In: Proceedings of the 25th International Conference on Machine Learning, pp. 968–975. ACM, New York (2008)

[8] Enzenberger, M.: Evaluation in Go by a neural network using soft segmentation. In: Advances in Computer Games: Many Games, Many Challenges: Proceedings of the ICGA/IFIP SG16 10th Advances in Computer Games Conference (ACG 10), Graz, Styria, Austria, November 24-27, p. 97. Kluwer Academic Pub., Dordrecht (2003)

[9] Schraudolph, N.N., Dayan, P., Sejnowski, T.J.: Temporal difference learning of position evaluation in the game of Go. In: Advances in Neural Information Processing Systems, pp. 817–817 (1994)

[10] Graves, A., Fernandez, S., Schmidhuber, J.: Multi-dimensional recurrent neural networks. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 549–558. Springer, Heidelberg (2007)

[11] Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)

[12] Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10, 99–127 (2002)

[13] Stanley, K.O., Miikkulainen, R.: Continual coevolution through complexification. In: Genetic and Evolutionary Computation Conference (2002)

[14] Fogel, D.B.: Blondie24: Playing at the Edge of AI (2002)

[15] Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life 15(2), 185–212 (2009)

[16] Hornby, G.S., Pollack, J.B.: Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8(3) (2002)

[17] Bongard, J.C.: Evolving modular genetic regulatory networks. In: Proceedings of the 2002 Congress on Evolutionary Computation (2002)

[18] Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2), 93–130 (2003)

[19] 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)

[20] Gauci, J., Stanley, K.O.: A case study on the critical role of geometric regularity in machine learning. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008). AAAI Press, Menlo Park (2008)

[21] Gauci, J., Stanley, K.O.: Autonomous evolution of topographic regularities in artificial neural networks. Neural Computation 22(7), 1860–1898 (2010)

[22] Enzenberger, M., Müller, M.: Fuego–an open-source framework for board games and go engine based on monte-carlo tree search. Technical report, Technical Report TR09-08, University of Alberta, Edmonton (2009)

[23] Gelly, S., Silver, D.: Combining online and offline knowledge in uct. In: Ghahramani, Z. (ed.) Proceedings of the International Conference of Machine Learning (ICML 2007), pp. 273–280 (2007)

Comparison-Based Optimizers Need Comparison-Based Surrogates

Ilya Loshchilov[1,2], Marc Schoenauer[1,2], and Michèle Sebag[1,2]

[1] TAO Project-team, INRIA Saclay - Île-de-France[⋆]
[2] Laboratoire de Recherche en Informatique (UMR CNRS 8623)
Université Paris-Sud, 91128 Orsay Cedex, France
FirstName.LastName@inria.fr

Abstract. Taking inspiration from approximate ranking, this paper investigates the use of rank-based Support Vector Machine as surrogate model within CMA-ES, enforcing the invariance of the approach with respect to monotonous transformations of the fitness function. Whereas the choice of the SVM kernel is known to be a critical issue, the proposed approach uses the Covariance Matrix adapted by CMA-ES within a Gaussian kernel, ensuring the adaptation of the kernel to the currently explored region of the fitness landscape at almost no computational overhead. The empirical validation of the approach on standard benchmarks, comparatively to CMA-ES and recent surrogate-based CMA-ES, demonstrates the efficiency and scalability of the proposed approach.

1 Introduction

The importance of invariances in science has long been acknowledged. In computer science in particular, the invariance of an algorithm with respect to a given transformation of the problem domain is a source of robustness, as any theoretical or empirical result that is demonstrated for a given problem instance can be extended to the whole class of problems obtained by applying the transformation. For instance, many bio-inspired optimization algorithms such as tournament-based EAs, PSO, or DE only rely on comparisons of the fitness function, making them invariant under any monotonous transformation of the fitness. From a theoretical perspective, this invariance property is a source of robustness [5]; from an algorithmic perspective, it removes the need to tune the algorithm hyper parameters according to some (generally unknown) scale of the fitness function. In the realm of continuous optimization, the state-of-the-art CMA-ES [8] is known to achieve invariance with respect to orthogonal transformations of the search space. CMA-ES extreme robustness with respect to internal parameter tuning, and its outstanding performances for many types of fitness functions [7] are attributed in part to this invariance property, the importance of which is witnessed by the variability of other algorithm performances depending on e.g. the separability or condition number of the fitness function [1].

[⋆] Work partially funded by FUI of System@tic Paris-Region ICT cluster through contract DGT 117 407 *Complex Systems Design Lab* (CSDL).

Meta-model assisted optimization is used to decrease the number of evaluations of computationally expensive fitness functions in the framework of continuous optimization: a *surrogate model* of the fitness is built on the fly, and used *in lieu* of the actual fitness. Such method, also known as "Response Surface Method", has been used for long in the Numerical Engineering community [2]. Because Evolutionary Algorithms (EAs) require a lot of fitness evaluations, much work has been devoted in the last decade to specifically tune meta-model assisted approaches to EAs (see e.g. [9] for a survey – and section 2).

Unfortunately, building a surrogate model from the fitness values gathered during the search definitely obliterates any invariance by monotonous transformation of the fitness. Preserving such invariance in a surrogate-based approach requires the surrogate model to only comply with the ranks of the sample points with respect to the fitness function. In the realm of statistical Machine Learning, rank-based Support Vector Machines (SVMs) precisely aims at learning a model from the only ordering of the sample points [15]; such a rank-based surrogate could be used *in lieu* of a value-based surrogate, enforcing the comparison-based invariance of the underlying optimization algorithm. To the best of our knowledge, the only work investigating the use of rank-based SVM as surrogate model within a meta-model assisted EA is Runarsson's [14]; while this approach was reported to bring small improvements over a CMA-ES baseline, a major issue regards the choice of the kernel, the Achilles heel of all SVM-based methods.

Following the path opened by [14], this paper investigates the use of rank-based SVM surrogate models within CMA-ES. It further borrows [11] the use of the Covariance Matrix adapted by CMA-ES, viewed as the proper metric to look at the region of the fitness landscape currently explored. Finally, the paper contribution is to integrate a rank-based Support Vector Machine surrogate within CMA-ES, where the SVM kernel is set to the covariance matrix adapted by CMA-ES. Section 2 surveys evolutionary model-assisted approaches, focussing on rank-based surrogates and CMA-ES. Section 3 introduces the proposed algorithm, called ACM-ES for (alphabetically) ranked CMA-ES; it details how a rank-based SVM is tightly coupled with CMA-ES, using the change of representation induced by the current covariance matrix to derive an adaptive kernel with almost no computational overhead. The experimental validation of the approach is reported and discussed in section 4, and directions for further work are sketched in section 5.

2 Surrogate Models and Ranking

2.1 Approximate Ranking

Most approaches to evolutionary meta-model assisted optimization build and use the surrogate model in a way similar to that of classical optimization. The surrogate model is trained by regression, depending on the underlying model space (from mostly quadratic polynomials to neural networks, kriging aka Gaussian Processes or SVMs); the surrogate model (possibly taking into account its uncertainty) is used *in lieu* of the actual fitness function, or it is used to pre-screen promising solutions; and the model is updated based on computations of the true fitness on those promising solutions (see [9] for a detailed survey).

To our best knowledge, the first work acknowledging the fact that EAs "only" require accurate ranking information, as opposed to accurate approximation of the fitness function, is that of Th. Runarsson [13]. This work introduced the idea of *approximate ranking*: a simple weighted nearest neighbor regression model is used as surrogate model, and its validity is assessed based on whether it preserves the (objective function-based) ordering of points. The most promising individuals according to the surrogate model are evaluated with the objective function until the ranking of the best individuals stabilizes. Significant savings are reported on test functions compared to the baseline algorithm (an ES variant tailored to constrained optimization), although they do not allow comparison with state-of-the-art results. Moreover, the surrogate model is probably too simple to lead to competitive results.

Approximate ranking however inspired *Local Meta-Model CMA-ES* (lmm-CMA), proposed by Kern et al. [11]: a local quadratic model is build anew for each offspring generated with the usual CMA-ES procedure, and approximate ranking is used to adaptively determine the number of actual objective evaluations to be run at each generation. lmm-CMA significantly outperforms the original CMA-ES with a speed-up factor of circa 2-3, making it competitive with state-of-the-art approaches. The requirement on approximate ranking was later relaxed by another variant, nlmm-CMA [3], using the rank stability of the set of μ best offspring as stopping criterion (as opposed to, the rank of each offspring), and thus improving over lmm-CMA on most benchmark functions (detailed results will be given in section 4.2 for the sake of comparative validation).

There are however a few drawbacks with lmm-CMA algorithms, apart from the fact that they use a regression surrogate model and hence depart from the comparison-based invariance of CMA-ES. Firstly, they rely on quadratic approximations, and thus their performances decrease when the objective function is far from being quadratic (see section 4.2). Secondly, they must use the full quadratic model [11], and hence the surrogate model must be of order d^2, where d denotes the problem dimension; the regression problem thus is of order d^6, which makes it hardly scalable for medium size problems ($d > 20$).

2.2 Rank-Based Surrogate Model with Rank-SVMs

Another seminal idea regarding the comparison-based issue in meta-model assisted EAs is again due to Th. Runarsson [14], using rank-based learning to train the surrogate model. Let us briefly recall rank-based Support Vector Machines, assuming the reader's familiarity with SVM first principles [15].

Let $(x_1, \ldots, x_N)$ denote an N-sample in instance space X, assuming with no loss of generality that point x_i has rank i. Rank-based SVM learning [10] aims at a real-valued function $\mathcal{F}$ on X such that $\mathcal{F}(x_i) < \mathcal{F}(x_j)$ for all pairs i, j such that $i < j$. In the SVM framework, this goal is formalized through minimizing the norm of $\mathcal{F}$ (regularization term) subject to the ordering constraints, thus involving $N(N-1)/2$ constraints. A more tractable formulation [15] only involves the $N-1$ constraints related to consecutive points, $\mathcal{F}(x_i) < \mathcal{F}(x_{i+1})$ for $i = 1 \dots N-1$. The latter formulation was used in [14] and will also be used in the presented approach.

Using the kernel trick[1], ranking function $\mathcal{F}$ is defined as a linear function w w.r.t. some feature space $\Phi(X)$, i.e. $\mathcal{F}(x) = \langle\, w, \Phi(x)\, \rangle$. With same notations as in [15], the primal optimization problem is defined as follows, where slack variable ξ_i and constant C_i respectively account for the violation of the i-th constraint, and the weight of the violation, to be minimized:

$$\begin{array}{l}\text{Minimize}_{\{w,\,\xi\}} \frac{1}{2}||w||^2 + \sum_{i=1}^{N} C_i \xi_i \\ \text{subject to } \begin{cases} \langle\, w, \Phi(x_i) - \Phi(x_{i+1})\, \rangle \geq 1 + \xi_i \;\; (i = 1 \dots N-1) \\ \xi_i \geq 0 \;\; (i = 1 \dots N-1) \end{cases}\end{array} \tag{1}$$

The corresponding dual problem, quadratic in the Lagrangian multipliers α, can be solved easily. Finally, the rank surrogate $\mathcal{F}$ is given as

$$\mathcal{F}(x) = \textstyle\sum_{i=1}^{N-1} \alpha_i (K(x_i, x) - K(x_{i+1}, x))$$

Like for any SVM-based approach, the main critical issue behind rank-based SVMs remains the choice of the kernel, that is known to be highly problem-dependent [15]. Furthermore, as pointed out in [4], the kernel used within an SVM-based surrogate should adapt to the optimization process: the optimal kernel is likely to change as search proceeds, exploring different regions of the search space. Some results related to kernel adaptation within SVM-based surrogate in CMA-ES, using fixed kernels, have been obtained by updating the surrogate model using Kendall tests on ranks (although approximate ranking could also have been used). The computational gains in terms of number of function evaluations do depend on the kernel, as was expected; the gains however are reported in [14] to rapidly decrease with the dimension d of the problem.

3 Rank-SVM CMA-ES

The main contribution of the paper is to integrate the rank-based surrogate approach first proposed by [14] within the CMA-ES framework, taking advantage of the Covariance-Matrix Adaptation scheme to adaptively define the kernel of the rank-based surrogate.

3.1 From CMA-ES to Rank-SVM Kernel

By construction, CMA-ES adapts the covariance matrix describing the local structure of the fitness landscape. After [6], CMA-ES proceeds by adapting the

1 The so-called kernel trick supports the extension of the SVM approach from linear to non-linear functional spaces, by mapping instance space X onto some *feature space*. It only requires the scalar product in feature space to be computable on instance space X through a *kernel* function K: $K(x, x') =_{def} \langle\, \Phi(x), \Phi(x')\, \rangle$.

problem encoding, and performing a Cumulative Step-size Adaptation algorithm in the transformed space. The change of coordinates, defined from the current covariance matrix C and the current mean value m, reads:

$$x'_j = C^{-1/2}(x_j - m), \tag{2}$$

Notably, the CMA information was directly used in [11] to building quadratic surrogate models; when training a quadratic surrogate model centered on x^*, the weight of each sample x was set to $\sqrt{(x - x^*)^T C^{-1} (x - x^*)}$.

In the case of a kernel-based surrogate model, it thus comes naturally to set the Radius-Based (RBF) kernel directly to the covariance matrix, with $\sigma > 0$:

$$K_C(x_i, x_j) = e^{-\frac{(x_i - x_j)^T C^{-1} (x_i - x_j)}{2\sigma^2}} \tag{3}$$

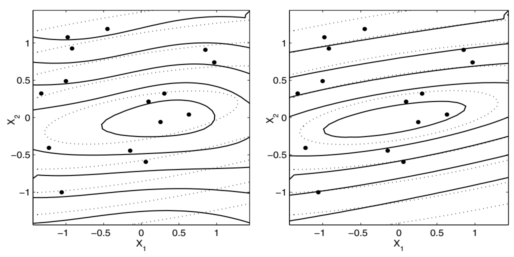
Fig. 1 (left) illustrates the potential gain of using such transformation for the simple case of the ellipsoid function, where the matrix C is exactly known. Interestingly, the change of coordinates is already computed within CMA-ES, therefore the transformation comes at almost no additional cost. Kernel width σ is set to the average distance between training points in the experiments. Another possibility, left for further work, could be to tie σ to CMA step size.

3.2 Overview of ACM-ES

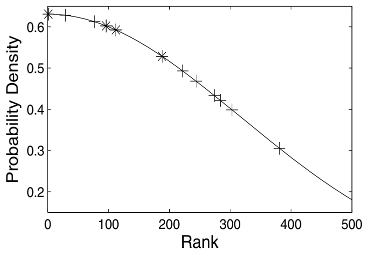
Having chosen its kernel after Eq. (3), the integration of a rank-SVM as surrogate model within CMA-ES raises three main issues: i/ how to train the surrogate model, i.e. how to select the current training sample in the set of all points evaluated with the true objective function; ii/ how to use the model within CMA-ES, without perturbing the delicate adaptive mechanism thereof; and iii/ how to select the new points which will be evaluated with the true objective function.

Regarding the first issue, i.e. the selection of the training sample, several requirements have been identified. Firstly, the number $N_{training}$ of training samples must increase with the dimension d of the search space. Using statistical learning arguments, $N_{training}$ should be of the order of the VC dimension of the model space. Note that after transformation (Eq. 2) the decision space is a variant of the sphere function, in the best case, or a noisy multimodal variant thereof in the worst case. A second requirement is that the training samples should not lie too far from the current mean m of the distribution used by CMA-ES to generate its offspring, since the transformation defined by the current covariance matrix only aims at the local structure of the fitness landscape around m. Finally, the analysis of preliminary experiments on the d-dimensional sphere function shows that $N_{training}$ should increase proportionally to $\sqrt{d}$; the proportionality constant however remains problem-dependent as will be seen in section 4.1. These $N_{training}$ selected points are the best points evaluated with the true fitness function so far.

The second issue regards how to use the rank-based surrogate within CMA-ES. Using the surrogate model *in lieu* of the true fitness is a risky option due



(a) Contour plots of ellipsoid function (dotted lines) and Rank-SVM surrogates (solid lines) obtained with isotropic RBF kernel (left) or exact transformed RBF kernel (right).



(b) $\lambda = 12$ offspring ('+' and '$*$') and $\lambda' = 4$ candidates for true evaluation ('$*$') selected among 500 pre-children.

Fig. 1. (a) The transformed RBF kernel is more appropriate than the isotropic one. (b) Selecting offspring from pre-children: mapping the ranks to a normal distribution.

to the lack of guarantees about errors in regions outside the training sample. A more conservative option thus is to use the surrogate model to pre-screen the offspring [12], generating many more *pre-children* than required, and keeping the best ones after the surrogate model. Such an approach however rapidly looses the offspring diversity, hindering the CMA-ES adaptive mechanism used to adapt the covariance matrix. Some tradeoff between the optimization of the objective and the adaptation of the covariance matrix must thus be found.

The proposed approach finally is a two-step process. In order to prevent premature convergence, and interfere as little as possible with CMA-ES cumulative step-size adaptation, a large number N_{test} of pre-children is drawn using the standard CMA Gaussian distribution; let them be noted $x_1, \ldots x_{N_{test}}$, assuming with no loss of generality them to be ranked after the surrogate model. The λ offspring are obtained by iteratively drawing a real number $a < N_{test}$ from distribution $\mathcal{N}(0, \sigma^2_{sel0})$ (where σ_{sel0} is a parameter of the algorithm), and retaining the pre-child with rank $\lfloor a \rfloor$. The same procedure is followed to select the points to be evaluated according to the true objective function, with the same rationale: on the one hand, one should select the best points according to the current surrogate model; on the other hand, some diversity must be preserved. Finally, i/ the point with top rank is selected and always evaluated (as in the approximate ranking approach [13]); ii/ other (λ'-1) points selected among the pre-children using a rank distribution $\mathcal{N}(0, \sigma^2_{sel1})$ are evaluated, using the same process as for the offspring selection albeit with a larger standard deviation ($\sigma_{sel1} > \sigma_{sel0}$). A typical distribution of the ranks of the λ offspring is depicted on Fig. 1 (right), legend +, for $N_{test} = 500$, $\lambda = 12$, and $\sigma^2_{sel0} = 0.4$, while points that will be evaluated with the true fitness are represented by $*$ ($\lambda' = 4$ and $\sigma^2_{sel1} = 0.8$).

In ACM-ES, a fixed number λ' of points is evaluated in each generation, thus bounding the complexity in terms of true fitness evaluation. The choice of the ratio λ/λ' thus controls the efficiency of the approach and the speedup w.r.t. the standard CMA-ES (where λ offspring are evaluated in each generation).

4 Experimental Validation

The experimental validation of ACM-ES investigates the performance of the approach comparatively to CMA-ES and nlmm-CMA, focussing on its scalability w.r.t. the problem dimension d, the robustness with respect to multi-modality, and with respect to the calibration of the surrogate training.

4.1 Experimental Settings

Seven uni- and multimodal benchmark functions have been considered (see Table 1, definitions in [11] and [3]), with dimension d ranging in $[2, 40]$ except for the Rastrigin function. Within ACM-ES, CMA-ES is used with its default parameters [8]. Reported results are based on 20 independent runs. The stopping criterion is reaching target value 10^{-10}, with a maximum of $1000d^2$ evaluations.

The rank-based surrogate was trained using $N_{training} = 30\sqrt{d}$ samples for all functions, except for Ellipsoid and Rosenbrock where it was set to $70\sqrt{d}$. The maximum number of iterations of the SVM learning algorithm was arbitrarily set to $50000\sqrt{d}$. The constraint weights C_i (Eq. 1) were set to $10^6(N_{training} - i)^{2.0}$, implying that the cost of constraint violation quadratically increases for top-ranked samples. For all functions except Rastrigin, $\lambda' = \frac{\lambda}{3}$, $\sigma^2_{sel0} = 0.4$, $\sigma^2_{sel1} = 2\sigma^2_{sel0} = 0.8$, $N_{test} = 500$. For Rastrigin function $\sigma^2_{sel0} = \sigma^2_{sel1} = 0.6$.

4.2 Results and Discussion

Firstly, experiments are conducted to estimate the empirical complexity of the surrogate training and using, using $100\sqrt{d}$ training points, stopping after $50000\sqrt{d}$ iterations and assessing the surrogated model on 500 test points. The empirical complexity with respect to dimension d (Fig. 2 (left) in log scale) is 1.13 (thus, slightly super-linear, contrasting with lmm-CMA complexity of $\mathcal{O}(d^6)$).

Secondly, the comparative validation of ACM-ES, nlmm-CMA and standard CMA-ES on all benchmark functions is reported in Table 2; lmm-CMA and nlmm-CMA results have been taken from original papers [11] and [3] when available; those of CMA-ES have been recomputed. Overall, ACM-ES outperforms lmm-CMA and nlmm-CMA algorithms on most problems, particularly so for problems with dimension $d > 4$. The invariance of ACM-ES w.r.t. monotonous

Table 1. Test functions, initialization intervals and initial std. dev. (from [11,3])

Noisy Sphere	$f_{NoisySphere}(\mathrm{x}) = (\sum_{i=1}^{d} x_i^2)\exp(\epsilon\mathcal{N}(0,1)$	$[-3,7]^d$	5
Ellipsoid	$f_{Elli}(\mathrm{x}) = \sum_{i=1}^{d} 10^{\frac{i-1}{d-1}} x_i^2$	$[1,5]^d$	2
Schwefel	$f_{Schwefel}(\mathrm{x}) = \sum_{i=1}^{d}(\sum_{j=1}^{i} x_j)^2$	$[-10,10]^d$	10
Schwefel$^{1/4}$	$f_{Schwefel^{1/4}}(\mathrm{x}) = (f_{Schwefel}(x))^{1/4}$	$[-10,10]^d$	10
Rosenbrock	$f_{Rosenbrock}(\mathrm{x}) = \sum_{i=1}^{d-1}\left(100.(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$	$[-5,5]^d$	0.5
Ackley	$f_{Ackley}(\mathrm{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) + \exp(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i))$	$[1,30]^d$	14.5
Rastrigin	$f_{Rastrigin}(\mathrm{x}) = 10d + \sum_{i=1}^{d}(x_i^2 - 10.\cos(2\pi x_i))$	$[1,5]^d$	2

Table 2. Computational effort SP1 (i.e. average number of function evaluations of successful runs divided by proportion of successful runs), standard deviations and speedup performance (spu) of ACM-ES, (n)lmm-CMA-ES and CMA-ES. Results in the (n)lmm-CMA column are the best of those in [11] and [3] (marked with leading "n:" for the latter). Successful runs are those who reached the target fitness value of 10^{-10}. The proportion of successful runs is given in parentheses if less than 100%. ϵ is the noise level (when relevant).

Function	n	λ	λ'	ϵ	ACM-ES		spu	(n)lmm-CMA		spu	CMA-ES	
$f_{Schwefel}$	2	6	3		186±5		2.0	**81±5**		4.5	370±32	
	4	8	3		289±9		3.0	**145±7**		6.0	879±60	
	5	8	3		344±9		3.2				1112±72	
	8	10	3		558±18		3.6	**282±11**		7.1	2010±82	
	10	10	3		801±36		3.3				2667±87	
	16	11	3		2204±74		2.3	**626±17**		8.2	5156±161	
	20	12	4		3531±179		2.0				7042±172	
	32	14	4		8933±337		1.7				15072±377	
	40	15	5		13440±281		1.7				22400±289	
$f_{Schwefel^{1/4}}$	2	6	3		551±12		2.8	**n:413±25**		3.7	1527±76	
	4	8	3		**783±8**		3.6	n:971±36		2.9	2847±109	
	5	8	3		**914±15**		3.8	n:1302±31		2.7	3505±114	
	8	10	3		**1366±25**		4.3				5882±146	
	10	10	3		**1774±37**		4.1				7220±206	
	16	11	3		**4193±88**		3.0				12411±198	
	20	12	4		**6138±82**		2.5				15600±294	
	32	14	4		**14796±310**		2.0				29378±330	
	40	15	5		**22658±390**		1.8				41534±466	
$f_{Rosenbrock}$	2	6	3		511±84		1.4	**n:252±52**		2.8	700±194	
	4	8	3		775±108		2.8	**n:719±54**	(0.85)	3.0	2187±376	(0.85)
	5	8	3		**854±89**		3.0	n:1014±94	(0.90)	2.5	2526±308	(0.95)
	8	10	3		**1388±139**		4.2	2494±511	(0.90)	2.3	5769±547	(0.85)
	10	10	3		**2059±143**	(0.95)	3.7				7669±691	(0.90)
	16	11	3		**5255±560**		3.1	7299±1154		2.2	16317±1281	(0.90)
	20	12	4		**11793±574**	(0.75)	1.8				21794±1529	
	32	14	4		**32261±2165**	(0.8)	1.6				52671±5587	
	40	15	5		**49750±2412**	(0.9)	1.6				82043±3991	
$f_{NoisySphere}$	2	6	3	0.35	413±114		1.0	**n:109±12**		3.7	407±61	(0.95)
	4	8	3	0.25	428±46		2.0	**n:236±19**		3.6	844±141	
	5	8	3	0.22	480±66		2.1				1014±68	
	8	10	3	0.18	**630±76**		2.6	n:636±33		2.6	1663±140	
	10	10	3	0.15	**766±90**	(0.95)	2.7				2058±148	
	16	11	3	0.13	**1119±115**		2.8	n:2156±216		1.4	3120±168	
	20	12	4	0.11	**1361±212**		2.8				3777±127	
	32	14	4	0.09	**1997±247**		2.9				5767±162	
	40	15	5	0.08	**2409±120**		2.9				7023±173	
f_{Ackley}	2	6	3		352±39		2.1	**n:227±23**		3.2	735±55	
	4	8	3		**540±29**	(0.95)	2.9				1577±83	
	5	8	3		**566±33**		3.4	n:704±24	(0.90)	2.2	1904±122	(0.95)
	8	10	3		**800±22**	(0.95)	3.8				3066±114	
	10	10	3		**892±28**		4.1	n:2066±119	(0.95)	1.8	3641±154	
	16	11	3		**1530±39**		3.7				5672±151	
	20	12	4		**1884±50**		3.5	8150±196		0.8	6641±108	
	32	14	4		**2747±62**		3.7				10063±203	
	40	15	5		**3690±80**		3.3				12084±247	
f_{Elli}	2	6	3		**393±19**		2.0				774±73	
	4	8	3		**582±24**		2.9				1688±11	
	5	8	3		**683±33**		3.4				2342±162	
	8	10	3		**1142±53**		4.0				4542±155	
	10	10	3		**1628±95**		3.8				6211±264	
	16	11	3		**4706±148**		2.8				13177±341	
	20	12	4		**8250±393**		2.3				19060±501	
	32	14	4		**27281±753**		1.6				44562±530	
	40	15	5		**33602±548**		2.1				69642±644	
$f_{Rastrigin}$	2	50	25		1640±242	(0.6)	1.2	**n:528±48**	(0.95)	3.6	1970±418	(0.85)
	5	140	70		23293±1374	(0.3)	0.5	**n:4037±209**	(0.60)	3.0	12310±1098	(0.75)

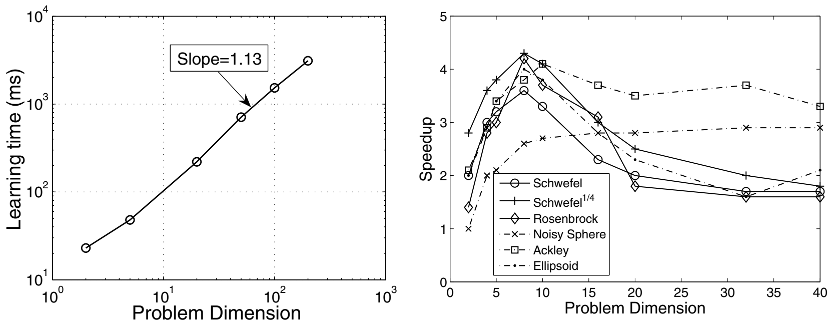


Fig. 2. Left: the cost of model learning/testing increases quasi-linearly with d. Right: the average speedup and speedup for all problems except Rastrigin.

transformations of the fitness is witnessed by its almost identical results on $f_{Schwefel}$ and $f_{Schwefel^{1/4}}$ functions, when the stopping criterion is adjusted accordingly (which is not the case for the results of Table 2). Likewise, the results on f_{Elli} confirm that ACM-ES also retains the good behavioral properties of CMA-ES with respect to the ill-conditioning of the fitness function. The speedup w.r.t CMA-ES is depicted on Fig. 2 (right) versus the problem dimension d. Interestingly, the speedup reaches its peak for d ranging in 8..10, then it decreases – except on the Noisy Sphere function. A possible explanation is that the noise level is comparatively less when the dimension increases (as in [3]), enabling the regularization involved in the model optimization to counteract the noise effects.

On the negative side, ACM-ES performs poorly on $f_{Rastrigin}$ function, and only solves it marginally for dimensions $d > 8$. This failure is attributed to the fact that ACM-ES does not handle well multi-modal diversity at the moment; it tends to accelerate the premature convergence to a local optimum, thus amplifying the weakness of CMA-ES on this benchmark problem: the best-performing versions of CMA-ES require an increasing population size [7]. Further work will consider the use of niching techniques to overcome this weakness.

5 Conclusion and Perspectives

The main contribution of the paper, ACM-ES, is a surrogate-based CMA-ES preserving invariance with respect to both monotonous transformations of the fitness function and orthogonal transformations of the search space. Comparison-based invariance is enforced by using rank-based Support Vector Machines to learn the surrogate model; coordinate invariance is enforced through using the covariance matrix adapted by CMA-ES as SVM kernel. Experimental validation confirms both invariance claims, and demonstrates the merits of the approach in terms of fitness evaluations and scalability w.r.t. the space dimension.

The main weakness of the approach is due to the failure of the surrogate model to account for multi-modal landscapes, as shown on the Rastrigin function; some improvements, e.g. related to niching, have been mentioned in the previous section and their validation is under way. Another issue regards the surrogate model hyper-parameters, which have been calibrated after preliminary experiments on the Sphere function conditionally to the carefully tuned hyper-parameters of CMA-ES [8]. A global approach, considering both sets of hyperparameters in an integrated way, would be appropriate. Another perspective, pointed out in [13], is to extend the approach to constrained optimization.

References

1. Auger, A., Hansen, N., Perez Zerpa, J., Ros, R., Schoenauer, M.: Experimental comparisons of derivative free optimization algorithms. In: Vahrenhold, J. (ed.) 8th Intl Symp. Experimental Algorithms. LNCS, vol. 5526, pp. 3–15. Springer, Heidelberg (2009)
2. Barthelemy, J.-F.M., Haftka, R.: Approximation concepts for optimial structural design – a review. Structural Optimization 5, 129–144 (1993)
3. Bouzarkouna, Z., Auger, A., Ding, D.: Investigating the local-meta-model CMA-ES for large population sizes. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010. LNCS, vol. 6024, pp. 402–411. Springer, Heidelberg (2010)
4. Cristianini, N., Shawe-Taylor, J.: An introduction to Support Vector Machines. Cambridge University Press, Cambridge (2000)
5. Gelly, S., Ruette, S., Teytaud, O.: Comparison-based algorithms are robust and randomized algorithms anytime. Evolutionary Computation 15(4), 411–434 (2007)
6. Hansen, N.: Adaptive encoding: How to render search coordinate system invariant. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 205–214. Springer, Heidelberg (2008)
7. Hansen, N., Auger, A., Ros, R., Finck, S., Pošík, P.: Comparing results of 31 algorithms from the BBOB-2009. In: GECCO Workshop Proc. ACM Press, New York (2010)
8. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
9. Jin, Y.: A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. Soft Computing 9(1), 3–12 (2005)
10. Joachims, T.: A support vector method for multivariate performance measures. In: Raedt, L.D., Wrobel, S. (eds.) Proc. ICML 2005. ACM International Conference Proceeding Series, vol. 119, pp. 377–384. ACM, New York (2005)
11. Kern, S., Hansen, N., Koumoutsakos, P.: Local meta-models for optimization using evolution strategies. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 939–948. Springer, Heidelberg (2006)
12. Rasheed, K., Hirsh, H.: Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In: Whitley, D., et al. (eds.) GECCO 2000, pp. 628–635. Morgan Kaufmann, San Francisco (2000)
13. 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. 401–408. Springer, Heidelberg (2004)
14. Runarsson, T.P.: Ordinal regression in evolutionary computation. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 1048–1057. Springer, Heidelberg (2006)
15. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)

A Cooperative Coevolutionary Approach to Partitional Clustering

Mitchell A. Potter[1] and Christine Couldrey[2]

[1] U.S. Naval Research Laboratory, Washington DC, USA
mitchell.potter@nrl.navy.mil
[2] AgResearch Ltd, Hamilton, New Zealand
christine.couldrey@agresearch.co.nz

Abstract. A challenge in partitional clustering is determining the number of clusters that best characterize a set of observations. In this paper, we present a novel approach for determining both an optimal number of clusters and partitioning of the data set. Our new algorithm is based on cooperative coevolution and inspired by the natural process of sympatric speciation. We have evaluated our algorithm on a number of synthetic and real data sets from pattern recognition literature and on a recently-collected set of epigenetic data consisting of DNA methylation levels. In a comparison with a state-of-the-art algorithm that uses a variable string-length GA for clustering, our algorithm demonstrated a significant performance advantage, both in terms of determining an appropriate number of clusters and in the quality of the cluster assignments as reflected by the misclassification rate.

1 Introduction

Cluster analysis consists of partitioning a set of patterns so those that are similar in some respect are clustered together and those that are dissimilar are clustered apart. The patterns, often in the form of numerical or categorical vectors, may represent many different types of observations. For example, in the field of epidemiology the patterns may represent the time and location of cases of a particular disease, in the field of image analysis the patterns may represent pixel intensities in a number of different spectral bands, in the field of molecular biology the patterns may represent the expression level of genes at different points in time, and so on. By performing cluster analysis on the patterns, we may gain useful information—enabling us, for example, to determine how a disease is spreading, segment an image into different land-cover regions, or characterize tumors based on their genetic signature.

Clustering algorithms can be categorized as hierarchical or partitional. While hierarchical clustering consists of creating larger clusters that are further partitioned into smaller clusters, or the agglomerative approach of starting with small clusters and merging them into larger clusters, partitional clustering creates a single partition of the observations given a specified number of clusters. The two best-known partitional clustering algorithms are K-means [1] and fuzzy C-means clustering [2]. These algorithms are similar, but K-means assigns each pattern uniquely to one cluster, while C-means assigns each pattern to multiple clusters with various levels of membership or *fuzziness*. K-means and its variants compute optimal cluster centroids and use these centroids to partition the patterns based on a relevant distance metric.

A challenge in partitional clustering is determining the number of clusters that best characterize the observations (see, for example [3]). In this paper, we present a novel approach for determining an optimal *number* of clusters and *assignment* of patterns to clusters simultaneously. Our approach is inspired by the process of sympatric speciation in nature. Specifically, each species represents a cluster, and its individuals are alternatives for that cluster's centroid. As a cooperative coevolutionary algorithm (CCEA) [4] evolves the centroids, new species are created that "steal" pattern assignments from the preexisting species while increasing the overall health of the ecosystem (reflected by the cluster validity). Once new species can no longer be created without diminishing cluster validity, the algorithm returns the best possible partitioning of the observations.

The rest of this paper is organized as follows. After an introduction to clustering with evolutionary algorithms we describe our new coevolutionary clustering algorithm in detail. This is followed by an empirical analysis of our approach on a number of synthetic and real data sets, and a comparison of its performance with that of the previous best approach to evolutionary clustering that also determines an optimal number of clusters and pattern assignments simultaneously. This earlier approach uses a variable string-length representation to evolve an optimal number of clusters [5,6]. Finally, we will present results that suggest not only why coevolution works better than the variable string-length representation on this particular task, but on compositional tasks in general.

2 Evolutionary Clustering

The original motivation for applying evolutionary algorithms (EAs) to clustering was the observation that partitions produced by K-means and its variants depend on the choice of seed values for the initial cluster centroids. While locally optimal partitions are found, there is no guarantee of finding the global optimum partitioning. It was shown in a comparison with K-means that the population-based approach used by EAs helps in this regard by making it less likely that the algorithm will converge to a local optimum or saddle point [7].

The earliest application of an EA to this problem by Raghavan [8] was to evolve cluster membership directly, but this proved to be quite slow in converging to an optimal solution. A more successful approach first proposed by Babu and Murty [9] evolved cluster centroids with an evolution strategy, which were then used to partition the patterns as in the K-means and fuzzy C-means algorithms. This approach has continued to be modified and enhanced by others, including using a genetic algorithm rather than an evolution strategy to evolve cluster centroids [10,11,12], the use of multiobjective optimization in evaluating the fitness of clusters [13], and alternative distance metrics such as those based on symmetry [14]. Other work includes using "do not care" symbols and variable-length chromosomes to simultaneously evolve cluster centroids and determine an optimal number of clusters [15,16,5,6], but as we will show, using the same objective function for both purposes can be overly restrictive.

3 Coevolutionary Approach

The only previous use of coevolution for clustering involved coevolving cluster centroids and the feature set used in the distance computation [17]. Our approach, which we call CCEA clustering, is quite different from this earlier approach in that ours is the first use of coevolution to simultaneously determine the number of clusters that best characterize the observations along with optimal partitioning. This approach is inspired by the process of sympatric speciation, which is a form of speciation that occurs when organisms share a common range or geographic area. Using this analogy, a species represents a population of alternative centroids for a particular cluster and the ecological range of the species is the entire data set of patterns being clustered.

CCEA clustering begins by evolving two species in separate populations. We have experimented with both GA- and ES-style algorithms for transitioning these populations from one generation to the next and achieved similar results. Populations have the form: $((x_1, x_2, \ldots, x_f)_1, (x_1, \ldots, x_f)_2, \ldots, (x_1, \ldots, x_f)_p)$, where x is a feature, f is the dimensionality of the patterns and p is the population size. Features could be represented as a binary strings or real numbers. Populations are initialized with random patterns from the data set to be clustered.

Since an individual represents only a single cluster centroid, before it can be evaluated it needs to be combined with a centroid from each of the other species, i.e., the number of species is equal to the number of clusters in a complete solution. Normally the current best centroid from each of the other species' populations are chosen. However, during the initialization phase it is not known which individuals are the best, so for the first round of evaluations a random centroid is chosen from each of the other species.

The next step in the evaluation process consists of computing the cluster membership of each pattern in the data set as in fuzzy C-means [2]:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{d(X_j, V_i)}{d(X_j, V_k)} \right)^{2/(m-1)}}, \tag{1}$$

where u_{ij} is the membership strength between pattern X_j and centroid V_i, c is the number of clusters (species), and $d(X, V)$ is the distance between a pattern and centroid using a relevant metric. In this study we use Euclidian distance. The exponent m controls the amount of fuzziness and will typically be set to 2.0, with values approaching 1.0 making the membership function less fuzzy.

Centroids are then recomputed based on cluster membership as follows [2]:

$$V_i = \frac{\sum_{j=1}^{n} u_{ij}^m X_j}{\sum_{j=1}^{n} u_{ij}^m}, \tag{2}$$

where n is the number of patterns in the data set. The individual being evaluated is then updated with its corresponding recomputed centroid and assigned a fitness using the following objective function from fuzzy C-means [2], which is a measure of cluster compactness:

$$J_m = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m d^2(X_j, V_i). \tag{3}$$

The species continue to evolve until their fitnesses are no longer significantly improving. This is measured by comparing the current fitness with the best fitness achieved over the past τ generations, where τ is a user specified parameter. When improvement stagnates, the health of the ecosystem is measured using a cluster validity index (see below). The first time stagnation occurs (when there are only two species) the current best solution is saved, a new species is created and evolution continues. Adding a new species will cause a reshuffling of pattern assignments as some of the patterns in the data set switch their membership over to the cluster represented by the new species and the preexisting species adapt to this by adjusting the location of their centroids. When subsequent stagnations occur, if the validity has *not* improved since the last species was created, that species is removed and the algorithm terminates—returning the best solution from just before the last species was added. Otherwise, as before, the current best solution is saved, a new species is created and evolution continues.

We have experimented with two different cluster validity indexes in this study: the commonly used Xie-Beni index (XB) [18] and the more recently developed PBMF index [19]. Both of these validity indexes use a combination of cluster compactness and separation to estimate the goodness of the partitioning.

Figure 1 illustrates the dynamics of CCEA clustering. As new species are added, the cluster validity index improves for awhile and then drops. The drop indicates the point at which too many clusters have been created, which occurs in this example when the sixth species is created at generation 26. The youngest species is then removed, the validity index returns to its previous level and the algorithm terminates. Note that the objective J_m continues to decrease (a smaller J_m indicates clusters are more compact) beyond the point where too many clusters have been created, which illustrates why J_m alone is not appropriate for determining the correct number of clusters.

4 Empirical Analysis

We compare the performance of CCEA clustering with a variable string-length GA (VGA) previously used by Maulik et al. for partitional clustering [5,6].

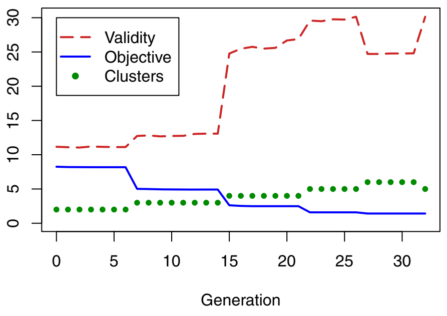


Fig. 1. Typical CCEA learning dynamics using the objective function J_m and the PBMF index for validity on a data set optimally partitioned into five clusters

Specifically, we are interested in evaluating sympatric speciation versus variable string-length to evolve an appropriate number of clusters along with cluster assignments. The VGA we use here shares a common code base with our CCEA to insure the results are not biased by implementation details of the algorithms.

Briefly, VGA initializes a single population with strings representing different numbers of centroids. The number of clusters then becomes an inherited trait along with the assignment of patterns to clusters. Over time evolutionary pressure will fill the population with individuals representing both the correct number of clusters and producing the correct assignments. VGA uses a crossover operator that can be applied to parents with different lengths, can produce offspring with lengths unlike either of the parents, and is constrained to cut chromosomes at cluster boundaries. For more information on the implementation of the VGA crossover operator, see the detailed description by Maulik [5].

Parameters shared by both algorithms include a 16-bit binary representation with values constrained to a range appropriate for the data set, population size of 50, bit-flip mutation at the rate equal to the reciprocal of the chromosome length, size-2 tournament selection, and a crossover rate of 0.6. The VGA uses the specialized crossover operator described above while the CCEA used standard two-point crossover. Both algorithms use the XB or PBMF validity index with $m = 2.0$ for the XB index and $m = 1.5$ for the PBMF index. While the VGA uses the validity index directly as its objective function, the CCEA uses the validity index only for determining the health of the ecosystem and J_m (eq. 3) as its objective function. Parameters unique to the CCEA include a minimum fitness improvement of 0.01 over 5 generations for the purpose of determining whether stagnation has occurred, which triggers the creation of a new species.

4.1 Evaluation Data Sets

We evaluated the CCEA and VGA algorithms by applying them to a number of synthetic and real-world data sets, most of which are similar to those used in previous studies. We chose the data sets to range from very easy to cluster to more challenging ones that typically mislead clustering algorithms.

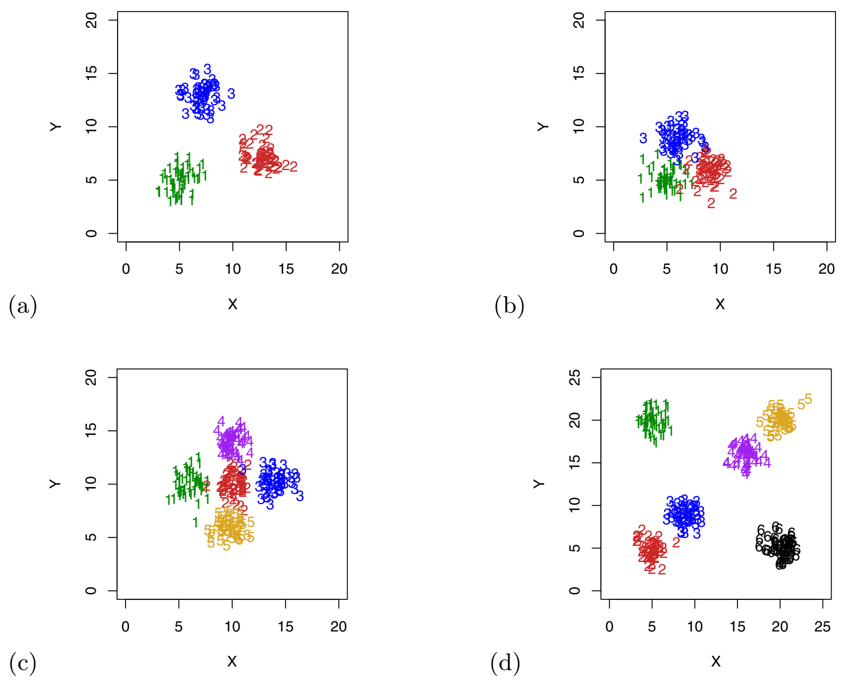
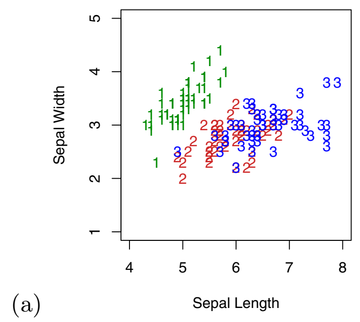


Fig. 2. Synthetic data sets produced with a “cluster generator”. Data sets include (a) Separate$_3$, (b) Overlap$_3$, (c) Overlap$_5$ and (d) Separate$_6$

The first four data sets, shown in Figure 2, are produced by sampling from normal distributions with a cluster generator rather than being produced by hand. This enables us to test the algorithms on multiple instances of the data sets. Specifically, we used ten instances of each of these data sets in our evaluation and multiple runs were distributed evenly across the instances. Of the four, Separate$_3$ is the easiest to cluster. Overlap$_3$ is more difficult to cluster due to the slight overlap of its three clusters. Clustering algorithms will typically not classify all the patterns in Overlap$_3$ correctly because with only two features there is not enough information to disambiguate all the patterns given the overlap. The proximity of the cluster distribution means will also mislead some cluster validity indexes. Overlap$_5$ has both overlap, which will produce misclassifications, and a central cluster surrounded by others at equal distance, which will mislead some validity indexes. In particular, the XB index will indicate this data set has only four clusters. Separate$_6$ clusters are separated, so one might assume they would be easy to cluster, but this data set has a form of symmetry that will mislead XB into indicating it has four clusters as in Overlap$_5$.

We also evaluated the algorithms on two real-world data sets shown in Figure 3. The first is the Iris data set from the UCI Machine Learning Repository [20]. This data set is one of the most commonly used pattern recognition test case, dating back to a 1936 paper by R.A. Fisher. It consists of 150 patterns,



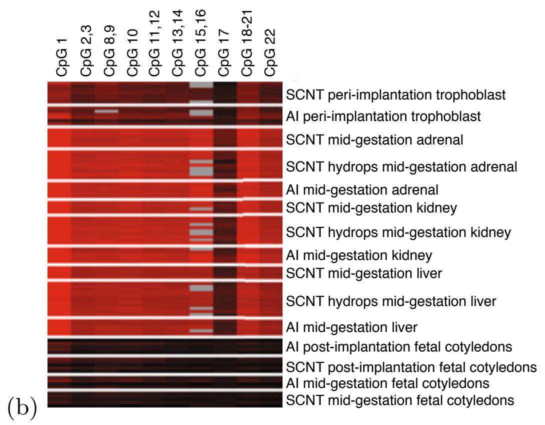


Fig. 3. Real-world data sets. (a) 2-dimensional slice of the 4-feature iris database. (b) Heat map showing DNA methylation levels of various CpG sites.

each having four real-valued features (two are shown in figure) describing the sepal and petal length and width of three classes of iris. One of the classes is linearly separable, but the other two are highly overlapping. Most cluster validity indexes will peak at two classes rather than three for this data set. The second is a recently collected epigenetic data set generated using a high-throughput Sequenom MassARRAY system that allows high-resolution interrogation of DNA methylation in defined genomic regions [21]. In mammals, methylation only occurs where the base cytosine is immediately followed on the DNA strand by the base guanine, which is called a CpG site, and is one of the key determinants in the control of gene expression. This data set, shown in Figure 3 using a heat map, consists of 90 patterns (rows), each having 10 features (columns) representing the average methylation at one or more CpG sites. Values range from 0.0 (dark) indicating no methylation to 1.0 (light) indicating all the cells in the sample are fully methylated at that site. This data set is characterized by a small number of samples per class, missing values (gray) and noise. It is also the only data set in which we do not know the "correct" clustering, i.e., while samples come from 15 distinct classes of various tissue types, gestation periods, and whether produced by artificial insemination (AI) or somatic cell nuclear transfer (SCNT), there may not be epigenetic differences between all of these classes. From the heat map we can clearly see three clusters, but there may be as many as fifteen.

4.2 Results

The results averaged over 50 runs on each of the evaluation data sets using the PDMF validity index are shown in Table 1. For each method, the table includes the average number of evolved generations, the average number of clusters in the final partition, and the average number of patterns misclassified each run. 95% confidence intervals for the means are included for all but the number of VGA generations, which is always set to 50. CCEA generally outperformed the VGA

Table 1. Performance of VGA and CCEA

	VGA			*CCGA*		
Data set	*Gens*	*Clusters*	*Misclass*	*Gens*	*Clusters*	*Misclass*
Separate$_3$	50	3.06 ± 0.07	0.34 ± 0.40	18.18 ± 0.37	3.00 ± 0.00	0.00 ± 0.00
Overlap$_3$	50	4.22 ± 0.28	26.56 ± 5.11	20.94 ± 1.67	3.48 ± 0.21	16.22 ± 5.21
Overlap$_5$	50	5.16 ± 0.13	16.82 ± 3.78	33.74 ± 1.11	5.00 ± 0.00	9.90 ± 0.62
Separate$_6$	50	6.12 ± 0.11	3.08 ± 2.52	38.36 ± 0.54	6.00 ± 0.00	0.30 ± 0.13
Iris	50	3.00 ± 0.00	27.96 ± 2.39	17.48 ± 0.18	3.00 ± 0.00	16.76 ± 0.12
Methylation	50	4.94 ± 0.07	11.14 ± 0.45	25.00 ± 0.00	5.00 ± 0.00	10.98 ± 0.04

Table 2. PBMF validity values for various numbers of clusters

	PBMF Objective					*J_m Objective*				
	Clusters					*Clusters*				
Data set	*3*	*4*	*5*	*6*	*7*	*3*	*4*	*5*	*6*	*7*
Separate$_3$	145.5	**146.7**	137.2	124.2	115.1	**143.9**	121.3	108.3	102.8	94.9
Overlap$_3$	14.1	**18.1**	18.0	16.5	15.4	**11.9**	11.8	11.3	10.7	10.5
Overlap$_5$	26.5	31.3	33.0	**34.0**	32.7	12.6	24.2	**27.4**	24.8	22.1
Separate$_6$	141.4	387.0	449.5	**666.5**	616.1	120.0	384.9	446.9	**659.1**	560.6
Iris	**32.2**	30.6	25.7	23.9	21.2	**28.1**	24.6	23.1	21.5	19.0
Methylation	7.1	7.7	**9.1**	8.6	7.3	6.6	7.5	**8.8**	7.1	7.1

both in terms of the number of clusters produced and quality of the assignments as indicated by the misclassification rate (Table 1). The CCEA also ran in fewer generations because of its automatic termination feature based on stagnation.

A two-sample Student's t-test was used to measure the significance of the difference in the mean number of clusters and misclassifications produced by CCEA versus VGA. The misclassification p-value was well below the 0.05 significance threshold for all data sets except Separate$_3$, which had a p-value of 0.0909, and Methylation, which had a p-value of 0.4820. The p-value for clusters was well below threshold for Overlap$_3$, Overlap$_5$, and Separate$_6$, while the p-value for both Separate$_3$ and Methylation was 0.0832. CCEA and VGA produced the same number of clusters in all runs for the Iris data set so no t-test was done.

We suspected that CCEA performed significantly better than the VGA because it was able to use the compactness measure J_m as its evolutionary objective due to the separate PBMF-based speciation mechanism for determining an appropriate number of clusters. The VGA had to use the PBMF index for its objective because it uses evolutionary pressure to determine both the number of clusters and the cluster assignments. This is contrary to speculation in [6] that evolving the PBMF validity index directly would produce better results.

To verify our hypothesis, we evolved different numbers of clusters with a GA and measured the PBMF values after using PBMF as the evolutionary objective versus using J_m as the objective. The results, averaged over 50 runs, are shown in Table 2 with the index peaks highlighted in **bold**. Using the J_m objective, all the PBMF peaks correspond to the correct number of clusters, while using PBMF as the objective only half of the peaks are correct.

We also performed VGA and CCEA clustering using the XB validity index, but in this case both methods produced the same number of clusters and the differences in misclassification rate were not statistically significant. Clustering using a GA over the same range of cluster numbers as before and comparing the XB values after using XB as the objective function versus using J_m as the objective, we found the index peaked on the same number of clusters in both cases. This explains why there were no significant differences between the VGA or CCEA clustering results using the XB index.

5 Conclusions

This paper presented a novel partitional clustering algorithm based on cooperative coevolution and inspired by the process of sympatric speciation that simultaneously evolves optimal cluster assignments while determining the number of clusters that best characterize the observations. The algorithm was evaluated on a number of synthetic and real data sets, and compared with a state-of-the-art algorithm that uses a variable string-length GA (VGA) for clustering.

Our algorithm demonstrated a significant performance advantage over the VGA, both in terms of determining an appropriate number of clusters and in the quality of the cluster assignments as reflected in the misclassification rate. Our algorithm also requires less computation because solutions with more clusters than necessary do not need to be evaluated, has a built-in termination procedure based on evolutionary stagnation so it does not evolve for more generations than necessary and does not require an estimate of the upper bound on the number of clusters. With respect to disadvantages, when clustering a data set in which the chosen validity index is multimodal, our algorithm may be more likely than the VGA to miss the globally optimal number of clusters if this solution is blocked by a coarser partitioning producing a locally optimum validity value. However, in practice this form of validation curve rarely occurs [19].

We believe these results go beyond clustering and illustrate an advantage of coevolution over VGAs for compositional problems in general. Coevolution outperformed the VGA because it partially decoupled the compositional aspect of the solution from evolving the populations, enabling one objective function to be used to adapt the number of components and other objective functions to be used to evolve the components themselves. Given the prevalence of problems that are both multiobjective and compositional, this will often be beneficial.

Acknowledgments. This work was supported by the New Zealand Royal Society Bilateral Research Activities Program and the New Zealand Foundation for Research, Science, and Technology (C10X0303 and CX10X0702).

References

1. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: 5th Berkeley Symp. on Mathematical Statistics and Probability, pp. 281–297. Univ. of Calif. Press, Berkeley (1967)
2. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Dordrecht (1981)
3. Beringer, J., Hüllermeier, E.: Adaptive optimization of the number of clusters in fuzzy clustering. In: IEEE Int. Conf. on Fuzzy Systems. IEEE, Los Alamitos (2007)
4. Potter, M.A., De Jong, K.A.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8(1), 1–29 (2000)
5. Maulik, U., Bandyopadhyay, S.: Fuzzy partitioning using real-coded variable-length genetic algorithm for pixel classification. IEEE Trans. on Geoscience and Remote Sensing 41(5), 1075–1081 (2003)
6. Pakhira, M.K., Bandyopadhyay, S., Maulik, U.: A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification. Fuzzy Sets and Systems 155, 191–214 (2005)
7. Murthy, C.A., Chowdhury, N.: In search of optimal clusters using genetic algorithms. Pattern Recognition Letters 17, 825–832 (1996)
8. Raghavan, V.V., Birchand, K.: A clustering strategy based on a formalism of the reproductive process in a natural system. In: 2nd Annual International ACM SIGIR Conf. on Information Storage and Retrieval, pp. 10–22. ACM, New York (1979)
9. Babu, G.P., Murty, M.N.: Clustering with evolution strategies. Pattern Recognition 27(2), 321–329 (1994)
10. Scheunders, P.: A genetic c-means clustering algorithm applied to color image quantization. Pattern Recognition 30(6), 859–866 (1997)
11. Hall, L.O., Ozyurt, B., Bezdek, J.C.: Clustering with a genetically optimized approach. IEEE Trans. on Evolutionary Computation 3(2), 103–112 (1999)
12. Maulik, U., Bandyopadhyay, S.: Genetic algorithm-based clustering technique. Pattern Recognition 33, 1455–1465 (2000)
13. Bandyopadhyay, S., Maulik, U., Mukhopadhyay, A.: Multiobjective genetic clustering for pixel classification in remote sensing imagery. IEEE Trans. on Geoscience and Remote Sensing 45(5), 1506–1511 (2007)
14. Bandyopadhyay, S., Saha, S.: GAPS: A new symmetry based genetic clustering technique. Pattern Recognition 40, 3430–3451 (2007)
15. Lee, C.Y.: Efficient Automatic Engineering Design Synthesis via Evolutionary Exploration. PhD thesis, Calif. Institute of Technology, Pasadena, Calif. (2002)
16. Bandyopadhyay, S., Maulik, U.: Genetic clustering for automatic evolution of clusters and application to image classification. Pattern Recognition 35, 1197–1208 (2002)
17. Kharma, N., Suen, C.Y., Guo, P.F.: PalmPrints: A novel co-evolutionary algorithm for clustering finger images. In: GECCO, pp. 322–331. Springer, Heidelberg (2003)
18. Xie, X.L., Beni, G.: A validly measure for fuzzy clustering. IEEE Trans. on Pattern Analysis and Machine Intelligence 13(8), 841–847 (1991)
19. Pakhira, M.K., Bandyopadhyay, S., Maulik, U.: Validity index for crisp and fuzzy clusters. Pattern Recognition 37, 487–501 (2004)
20. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
21. Ehrich, M., Nelson, M.R., Stanssens, P., Zabeau, M., Liloglou, T., Xinarianos, G., Cantor, C.R., Field, J.K., van den Boom, D.: Quantitative high-throughput analysis of dna methylation patterns by base-specific cleavage and mass spectrometry. Proceedings of the National Academy of Sciences USA 102(44), 15785–15790 (2005)

Feature Selection for Multi-purpose Predictive Models: A Many-Objective Task

Alan P. Reynolds, David W. Corne, and Michael J. Chantler

School of Mathematical and Computer Sciences,
Heriot-Watt University, Edinburgh, Scotland

Abstract. The target of machine learning is a predictive model that performs well on unseen data. Often, such a model has multiple intended uses, related to different points in the tradeoff between (e.g.) sensitivity and specificity. Moreover, when feature selection is required, different feature subsets will suit different target performance characteristics. Given a feature selection task with such multiple distinct requirements, one is in fact faced with a very-many-objective optimization task, whose target is a Pareto surface of feature subsets, each specialized for (e.g.) a different sensitivity/specificity tradeoff profile. We argue that this view has many advantages. We motivate, develop and test such an approach. We show that it can be achieved successfully using a dominance-based multiobjective algorithm, despite an arbitrarily large number of objectives.

1 Introduction

One of our motivating applications concerns images of textures (e.g. images of sections of wallpaper, fabric, carpet, etc.). Determining computationally whether two textures are similar to human eyes is a challenging and unsolved problem. However, experimental data are available that, for a varied set of textures, indicate which pairs users considered to be similar; we also have $\sim$5000 computational features for each texture. To support applications in texture search and browsing, we need to predict whether two textures are perceptually similar, using only the computational features. We also wish to reduce, via feature selection (FS), the number of features that need to be computed.

The selected features need to serve multiple purposes. Consider a search engine that, when given a 'query' texture, searches a database for other textures that would be perceived as being similar. Some users will be interested in as many 'matching' textures as possible and not be troubled by false positives. Others may require only a few textures but may insist that those provided be similar to the query case. Similar considerations apply in any domain where, for different predictive tasks involving the same data, the relative costs of false positives and false negatives vary significantly.

For such scenarios, in which FS is needed but the required performance profiles of the reduced feature set are complex and varied, we introduce a multiobjective (MO) approach that aims to find multiple subsets of features, each specialized for distinct required performance characteristics. In general, FS is easily phrased as a MO problem, e.g. one may maximize accuracy while minimizing a measure of feature subset complexity [7,8,9]. However, based on the many ways of measuring accuracy, we argue that this may be considered a problem with an infinite set of objectives. We explain this in sections 2 and 3, showing how the choice of sensitivity and specificity as measures of classifier performance leads naturally to a problem with an infinite set of objectives. In sections 4 and 5 we then describe an algorithm capable of handling such a problem. Sections 6 and 7 describe an investigation of this algorithm on three datasets. The effectiveness of the approach is discussed in section 8, along with ideas for further work.

2 Feature Subset Evaluation in the Wrapper Approach

When selecting features for a particular target application, the quality of the feature set is determined by the resulting performance of the application. Here we consider two-class problems, with the 'class of interest' considered 'positive' and the other 'negative'. Performance is calculated using the number of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). We make particular use of the following measures: *sensitivity* ($|TP|/(|TP|+|FN|)$); *specificity* ($|TN|/(|TN|+|FP|)$); and *confidence* ($|TP|/(|TP|+|FP|)$).

In the 'wrapper' approach to FS, feature set quality is estimated by applying a simple classification algorithm. So if the balanced error rate is to be minimized in the target application, the evaluation of a feature set should be an attempt to minimize the balanced error rate using a suitable classifier. If, as in the case of the texture search engine, the target application's performance is judged according to multiple, perhaps unknown accuracy measures, then feature subset evaluation should be an attempt to optimize each of these measures. In this case, use of a classification algorithm such as basic k-Nearest Neighbour (k-NN, as used by Emanouilidis [4], in an effort to optimize specificity, sensitivity and feature set size) is not appropriate, since k-NN generates only a single sensitivity-specificity pair that cannot simultaneously optimize each of the competing objectives. On the other hand, a model-based algorithm such as naive Bayes (NB) produces a probability that each record belongs to the class of interest. Concrete predictions are obtained by assigning a record to the class of interest if the probability is above a threshold. By varying the threshold, NB classifiers produce a range of different sensitivity-specificity trade-offs.

3 Uncountably Many-Objective Feature Selection

Users of a texture search engine will have varying preferences for the balance to be struck between sensitivity and specificity. Figure 1 shows the results of applying a classifier like NB to a single feature subset and the preferences of three users. User 1 is happy with just a few (12%) similar textures being returned, but is irritated by false positives. User 3 requires almost all (94%) similar textures to be returned, and will tolerate a large number of false positives. User 2 takes the middle ground, being satisfied with a sensitivity of 48%. Each user has set

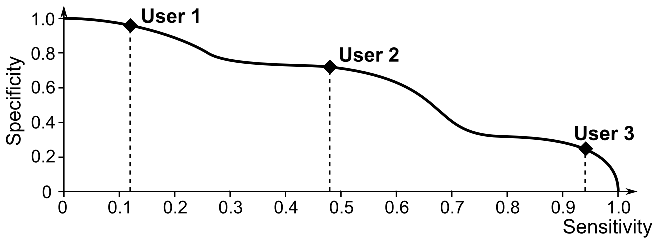


Fig. 1. User 1 is happy with few correct matches, but is easily irritated by false positives. User 3 requires most truly similar textures to be provided and can tolerate many false positives. User 2 strikes a balance between these extremes.

a threshold on sensitivity and wishes specificity to be maximized subject to this constraint. Users may state their requirements in different ways, e.g. via estimates of the relative costs of false positives and false negatives, yet clearly we wish the graph to be as high as possible at each value of sensitivity.

Conceptually, this results in uncountably many objectives: to please each potential user of our search engine we should maximize specificity for each value of sensitivity. In practice, however, the graph of specificity against sensitivity is piecewise horizontal, with the number of pieces bounded by the number of records in the class of interest. This reduces the number of objectives to 'very many'. The height of neighbouring points on the graph are also highly correlated, which increases the chance that any pair of feature sets are comparable, i.e. that one dominates the other. Finally, section 4 introduces modified dominance relations that further increase the probability that a pair of solutions are comparable, enabling an effective dominance-based approach to this problem. Meanwhile, note that the methods developed here can handle the conceptually infinite-objectives case — the resulting dominance relations and crowding measures are suitable for the comparison of graphs, rather than vectors of objectives.

There are many approaches to evaluating feature subsets; in this paper we examine two. In each case, an objective is the value of some measure of quality at a fixed value of some other quality measure or parameter. The first approach plots specificity against sensitivity (equivalent to optimizing ROC curves [5]); the second plots confidence against sensitivity. In either case, if a threshold value produces a sensitivity-specificity pair or a sensitivity-confidence pair that is dominated by some other pair, it is not considered part of the curve produced.

4 Dominance and Crowding

The basic dominance relation is familiar: one solution dominates another if it is at least as good on all objectives and better on at least one. In the case of specificity vs. sensitivity curves, this translates to the curve for the first solution being at least as high as the curve for the second in all places, and higher in some. While this seems reasonable, there is a concern that the large number of objectives will result in a weak dominance relation. (Here, a dominance relation

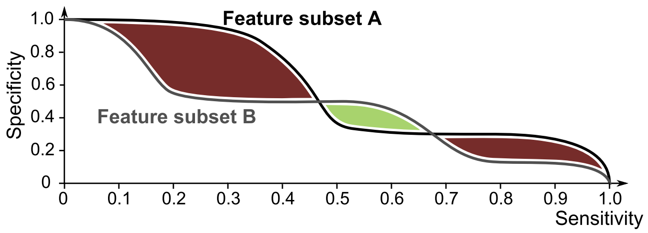


Fig. 2. Dividing the area of dark red by the area of pale green gives a value used by the modified dominance relation

is considered 'strong' if, given a random pair of solutions, the probability that they are comparable is high.) Solution B need only beat solution A over a tiny portion of the curve in order to avoid being dominated by A. This may result in a lack of selection pressure in dominance based algorithms such as NSGA II [3] and a potentially unmanageable number of non-dominated solutions. Hence we apply a simple modification to the dominance relation. In Fig. 2 the area in dark red is dominated by feature set A but not by feature set B, while the pale green area is dominated by B but not A. The modified dominance relation states that A dominates B if there is a red area but no green area, or if the result of dividing the red area by the green area exceeds a given *dominance factor*. A dominance factor of 1 makes almost every pair of solutions comparable, reducing to the problem of maximizing the area under the curve — a commonly used measure of the performance of a machine learning algorithm (e.g. [2]). At the other extreme, an infinite dominance factor produces the basic dominance relation.

Any well-behaved dominance relation should be anti-symmetric and transitive. If the dominance factor is at least 1, then the relation is clearly anti-symmetric — if the red area is bigger than the green area then the green area cannot be bigger than the red area. Transitivity can also be shown, though this requires a little more work. A brief proof is available in supplementary material at `http://www.macs.hw.ac.uk/~ar136`, along with our code and datasets.

Finally, to underpin maintenance of a limited-size archive of non-dominated solutions, we consider the choice of 'crowding' measure. It is possible to generalize the standard crowding measure of NSGA II [3], but we elect to use a crowding measure based on distances between pairs of solutions. We define this as the area between the two curves, i.e. the sum of the red and green areas in Fig. 2.

5 Implementation

Two classification algorithms, logistic regression (LR) and naive Bayes (with Laplace correction) (NB) [10], are used to evaluate the feature subsets. LR requires numeric data, so categorical data were converted. For example, a categorical field with three categories, cyan, magenta and yellow, is converted into two numeric fields, taking the values zero and one. A one in the first field translates to 'cyan', while a one in the second field translates to 'magenta'. Two zeroes

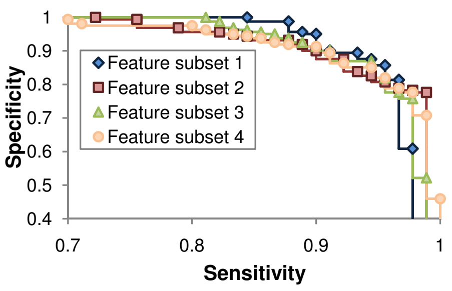


Fig. 3. Quality plots for four feature subsets, generated from the ionosphere data with dominance factor set to 2 and a limit of 4 features per subset

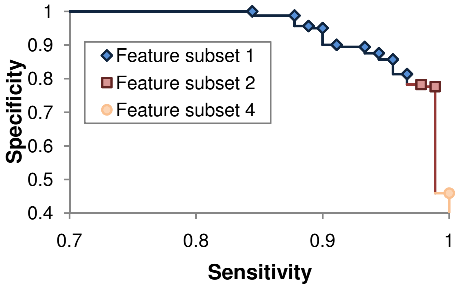


Fig. 4. Plotting the envelope of the quality plots for four feature subsets. Notice that feature subset 3 does not contribute, despite being non-dominated.

imply that the record is 'yellow'. In contrast, our implementation of NB requires categorical data. Any numeric field is discretized by partitioning its range into a number of bins. To avoid problems that may arise from a highly non-uniform distribution over the bins, we aim for an equal-frequency discretization. (Details may be found in the supplementary material.)

Feature subsets were optimized using NSGA II [3], with the dominance relation and crowding measure replaced by those described above. Solutions were encoded as bitstrings, with bits indicating the presence or absence of the corresponding feature. A limit on the number of features was imposed and enforced after crossover by removing random features as necessary. Three types of mutation were used at equal rates: addition of a feature (if permitted), removal of a feature or swapping a feature in the subset for one currently absent.

It has been suggested that dominance based algorithms such as NSGA II perform poorly for problems with more than 4 objectives [6], most likely due to the resulting weak dominance relation. Here we illustrate that despite the large number of objectives, good results can be obtained if the dominance relation is strong enough.

Finally, we note that, after performing the optimization, the presence of so many objectives raises issues with the presentation of the results. Figure 3 shows quality plots for four non-dominated feature subsets. Their quality may be compared without too much difficulty using the figure. However, this task becomes much more difficult given twenty or thirty non-dominated solutions.

One possibility is to plot only those points that are non-dominated with regard to sensitivity and specificity, producing Fig. 4. Notice that feature subset 3 does not contribute to the sensitivity-specificity front. However, this subset may be the best choice, since it performs well over all values of sensitivity. Moreover, this method of presenting results may result in an overoptimistic view of certain feature subsets on unseen data. An alternative approach is to present two types of graph. The main graph plots solutions according to, e.g., the specificity at two different values of sensitivity. Selecting a solution produces a second graph of specificity against sensitivity for the solution including two markers, in this case vertical lines, that indicate the objectives used on the main graph. With a suitable user interface, dragging these markers changes the objectives used on the main graph, allowing the user to fully explore the solution set.

6 Experimentation

We explore this approach to generating multiple feature subsets by testing our algorithm on three datasets. In each case the result is a Pareto front of feature subsets, each of which has its own characteristic tradeoff curve (e.g. specificity vs. sensitivity). In evaluating the technique, we are constrained by the fact that there are as yet no suitable alternative algorithms in the literature that address the same problem. The closest is the approach of Emanouilidis [4]. However, we maximize specificity for each possible value of sensitivity. Emanouilidis's use of 1-NN as the core classifier means that only a single sensitivity-specificity pair is obtained for each feature set and it is these single values of sensitivity and specificity that are maximized. Hence the algorithms optimize different measures of feature set quality. (Note that we can compare the best sensitivity-specificity values obtained by the two approaches, where we might expect Emanouilidis's use of 1-NN to restrict the spread of solutions across the sensitivity-specificity front.) Evaluation of our approach is therefore restricted mainly to illustrating that it achieves apparently effective results on the three datasets studied, in each case yielding a set of feature subsets with varied performance characteristics. Beyond this, we also report on aspects of performance that vary according to the dominance factor, and according to constraints on the size of feature subsets.

In all experiments, we used a crossover rate of 0.8, a mutation rate (the chance that a solution is mutated) of 0.2, a population size of 100, 500 generations, and 10-bin discretization when NB was used as the core classifier. Each dataset is split into training and test sets, used during optimization and final evaluation respectively. Whenever a feature subset is evaluated on the training or test set, cross-validation (CV) is used — leave-one-out-CV for the ionosphere data, and 5-fold CV in the other two cases. Dataset details are as follows:

Ionosphere: Available from [1] and used previously for MOFS [4], the ionosphere data comprises 351 records and 35 fields. The class field is either "good" (g) or "bad" (b), where "bad" is the class of interest. Non-class fields are numeric. The dataset was split into training and test data, with the test set containing 100 records, 36 in the class of interest.

Breast Cancer Wisconsin (Diagnostic): Again from [1], this dataset has a class field that takes the value 'M' (malignant) or 'B' (benign) and has 30 numeric input fields. The class of interest is the malignant class. The 569 records, 212 in the class of interest, are divided randomly into training and test sets, with the test set containing 169 records, including 63 in the class of interest.

Texture: In the texture data, each record corresponds with a pair of textures. 5376 numerical features were extracted computationally from a set of textures, using a range of methods including spectrum analysis, radon transforms, autocorrelation etc. This was reduced to 283 features using simple correlation-based methods. The input fields were obtained by calculating feature differences for each texture pair. The class field was obtained by asking 30 people to group similar textures, given either the full set of textures or a subset. Two textures were considered similar if at least a third of people grouped the pair together. The training set involved 19900 texture pairs (200 textures), 333 of which were considered similar. The test set was produced using another 100 textures, generating 4950 records, 85 in the class of interest.

The texture data is much larger than the other datasets, providing a more challenging test. The class of interest forms only a small part of the dataset, making it difficult to make true positive predictions without introducing many false positives, i.e. it is difficult to achieve high confidence values.

7 Results

First, to examine the effect of the dominance factor and to determine a suitable value for this parameter, experiments were performed on the ionosphere data using an upper limit of 4 features per feature subset. NB was used to evaluate feature sets, with sensitivity-specificity curves used to determine feature set quality. The time taken by the algorithm and the numbers of non-dominated solutions obtained, averaged over 30 runs, are outlined in Table 1. The last column corresponds with the use of the basic dominance relation.

Note that the number of non-dominated solutions is small compared with the number of solutions examined, even when the basic dominance relation is used. This implies that the dominance relation is stronger than one might expect for a problem with so many objectives. However, given the difficulty in comparing 397 feature subsets, modified dominance is used in the following experiments.

On the Ionosphere data, the algorithm was applied using NB, sensitivity-specificity curves and a dominance factor of 5. Runs were performed with different limits on the number of features. Table 2 shows the number of non-dominated

Table 1. The effect of modifying the dominance factor

Dominance factor	1	2	5	10	20	50	∞
No. non-dominated	1.00	3.77	16.0	41.9	85.7	187	397
Time (s)	20.7	23.0	23.9	25.5	26.9	28.4	20.3

Table 2. Number of non-dominated solutions for different feature set size limits

Max. features	1	2	3	4	5	6	7	8
No. non-dominated	7.00	19.0	16.0	16.0	36.0	32.4	70.0	173
Time (s)	11.0	19.6	20.1	23.6	26.1	27.5	30.7	34.2

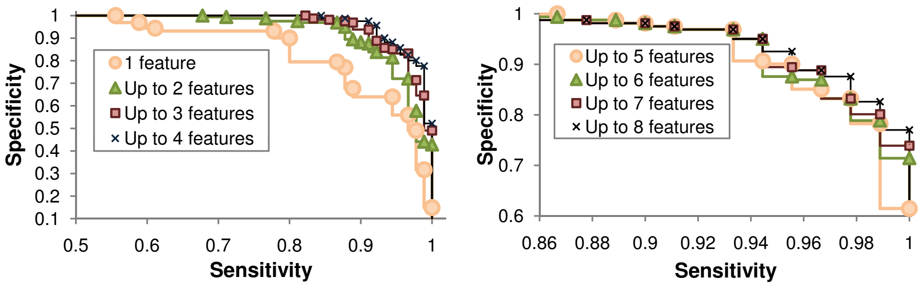


Fig. 5. Performance on Ionosphere training data

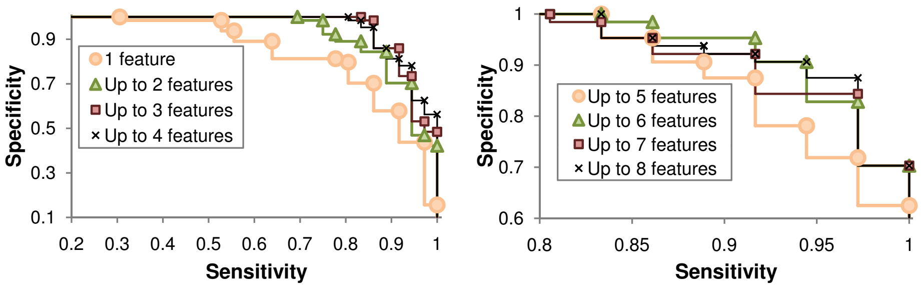


Fig. 6. Performance on Ionosphere test data

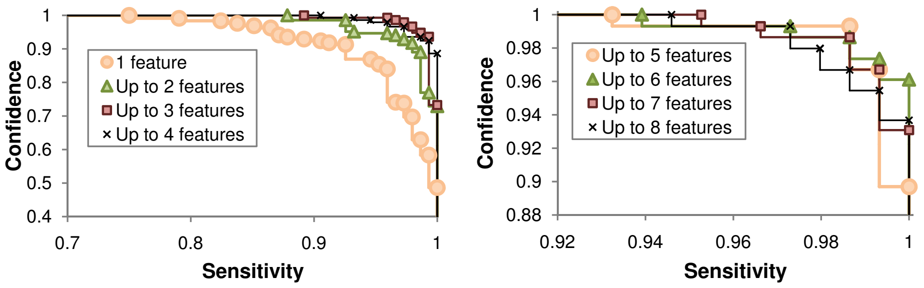


Fig. 7. Performance on Breast Cancer Wisconsin (diagnostic) training data

solutions obtained and time requirements, averaged over 30 runs. Time taken was sufficient to find all Pareto-optimal solutions for subsets of up to 5 features, in all runs. Results on training and test data are shown in Figs. 5 and 6 respectively. (For clarity, Figs. 5–9 show the envelope of the sensitivity-specificity curves.)

Comparing with [4], the most notable difference is that the results presented in Figs. 5 and 6 cover a much broader range of sensitivity-specificity values, since the classification algorithm used is capable of effectively evaluating feature sets that perform well at either high sensitivity or high specificity values.

On the breast cancer data, the algorithm was applied using LR, sensitivity-confidence curves and a dominance factor of 5. Typical time requirements were 23

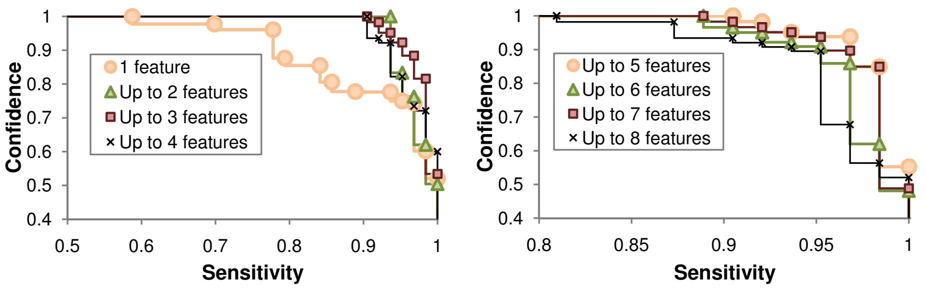


Fig. 8. Performance on Breast Cancer Wisconsin (diagnostic) test data

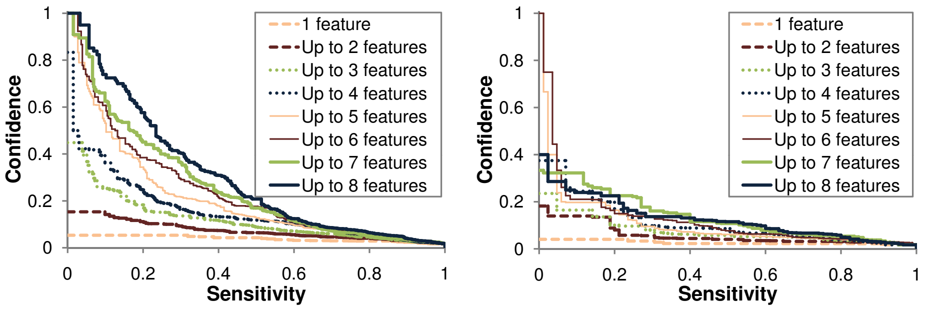


Fig. 9. Performance on texture data, for one to eight features; results on training (left) and test (right) data

min. for four features and 32 min. for eight. Results are shown in Figs. 7 and 8. Finally, on the texture data the algorithm was applied using NB, sensitivity-confidence curves and a dominance factor of 2. Typical time requirements were 23 min. for four features and 29 min. for eight. Results are in Fig. 9.

8 Discussion

This paper has shown that FS can be effectively treated as a MO optimization problem with an infinite set of objectives. The approach has advantages over other FS methods, in that each feature subset generated is evaluated across a range of values of sensitivity. There are many possible avenues of further research. For example, if one is only interested in the feature sets that contribute to the overall sensitivity-specificity (or sensitivity-confidence) front, an alternative approach to dominance is indicated. Meanwhile, the class field in the texture data originally indicated the proportion of people that considered a pair of textures to be similar. This can be considered as the 'probability of class membership'. Here we used a threshold to convert this into a binary field. However, research should be performed into adapting the approach of this paper to probabilistic class membership. Finally, an obvious avenue of further work is to generalize the method to multi-class problems. For example, given a three class problem we may evaluate feature subsets according to the accuracy on each class. So rather than dealing with curves and the area between curves, the problem becomes one of surfaces and the volume between surfaces.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
2. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997)
3. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.P. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
4. Emmanouilidis, C.: Evolutionary multi-objective feature selection and ROC analysis with application to industrial machinery fault diagnosis. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailiou, K., Fogarty, T. (eds.) Evolutionary Methods for Design, Optimisation and Control. CIMNE (2002)
5. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27, 861–874 (2006)
6. Hughes, E.J.: Evolutionary many-objective optimisation: Many once or one many? In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 222–227. IEEE Service Center, Los Alamitos (2005)
7. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Feature selection using multi-objective genetic algorithms for handwritten digit recognition. In: Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002), vol. 1, pp. 568–571. IEEE Computer Society, Los Alamitos (2002)
8. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: A methodology for feature selection using multi-objective genetic algorithms for handwritten digit string recognition. International Journal of Pattern Recognition and Artificial Intelligence 17(6), 903–929 (2003)
9. Pappa, G.L., Freitas, A.A., Kaestner, C.A.A.: A multiobjective genetic algorithm for attribute selection. In: Proceedings of the 4th International Conference on Recent Advances in Soft Computing (RASC 2002), pp. 116–121 (2002)
10. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Incorporating Domain Knowledge into Evolutionary Computing for Discovering Gene-Gene Interaction

Stephen D. Turner, Scott M. Dudek, and Marylyn D. Ritchie

Center for Human Genetics Research, Department of Molecular Physiology & Biophysics, Vanderbilt University, Nashville, TN, USA
{stephen,dudek,ritchie}@chgr.mc.vanderbilt.edu

Abstract. Understanding the genetic underpinnings of common heritable human traits has enormous public health benefits with implications for risk prediction, development of novel drugs, and personalized medicine. Many complex human traits are highly heritable, yet little of the variability in such traits can be accounted for by examining single DNA variants at a time. Seldom explored non-additive gene-gene interactions are thought to be one source of this "missing" heritability. Approaches that can account for this complexity are more aptly suited to find combinations of genetic and environmental exposures that can lead to disease. Stochastic methods employing evolutionary algorithms have demonstrated promise in being able to detect and model gene-gene interactions that influence human traits, yet the search space is nearly infinite because of the vast number of variables collected in contemporary human genetics studies. In this work we assess the performance and feasibility of sensible initialization of an evolutionary algorithm using domain knowledge.

Keywords: Neural networks, grammatical evolution, gene-gene interaction, quantitative traits, domain knowledge.

1 Introduction

1.1 Genome-Wide Association Studies, Complex Disease, and Epistasis

The genome-wide association study (GWAS) is a commonly employed technique in human genetics research to investigate DNA variations associated with common human diseases. Several technologies are currently available that allow for rapid, highly accurate genotyping of >1 million common single nucleotide polymorphisms (SNPs) at low cost per genotype. We have yet to fully explore the abundance of data generated by these studies in part because maturation of our analytical strategies for data of this scale have not kept pace. The most commonly used analytical procedures for analyzing genetic data are very simple tests of association looking at one genetic variant (SNP) at a time. This approach has been somewhat successful in identifying genetic variants associated with complex traits, including age-related macular degeneration, type II diabetes, hypertension, and blood cholesterol levels, among others [1]. However, these single SNPs collectively explain little of the genetic contribution to the trait variance that is expected based on family and twin studies [2].

For instance, HDL-cholesterol level is highly genetic – up to 73% of variation in HDL can be explained by genetic factors [3] – yet even the most highly powered studies found that collectively only ~5% of this variance could be accounted for by single-SNP analysis [4]. Many agree that a portion of this "missing heritability" likely lies in gene-gene and gene-environment interactions [2;5], and it is well accepted that common traits are complex, and likely influenced by an elaborate interplay of multiple genetic and environmental factors [6-8]. Moreover, recent perspectives have emphasized that most true single locus genetic associations to complex traits carry a vanishingly small effect size [9], and experimental data from model organisms illustrates that gene-gene interaction is pervasive and often carries surprisingly large effects [10;11].

Several approaches to gene-gene interaction analysis include testing interactions between variants with statistically significant main effects [12], based on biological criteria [13], or exhaustively testing all possible interactions. Using biological criteria or statistical significance of main effects to guide an interaction analysis imposes severe limitations on the search for gene-gene interactions, and exhaustive interaction analysis is often computationally prohibitive in large GWAS datasets. This is the motivation for developing techniques that still utilize the full dimensionality of the data without exhaustively searching all possible combinations of variables with the goal of discovering a well-fitting model that explains variance in an outcome of interest.

1.2 Grammatical Evolution Neural Networks (GENN) and Domain Knowledge

Neural networks (NNs) are a robust and flexible modeling technique that attempt to mimic the basic structure and function of biological neurons to solve complex problems. NNs have been applied to many research fields, including robotics, speech recognition, optical character recognition, task scheduling, industrial processing, and to many problems in biological science, including microarray data analysis, genotype calling, human linkage analysis, genetic association studies, medical expert systems, survival analysis, and protein folding [14]. The conventional approach for applying NNs to a classification problem is to specify a network architecture, select which variables are included as inputs to the network, and fit network weights using a gradient-descent based approach such as backpropagation [15]. Recently, numerous evolutionary search strategies have been applied to NN classification problems to reduce the issues associated with the traditional NN approach. Genetic Programming Neural Networks [16] and Grammatical Evolution Neural Networks (GENN) [17] use genetic programming [18] or grammatical evolution (GE) [19] to evolve populations of neural networks for human genetics classification problems. These populations are a heterogeneous mix of architectures, weights, and input variables which undergo mating, crossover, and recombination to ultimately identify an optimum NN solution. Recent work has shown that certain features characteristic of human genetic data may provide advantages to methods that evolve NNs to detect gene-gene interactions by transforming the fitness landscape from a "needle in a haystack" to a broader, smoother surface [20].

The application of GE to find epistatic gene-gene interactions is still exceedingly difficult, especially when the underlying disease model is purely epistatic, where each variant has no independent effect on the phenotype [21]. After demonstrating the critical need for expert knowledge when applying genetic programming to GWAS [22], others have shown that using expert knowledge guided mutation, selection, and crossover is highly beneficial, and dramatically improves the performance of evolutionary algorithms [23;24]. In much of the previous work showing that expert knowledge increases the performance of natural computing algorithms for finding epistatically interacting SNPs, the statistical expert knowledge was gleaned *intrinsically* – typically using a data-driven approach using variants of the Relief algorithm for feature selection [24-26]. Our goal here was to evaluate with simulation whether biological domain knowledge obtained *extrinsically* would increase GENN's performance in discovering epistatic interactions between genetic variants contributing to a quantitative trait outcome. Here we present results of a simulation study showing that incorporating biological knowledge from external sources results in a modest increase in GENN's ability to detect and model gene-gene interactions among a large pool of unassociated noise variables.

2 Methods

2.1 Genetic Data Simulation with genomeSIMLA

Simulated data where the true identity and size of the genetic or environmental effect in the population is known is a necessity for developing and testing novel methodology. We recently developed genomeSIMLA [27] for simulating genome-wide scale data in population based case-control samples with a categorical outcome. Here we use an extension of genomeSIMLA capable of simulating gene-gene interactions in the presence of main effects, all of which influence a quantitative trait at a desired effect size [28].

The effect size of a genetic variant on a quantitative trait outcome is often expressed in terms of heritability – the proportion of variance in the trait explained by genetic variation. The narrow-sense heritability, here defined as the proportion of variance explained uniquely by a single source of genetic variation (e.g. the main effect of one member of an interacting pair of variants) is given by the semi-partial squared correlation coefficient in the equation below, where the first term represents the proportion of variance in the outcome explained by all sources of genetic variation currently modeled, and the second term represents the proportion of variance explained when a particular variable (*i*) is removed from the model [29]:

$$sr_i^2 = R^2_{Y.1,2,...i...k} - R^2_{Y.1,2,...(i)...k}, \tag{1}$$

Datasets are simulated using a linear regression equation where the genetic model can take a range of generally additive models, similar to the method implemented in [30] for a discrete outcome. Here we simulated a quantitative trait under additive and dominant interaction models as shown in Figure 1. In the additive model, the mean value of the simulated quantitative trait increases as a function of the number of copies of the less common allele an individual inherits both within and between the two functional genetic variants. In the dominant model, the mean value of the simulated trait is increased if individuals contain at least one or more copies of the minor allele.

Individuals are drawn from a homoscedastic normal distribution with the mean being determined by the genotypes at the corresponding functional genetic variants. The three different genotypes for each variant are represented as -1, 0, 1.

We simulated 500 SNPs in 2000 individuals, where only two SNPs were functional and the other 498 SNPs were unassociated "noise" variables. We simulated a gene-gene interaction between these two SNPs that carried a narrow-sense heritability (h^2) of 0.05, meaning that only 5% of the variation in the quantitative trait could be explained by this gene-gene interaction. This low effect size is typical of most findings in human genetic epidemiology [9]. We simulated this interaction in the context of very minimal main effects at each locus (h^2=0.01). A scenario such as this where main effects explain little of the overall outcome variance represents a very difficult problem for an evolutionary search procedure to model.

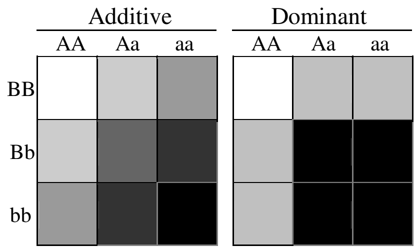


Fig. 1. Genetic model types simulated. The less common allele ("a" or "b") increases the value of a simulated quantitative trait by an amount based on type of genetic model (additive or dominant) and the number of copies of the less common allele an individual possesses. Darker cells indicate a higher mean value for the simulated trait. The genetic model type applies to both main effects and interactive effects within and between the two functional variants.

2.2 Domain Knowledge

A recently developed tool called Biofilter is capable of integrating information from several publicly available biological databases in order to assess specific combinations of genetic variations and their effect on the outcome based on prior statistical and biological knowledge [31]. Biofilter draws expert knowledge from databases containing protein family, gene ontology, and biological pathway information in order to construct two-SNP models that are supported by the biological literature. Their degree of support in the literature is characterized by an implication index – which is a count of how many times that two-SNP model appears across multiple databases incorporated into Biofilter.

To determine whether incorporation of domain knowledge into GENN training can improve its performance, simulated domain knowledge that mimics information obtained from Biofilter must be generated. Here, 4000 random undirected edges are drawn between a subset of the 500 SNPs simulated as described above. The implication index is the number of edges drawn between two models. This number typically ranges from 0 to 5, where implication index of zero indicates no support in the simulated knowledge pool, while an implication score of 5 indicates that this model is very well supported. The implication index corresponding to the functional two-SNP model where the true effect was embedded could be manually specified. Our specific goals were to determine if and to what degree GENN's performance would diminish if irrelevant domain knowledge were incorporated, and if and to what degree GENN's performance would increase if accurate domain knowledge were incorporated into the training process.

2.3 GENN and Incorporation of Domain Knowledge

GENN has been implemented as previously described [17]. Briefly, grammatical evolution (GE) is a variation of genetic programming (GP), an evolutionary algorithm originally proposed by Koza as a procedure to optimize NN architecture [18]. In GE, randomly initialized binary strings are transcribed into an ordered list of integers which are used to select from production rules in a Backus-Naur form grammar. Our grammar applies GE to construct neural networks, and can simultaneously select important predictor variables and optimize network weights and architecture.

Domain knowledge was used to perform sensible initialization. Rather than initializing a population of NNs randomly, the initial generation is partially composed of NNs containing as input variables SNPs that are represented in a domain knowledge source. This source can be two-SNP models supported by biological literature derived from Biofilter [31] or, as in this study, simulated domain knowledge which mimics domain knowledge derived from Biofilter. Part of the population is still initialized randomly. Here the proportion of the initial population which is initialized from domain knowledge was varied from 0 to 99% in intervals between 1-10%. Two-SNP models from domain knowledge are prioritized for incorporation in the initial generation based upon implication index – models with higher implication index are initialized first. The implication index on the functional two-SNP model in these experiments ranged from 0 (negative control – all domain knowledge incorrect/irrelevant) to 3 (functional two-SNP model is somewhere in the top half of the implication index-ranked list of 4000 domain knowledge two-SNP models). The implication index could have been raised even higher, but this would have guaranteed that the functional model would have been very well supported, and would represent an overly optimistic scenario.

Based on prior work [28], GENN was run for 200 generations using 10 demes of 100 individuals. This took approximately 6 minutes per dataset on five 1.8 GHz Opteron PCs. The respective probability of crossover and mutation were 0.9 and 0.01, typical values for these parameters in many genetic algorithms [32]. Addition was the only production rule available for the arithmetic operator at each activation node, as described previously [33]. This allowed for the implementation and optional usage of backpropagation (BP), a local fitting procedure designed to optimize the weights in a neural network [15]. This hybrid algorithm allows for weight optimization in the event that sensible initialization from domain knowledge resulted in the inclusion of either of the two functional variables in the initial generation. BP was either not used at all, or used at initialization and again at generations 100 and 200, using a learning rate of 0.3. BP was halted after either a maximum of 100 epochs had been run, or when further BP showed no improvement, after which the GE process continues. Networks undergoing BP were reverted back to a binary genome by marking blocks of codons corresponding to a weight, which was then replaced with a block containing a grammar compatible block that generates the appropriate weight when GE continues after BP.

3 Results

For the simulation study described above, sensitivity was measured as the proportion of datasets out of 100 simulated datasets, where the best performing neural network model contained the two functional SNPs, with no other SNPs in the model, i.e. a perfect match. The best neural network model for each dataset was chosen by maximizing cross-validation consistency [8;34]. In case of a tie (e.g. two different

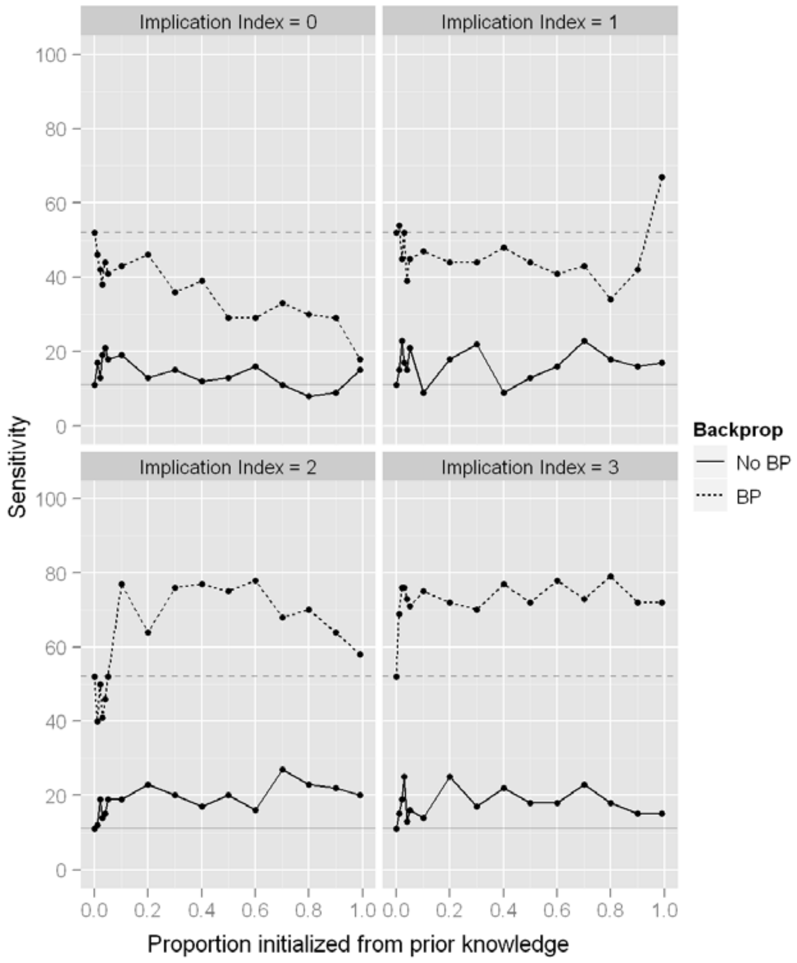


Fig. 2. Sensitivity of GENN to detect both functional SNPs as the proportion of SNPs initialized from domain knowledge increases from 0 to 99%. Panels show the implication index of the model that includes both functional variables. Solid line shows when GE alone was used to train NNs (no BP). Dashed line shows sensitivity when using the hybrid BP-GENN algorithm (see methods). Faint horizontal solid and dashed lines show for reverence the baseline sensitivity, for GENN and BP-GENN, when the population was initialized randomly, i.e. 0% initialized from domain knowledge.

models replicated across two CV intervals), the model with the higher R^2 (proportion variance explained) is deemed the overall best model. The results are summarized in Figure 2.

The results here show that sensitivity to detect both genetic variants contributing to the trait is always higher when BP was used in conjunction with GE. When the implication index is 0 (i.e. all domain knowledge is irrelevant), the sensitivity when using BP decreases substantially as the proportion of the initial population initialized from domain knowledge increases (upper left panel of Fig. 2, dashed line). This is likely due to the fact that as more NN models are initialized from a list of models from irrelevant domain knowledge, there is a smaller chance that either of the functional variables can be initialized by chance. When the implication index is at least 1 (meaning the functional two-SNP model is supported in our domain knowledge), as this proportion increases, sensitivity fluctuates around the baseline sensitivity (37%) at random initialization when BP is not used. This is not surprising, because even if a NN is initialized containing both functional variables which influence the trait, it is unlikely that by chance the NN would have suitable weights and architecture. An increase in performance can be seen when BP is then used to optimize the weights in the sensibly initialized NNs from relevant domain knowledge. Furthermore, as the implication index for the domain knowledge model containing the functional variables increases from 1 to 3, this model is more likely to be incorporated into NNs in the initial generation. For instance, when the implication index of the functional model is 1, approximately 99% of the population must be initialized from domain knowledge in order to see any benefit. When the implication index is 2 or higher, it is very likely that the initial generation will contain a NN with the truly functional variables even when only a small proportion of the initial population is initialized from domain knowledge.

4 Discussion

The results presented here show that the sensitivity of using GE to train NNs to find genes with a nonlinear influence on a quantitative outcome can be improved by effectively using extrinsic domain knowledge. We showed that initializing a proportion of the NN population from domain knowledge when BP is employed to locally optimize the weights in a NN can result in a modest improvement in GENN's ability to detect and model SNPs influencing a simulated trait (Figure 2). We also performed the same experiment shown in Figure 2 with two other population sizes: one larger (200 individuals) and one smaller (50 individuals) (data not shown). While sensitivity was higher when using a larger population size, the benefit of utilizing domain knowledge to initialize NNs was less striking. When a smaller population size was used, the benefit was even more noticeable, although absolute sensitivity was lower. This indicates that when the search space is small enough to be searched very throroughly or exhaustively, using domain knowledge is less beneficial than when the search space is very large compared to the number of individual solutions being evolved. In this scenario (such is the case in genome-wide association studies), using domain knowledge to bias an evolutionary search in favor of important features will be critical for acceptable performance.

While the benefits are not as striking as when using intrinsically obtained statistical expert knowledge [23;24], using this framework to initialize an evolutionary search for disease genes based on domain knowledge obtained from public biological databases is another means to improve the performance of genetic algorithms for feature selection. Supplementing an evolutionary search using domain knowledge will be critical if using evolutionary procedures to find and model the effect of disease genes on complex human traits. Natural, biological data will likely have many effects which will be enriched in knowledge sources, resulting in an improvement of the overall ability to find many members in the collection of influential loci. It is clear that there are more fruitful approaches for understanding the genetic architecture of common human phenotypes than ignoring the complexity of biology by testing single variants in isolation [35]. One of the strengths of the method presented here is that if any arbitrarily complex interaction of genetic and environmental exposures influences disease risk, a NN can approximate this function [36], given proper training. These experiments show that incorporating domain knowledge into an evolutionary algorithm for finding genes related to disease can aid the variable selection process if the domain knowledge is consistent with reality.

One limitation in the current study is that these experiments make the assumption that loci involved in a gene-gene interactions contributing to a heritable trait will carry with them some small main effect at either variant. There are, however, few examples of a consistently replicating, experimentally verified gene-gene interaction in the complete absence of main effects contributing to a complex quantitative trait in humans. Perhaps the reason for this, however, is the inadequacy of our methods for finding gene-gene interactions in the absence of main effects rather than the absence of such effects altogether. Future studies should aim to assess these and other extensions of GENN in their ability to detect and model epistatic interactions contributing to a quantitative trait in the absence of main effects, and should attempt to apply these methods in a natural, biological data analysis.

Acknowledgements

This work was supported by NIH grants NS066638, LM010040, and HG004608.

References

1. Hindorff, L.A., Sethupathy, P., Junkins, H.A., Ramos, E.M., Mehta, J.P., Collins, F.S., Manolio, T.A.: Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. Proc. Natl. Acad. Sci. USA 106(23), 9362–9367 (2009)
2. Maher, B.: Personal genomes: The case of the missing heritability. Nature 456(7218), 18–21 (2008)
3. Pietilainen, K.H., Soderlund, S., Rissanen, A., Nakanishi, S., Jauhiainen, M., Taskinen, M.R., Kaprio, J.: HDL subspecies in young adult twins: heritability and impact of overweight. Obesity (Silver. Spring) 17(6), 1208–1214 (2009)
4. Kathiresan, S., et al.: Common variants at 30 loci contribute to polygenic dyslipidemia. Nat. Genet. 41(1), 56–65 (2009)

5. Manolio, T.A., et al.: Finding the missing heritability of complex diseases. Nature 461(7265), 747–753 (2009)
6. Wright, S.: The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In: Proc. 6th Intl. Congress of Genetics, vol. 1, pp. 356–366 (1932)
7. Moore, J.H., Williams, S.M.: Traversing the conceptual divide between biological and statistical epistasis: systems biology and a more modern synthesis. Bioessays 27(6), 637–646 (2005)
8. Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. Am. J. Hum. Genet. 69(1), 138–147 (2001)
9. Goldstein, D.B.: Common Genetic Variation and Human Traits. N. Engl. J. Med. 360(17), 1696–1698 (2009)
10. Shao, H., et al.: Genetic architecture of complex traits: large phenotypic effects and pervasive epistasis. Proc. Natl. Acad. Sci. USA 105(50), 19910–19914 (2008)
11. He, X., Qian, W., Wang, Z., Li, Y., Zhang, J.: Prevalent positive epistasis in Escherichia coli and Saccharomyces cerevisiae metabolic networks. Nat. Genet. 42(3), 272–276 (2010)
12. Kooperberg, C., Leblanc, M.: Increasing the power of identifying gene x gene interactions in genome-wide association studies. Genet. Epidemiol. 32(3), 255–263 (2008)
13. Carlson, C.S., Eberle, M.A., Kruglyak, L., Nickerson, D.A.: Mapping complex disease loci in whole-genome association studies. Nature 429(6990), 446–452 (2004)
14. Turner, S.D., Crawford, D.C., Ritchie, M.D.: Methods for optimizing statistical analyses in pharmacogenomics research. Expert Reviews in Clinical Pharmacology 2(5), 559–570 (2009)
15. Bishop, C.M.: Neural Networks for Pattern Recognition, pp. 475–482. Oxford University Press, London (1995)
16. Ritchie, M.D., Coffey, C.S., Moore, J.H.: Genetic Programming Neural Networks as a Bioinformatics Tool for Human Genetics. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 438–448. Springer, Heidelberg (2004)
17. Motsinger-Reif, A.A., Dudek, S.M., Hahn, L.W., Ritchie, M.D.: Comparison of approaches for machine-learning optimization of neural networks for detecting gene-gene interactions in genetic epidemiology. Genetic Epidemiology 32(4), 325–340 (2008)
18. Koza, J., Rice, J.: Genetic generation of both the weights and architecture for a neural network. IEEE Transactions II (1991)
19. O'Neil, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language, 1st edn. Kluwer Academic Publishers, Norwell (2003)
20. Turner, S.D., Ritchie, M.D., Bush, W.S.: Conquering the Needle-in-a-Haystack: How Correlated Input Variables Beneficially Alter the Fitness Landscape for Neural Networks. In: Pizzuti, C., Ritchie, M.D., Giacobini, M. (eds.) EvoBIO 2009. LNCS, vol. 5483, pp. 80–91. Springer, Heidelberg (2009)
21. White, B.C., Gilbert, J.C., Reif, D.M., Moore, J.H.: A statistical comparison of grammatical evolution strategies in the domain of human genetics. In: Proceedings of the IEEE Congress on Evolutionary Computing, pp. 676–682 (2005)
22. Moore, J.H., White, B.C.: Genome-wide genetic analysis using genetic programming: The critical need for expert knowledge. Genetic Programming Theory and Practice 4, 11–28 (2007)
23. Moore, J.H., Barney, N., White, B.C.: Solving complex problems in human genetics using genetic programming: The importance of theorist-practitioner-computer interaction. Genetic Programming Theory and Practice 5, 69–85 (2008)

24. Greene, C.S., White, B.C., Moore, J.H.: An expert knowledge-guided mutation operator for genome-wide genetic analysis using genetic programming. In: Rajapakse, J.C., Schmidt, B., Volkert, L.G. (eds.) PRIB 2007. LNCS (LNBI), vol. 4774, pp. 30–40. Springer, Heidelberg (2007)
25. Moore, J.H., Andrews, P.C., Barney, N., White, B.C.: Development and Evaluation of an Open-Ended Computational Evolution System for the Genetic Analysis of Susceptibility to Common Human Diseases. In: Marchiori, E., Moore, J.H. (eds.) EvoBIO 2008. LNCS, vol. 4973, pp. 129–140. Springer, Heidelberg (2008)
26. Greene, C.S., Gilmore, J., Kiralis, J., Andrews, P.C., Moore, J.H.: Optimal Use of Expert Knowledge in Ant Colony Optimization for the Analysis of Epistasis in Human Disease. In: Pizzuti, C., Ritchie, M.D., Giacobini, M. (eds.) EvoBIO 2009. LNCS, vol. 5483, pp. 92–103. Springer, Heidelberg (2009)
27. Edwards, T.L., et al.: Generating Linkage Disequilibrium Patterns in Data Simulations Using genomeSIMLA. In: Marchiori, E., Moore, J.H. (eds.) EvoBIO 2008. LNCS, vol. 4973, pp. 24–35. Springer, Heidelberg (2008)
28. Turner, S.D., Dudek, S.M., Ritchie, M.D.: Grammatical Evolution of Neural Networks for Discovering Epistasis among Quantitative Trait Loci. In: Pizzuti, C., Ritchie, M.D., Giacobini, M. (eds.) EvoBIO 2010. LNCS, vol. 6023, pp. 86–97. Springer, Heidelberg (2010)
29. Cohen, P., et al.: Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences, 3rd edn. Lawrence Erlbaum, Philadelphia (2002)
30. Schmidt, M.A., Hauser, E.R., Martin, E.R., Schmidt, S.: Extension of the SIMLA Package for Generating Pedigrees with Complex Inheritance Patterns: Environmental Covariates, Gene-Gene and Gene-Environment Interaction. Statistical Applications in Genetics and Molecular Biology, Article 15, 4(1), 1–21 (2005)
31. Bush, W.S., Dudek, S.M., Ritchie, M.D.: Biofilter: A knowledge-integration system for the multi-locus analysis of genome-wide association studies. In: Pac. Symp. Biocomput., vol. 14, pp. 368–379 (2009)
32. Poli, R., Langdon, W.B., McPhee, N.F.: A Field Guide to Genetic Programming. Lulu Enterprises, United Kingdom (2008)
33. Holzinger, E.R., Buchanan, C., Turner, S.D., Dudek, S.M., Torstenson, E.S., Ritchie, M.D.: Optimizing Neural Networks for Detecting Gene-Gene Interactions in the Presence of Small Main Effects. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. ACM Press, New York (in press, 2010)
34. Moore, J., Parker, J., Olsen, N., Aune, T.: Symbolic discriminant analysis of microarray data in autoimmune disease. Genet. Epidemiol. 23, 57–69 (2002)
35. Moore, J.H., Asselbergs, F.W., Williams, S.M.: Bioinformatics challenges for genome-wide association studies. Bioinformatics 26(4), 445–455 (2010)
36. Kurkova, V.: Kolmogorov's Theorem is Relevant. Neural Computation 3, 617–622 (1991)

The Application of Pittsburgh-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Association Studies

Ryan J. Urbanowicz and Jason H. Moore

Dartmouth College, 1 Medical Center Dr.,
Hanover, NH 03755, USA
{ryan.j.urbanowicz,jason.h.moore}@dartmouth.edu
http://www.epistasis.org/

Abstract. Despite the growing abundance and quality of genetic data, genetic epidemiologists continue to struggle with connecting the phenotype of common complex disease to underlying genetic markers and etiologies. In the context of gene association studies, this process is greatly complicated by phenomena such as genetic heterogeneity (GH) and epistasis (gene-gene interactions), which constitute difficult, but accessible challenges for bioinformatisists. While previous work has demonstrated the potential of using Michigan-style Learning Classifier Systems (LCSs) as a direct approach to this problem, the present study examines Pittsburgh-style LCSs, an architecturally and functionally distinct class of algorithm, linked by the common goal of evolving a solution comprised of multiple rules as opposed to a single "best" rule. This study highlights the strengths and weaknesses of the Pittsburgh-style LCS architectures (GALE and GAssist) as they are applied to the GH/epistasis problem.

Keywords: Genetic Heterogeneity, Epistasis, Learning Classifier System, Genetic Algorithm, GAssist, GALE, SNP.

1 Introduction

In the modern era of complex disease research, bioinformatisists and genetic epidemiologists have teamed up in search of disease markers and etiologies. While genome-wide association studies are a current favorite strategy to search for markers and etiologies of disease, these studies tend to focus on "main effects", or associations between an individual SNP and some disease phenotype. While this may be well suited for Mendelian diseases, it has become increasingly clear that different statistical and analytical techniques are needed to address the demands of common complex disease [1]. Phenomena recognized to complicate the epidemiological mapping of genotype to phenotype include epistasis, genetic heterogeneity, phenotypic heterogeneity, trait heterogeneity, gene-environment interaction, phenocopy, and epigenetics [2]. Epistasis, or gene-gene interaction, is a particularly challenging problem given that a gene's association with disease may only be seen in the context of at least one other gene. Genetic heterogeneity (GH), refers to the presence of different underlying genetic mechanisms resulting in the appearance of the same or similar disease phenotype [3]. The quality and availability of SNPs and other genetic code information make epistasis and GH obvious targets for bioinformatic development. Previous work examining this pair of phenomena, evaluate the impact of GH on the detection and modeling of epistasis using Multifactor Dimensionality Reduction (MDR) [4]. This work demonstrated that GH dramatically hinders MDR's power to detect/model all underlying attributes involved in the underlying epistatic interactions, but additional examination indicated that MDR retains significant power to identify either the dominant branch of GH or at least one of the underlying attributes [5,6]. In the present study, GH and epistasis are modeled concurrently as they might occur simultaneously in a SNP-based genetic association study. Over the last decade, the detection and modeling of epistasis has seen a considerable amount of progress [7,8,9]. In contrast, methods for dealing with GH continue to lag behind, having relied on a traditional epidemiological paradigm with seeks to find a single best model of disease learned from a given data set [2]. These methods rely on covariate data (i.e. phenotypic data, genetic risk factors, demographic data, or endophenotypes) in order to identify more homogeneous subsets of patients. An obvious downside to this dependency, is that the success of these methods relies on the availability, quality, and relevance of these covariates. Additionally, stratification of a dataset represents a reduction in sample size for respective analysis, leading to an inevitable loss in power. In order to address these concerns, Urbanowicz and Moore recently proposed the application and exploration of learning classifier systems (LCSs) as an alternative approach to managing GH [10]. LCSs represent a class of algorithm which requires neither covariates nor data stratification, and breaks from the traditional single model paradigm by evolving a solution comprised of multiple rules. For these reasons, it was hypothesized that LCS algorithms would be useful for the detection, characterization, and modeling of GH. In [10], Michigan-style LCSs (one of two major veins of LCS architectures) were implemented and evaluated on the GH/epistasis problem. The results of that study provided a proof of principle for the use of LCSs, highlighted the strengths and weaknesses of the Michigan-style systems, and laid the foundation for the development of an LCS algorithm specifically designed to address GH. In the present study, we explore Pittsburgh-style systems, in an effort to obtain a well-rounded perspective on LCS's ability to address the GH/epistasis problem, and to compare their ability to handle the GH/epistasis problem so that a suitable architectural foundation may be selected upon which to develop an LCS customized to this epidemiological task. This study involves (1) implementing standardized versions of existing Pittsburgh-Style LCSs (GALE and GAssist), (2) performing a sweep of the major run parameters for each LCS implemented, and (3) performing a quantitative and qualitative evaluation of each LCS across the entire spectrum of simulated GH/epistatic data sets.

2 Methods

2.1 Learning Classifier Systems

LCSs combine machine learning with evolutionary computing and other heuristics to produce an adaptive system that learns to solve a particular problem. LCSs are closely related to and typically assimilate the same components as the more widely utilized genetic algorithm (GA). The goal of LCS is not to identify a single best model or solution, but to create a cooperative set of rules or models which together solve the task. The solution evolved by an LCS is represented as a population of rules, or rule-sets, which are utilized collectively to make decisions/classifications.

The infancy of LCS research saw the emergence of two founding classes of LCSs, referred to as the Michigan and Pittsburgh styles. The Michigan-style is characterized by a population of rules with a GA operating at the level of individual rules and an evolved solution represented by the entire rule population. Alternatively, the Pittsburgh-style is characterized by a population of variable length rule-sets (where each rule-set is a potential solution) with a GA operating at the level of a single rule-set. Additionally, Michigan-style systems learn iteratively from a dataset (learning once instance at a time) while Pittsburgh-style systems learn in a batch-wise fashion, learning from each instance in the dataset every iteration. The Pittsburgh-style systems implemented for this study evolve "ordered" rule sets (also known as decision lists) where rule order influences the decision making process. Because of these differences in algorithm architecture and solution size, the application of these two styles to the GH problem have been explored separately. The present study focuses on Pittsburgh-style systems, as a parallel to the recent Michigan-style LCS study [10].

Pittsburgh-style LCSs, generally possess three basic components; (1) a population of rule/classifier sets, (2) a performance component that assesses how well rule sets collectively explain the data, and (3) a discovery component that uses different operators to discover new rules and improve existing ones. For a complete LCS introduction and review, see [11].

Implementing LCSs. Each of the implemented Pittsburgh-style LCSs were selected based on their prominence, relevance, and availability. The systems implemented include GALE [12,13] ($http://www.illigal.uiuc.edu/web/xllora/wp-content/files/GALE/gale\_distribution\_0.9alpha.tar$), and GAssist [14] ($http://www.infobiotic.net/software/GAssist-Java.tar.gz$), each re-encoded and modified in Python. This was done to (1) standardize the coding language, (2) put the different LCS algorithms in a flexible, readable language to facilitate and promote future algorithm development for biologists and other non-computer scientists, (3) allow command line control over all run parameters, (4) gain a detailed understanding of each system, and most importantly (5) to standardize the rule representation. Additionally, wrapper scripts were written to run and evaluate each LCS using Dartmouth's 888 processor Linux Cluster, "Discovery". These implementations are freely available on the LCS & GBML Central webpage ($http://gbml.org/$).

A quaternary rule representation, identical to what was used in [10] was implemented in the selected Pittsburgh systems. This representation is well suited for the discrete, nominal, SNP attributes and the discrete affection status (case/control) characteristic of this problem domain. In short, the condition of the rule is represented by a string of characters from the quaternary alphabet $(0, 1, 2, \#)$ where # acts as a wildcard, and the intergers represent alternative SNP alleles. In standardizing the rule representation of GALE and GAssist, other algorithmic components which had relied on the original representation were adjusted accordingly (e.g. crossover and mutation mechanisms).

Two Pittsburgh-Style LCSs. While the systems GALE and GAssist share a number of features common to Pittsburgh systems, including batch/offline learning, ordered rule-sets (decision lists), accuracy based fitness, and supervised learning, they have distinct population architectures, and possess a very different set of supporting heuristics. Also, encodings of both systems were originally implemented in Java, and allowed for multiple knowledge representations. For simplicity only the applicable quaternary representation described above was encoded in the present python implementations. Every rule-set (a.k.a. agent) of a Pittsburgh system represents a potential solution to a classification problem. GALE, or the Genetic and Artificial Life Environment [12,13] is described as a fine-grained parallel evolutionary algorithm. GALE uses a 2D grid to evolve a population of rule-sets spatially, where discovery mechanisms may operate only within the local neighborhood. GAssist, or (Genetic clASSIfier sySTem) [14] descends from GABIL [15], having introduced several modifications to make it one of the most competitive and flexible Pittsburgh systems to date. These include elitism, an adaptive discretization intervals (ADI) rule representation, windowing, intelligent initialization, minimum description length (MDL)-based fitness, and the incorporation of an explicit default rule, detailed in [14]. In GAssist, genetic operators function at the population level.

2.2 System Evaluations

Each system was evaluated over the spectrum of simulated datasets described and generated in [10]. In brief, 1440 simulated SNP datasets, representing 96 data set configurations of differing GH heritability combinations, sample sizes, mix ratios, and difficulties were utlized. LCS evaluations were performed exactly as they were in [10], adding two new power estimates, and an additional tracking parameter to the previously used metrics; (1) an estimate of power to correctly detect predictive attributes (MichiganPower(MP)), (2) testing accuracy (10-fold cross validation strategy employed), (3) computational time, and (4) solution generality. As a precursor to evaluating each LCS across the spectrum of simulated datasets, a parameter sweep was conducted in order to roughly optimize the parameter settings for the respective algorithms. Parameters examined in GALE include: (1) the probability of wild incorporation (0.5 and 0.75), (2) pruning; a rule deletion mechanism (*on* or *off*), (3) resource allocation; what instances were made available to cells within the 2D board (uniform, and pyramidal), and (4) maximum population size (MPS); the maximum number of rule-sets $(100, 625, 2500)$ [12]. Alternatively, parameters examined in GAssist include: (1) probability of wild incorporation (0.5 and 0.75), (2) population initialization (random, smart, class-wise (cw)), (3) MDL fitness (*on* or *off*), (4) an explicit default class (*on* or *off*), (5) windowing/window size (1, 2, 4), and (6) MPS $(100, 625, 2500)$ [14].

Power, or success rate, is typically estimated in these types studies by tracking the frequency with which an algorithm successfully identifies the correct underlying model or attribute(s), across some number of data set replicates. This type of estimation is not applicable to either Michigan or Pittsburgh LCSs, which both evolve solutions made up of many rules. While the function of an LCS is to evolve a solution which can accurately classify patients, it is more important, and arguably much more difficult to evolve a solution that is meaningful and interpretable for a genetic epidemiologist. With this in mind, [10] developed "MP", based on the idea that attributes unimportant to the underlying model(s) (noise attributes) would tend to be generalized ('#'/don't-care symbol used) more frequently within rules of the population. Pittsburgh-style LCSs, which evolve solutions made up dramatically fewer rules, required the addition of two additional power estimates, (BothPower(BP) and SinglePower(SP)) which indicate whether a precise predictive rule exists for both underlying epistatic models, or for at least a single underlying model, respectively. In addition, this study also tracked the proportion of the rules which were accurate predictive rules (i.e rules which accurately specified both attributes of an epistatic pair, but generalized across all other attributes with "#").

All statistical evaluations were completed using R. Logistic regression, modeling each data set dimension and all pair-wise combinations of dimensions, was employed for the evaluations of power. ANOVA and Tukey's HSD posthoc analysis was employed for the evaluation of accuracies, generality, predictive rule proportion, and run time. Differences were considered to be significant at $(p < 0.05)$.

3 Results

3.1 Parameter Sweep

Completion of a parameter sweep led to the selection of the following parameters for GALE (MPS = 2500, Wild = 0.75, Pruning = On, Rescource Allocation = Uniform) and for GAssist (MPS = 625, Wild = 0.75, Initialization = CW, MDL Fitness = On, Default Class = On, Windows = 4). Any algorithm parameters not evaluated in the parameter sweep were left at their respective default settings. While both systems showed an improvement in both MP and testing accuracy with increasing MPS, GAssist runs allotted a MPS of 2500 failed to complete within a reasonable amount of time ($<$ 30 hours), thus an MPS of 625 was selected. Early testing of the two Pittsburgh systems indicated that both algorithms were struggling to perform well on the allotted task within the maximum time frame of 30 hours. Examination of the rule sets suggested that generalization may have been contributing to the slow initial learning of both systems. An examination of “wild” as a parameter, indicated that a value of 0.75 yielded a significant improvement for all power estimates, and testing accuracy. For GALE, the “uniform” resource allocation showed significantly improved testing accuracy at the expense of about twice as much run time, and “pruning” (a built in function of GAssist) approximately halved run time while significantly improving agent genenerality. In GAssist,(1) “CW” initialization significantly improved testing accuracy while taking significantly less run time than “smart”, (2) turning on MDL fitness significantly reduced run time while increasing generality, MP, and BP, (3) using an explicit default class significantly reduced run time, generality, and BP while significantly improving testing accuracy, MP, and SP, and (4) windowing, designed to reduce computational time and increase generalization [14], did just that, along with significantly improving MP and BP over the increase in window size from 1-4.

3.2 Pittsburgh LCS Evaluations

Table 1 summarizes statistics describing the performance of the respective Pittsburgh LCSs. It is important to note that averages are calculated over every simulated dataset with the intention of comparing the overall performance of systems. Low overall averages of power and testing accuracy are therefore expected. “Best” power and testing accuracies come from the configuration of datasets with the highest respective values. All values in the table are representative of the performance of agents with the highest fitness in their respective evolved populations. Figure 1 summarizes evaluations of testing accuracy (TA) and each power estimate across the four dimensions of simulated data. An immediate observation of this study was that the MP estimation method developed for [10], showed little to no success for any Pittsburgh systems examined. Logistic regression models indicate that each of the selected dataset dimensions (i.e. model combo, sample size, mix ratio, and difficulty) are significant predictors of all power estimations examined, as is the choice of LCS algorithm. Additionally, the interaction between algorithm choice and sample size for SP was significant, indicating that the choice of Pittsburgh LCS impacts the power to evolve at least one precise predictive rule given varying sample sizes. While MP and BP values over all simulated datasets were generally quite low, GAssist yielded significantly higher MP and SP than GALE. Accuracy evaluations indicate that GAssist evolves agents with significantly higher testing accuracies, and significantly lower training accuracies than GALE. All dataset dimensions were found to be significant predictors of both testing and training accuracies in both systems. Similar to [10], datasets with a mix of 75:25 yielded significantly higher testing accuracies than 50:50. This supports the expectation that LCSs would be more proficient at learning a model which dominates the sample population. It is therefore not surprising that while the LCSs examined obtain higher testing accuracies and SP for datasets with a 75:25 model ratio, power estimates which rely on finding both underlying models (i.e. MP, and BP) obtain significantly lower values. Further evaluation demonstrates that GAssist, while

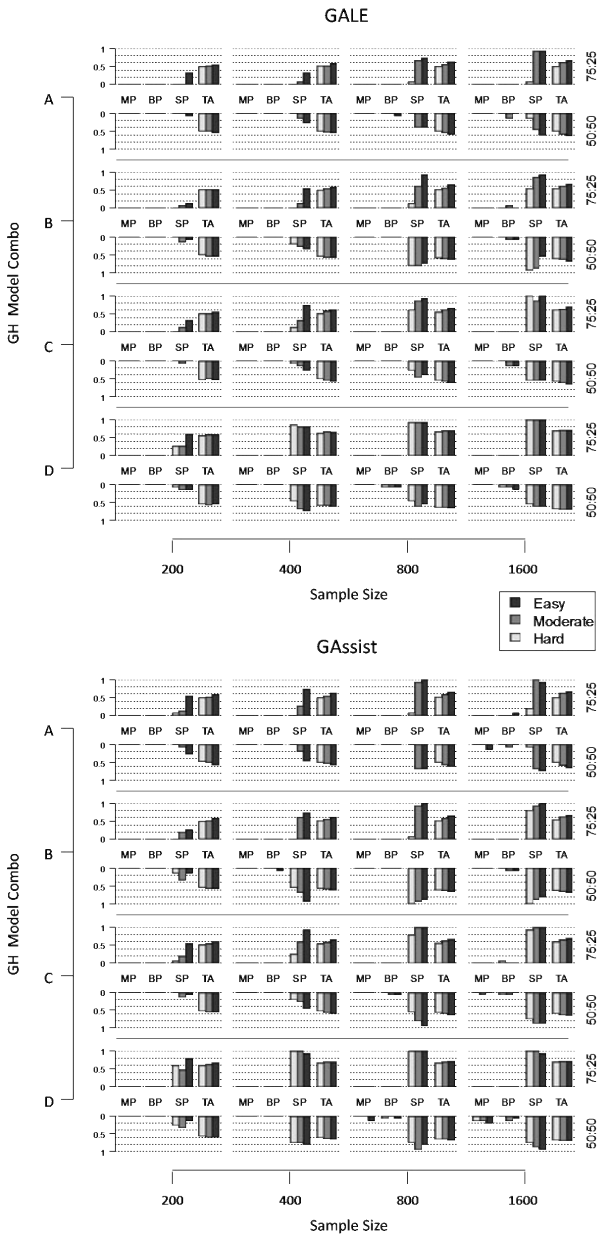


Fig. 1. An illustration of testing accuracy and the three power estimates gathered from each LCS evaluation. The plot for each LCS depicts the results over each dimension of the simulated dataset (i.e. model combination, sample size, mix ratio, and penetrance table difficulty / see section 2.2). The bars of each sub-plot represent an evaluation over 15 random seed datasets. Model combinations include the following pairs of heritability; A = 0.1 & 0.1, B = 0.1 & 0.4, C = 0.2 & 0.2, and D = 0.4 & 0.4.

Table 1. LCS Algorithm Summaries

Assorted	GALE	GAssist	Power Estimates	GALE	GAssist
Best Testing Accuracy	0.7162	0.7192	**Best MP**	0.0	0.2
Average Testing Accuracy	0.5828	0.6004	**Average MP**	0.0	0.0083
Average Training Accuracy	0.7862	0.7189	**Best BP**	0.1333	0.1333
Average Agent Size	29.60	8.01	**Average BP**	0.0118	0.0104
Average Generality	0.7889	0.7876	**Best SP**	1.0	1.0
Average Run Time (Min.)	113.79	208.02	**Average SP**	0.4403	0.5972
Average Predictive Rule %	0.03278	0.14703			

Table 2. Evolved GAssist Agent

Condition	Class	Matched	Correct Class	Accuracy
1 1 ##################	0	101	96	0.9505
0 # 0 0 ################	0	453	266	0.5872
## 1 1 ################	0	167	125	0.7485
0 0 #### 0 ############ 0	0	143	82	0.5734
####################	1	576	425	0.7378

evolving agents with the fewest average number of rules, also evolves solutions that on average are less general and require a longer runtime. Table 2 depicts one such agent evolved by GAssist. While every evaluation described above was performed over 100 learning iterations, performance after only 25, and 50 iterations was also tracked. The impact of learning iteration on testing accuracy and the power estimates is summarized by the following; (1) testing accuracy and SP continued to significantly improve within iterations 25-100 and 50-100 for all systems, (2) none of the Pittsburgh systems saw significant improvement in MP from 50-100 iterations, (3) from iterations 50-100, GALE and GAssist each saw a small drop in BP. Overall, additional iterations and run time would seem to benefit testing accuracy and SP while suggesting little promise of improving MP or BP in the Pittsburgh LCSs examined.

4 Discussion and Conclusions

One of the most obvious advantages of the two Pittsburgh-style LCSs examined is the compact solutions which they evolve. Compared to the population-sized solutions evolved by the Michigan-Style systems [10], GALE and GAssist evolve agents which are compact enough for an epidemiologist to directly extract knowledge. Overall, GALE and GAssist both show promise addressing the GH/epistasis problem domain, evidenced by competitive testing accuracies and manageable solution sizes. While the study-wide average testing accuracy of GAssist is comparable to that of UCS in [10], the apparent failure of both Pittsburgh LCSs to evolve a rule set with a correct predictive rule characteristic of both underlying epistatic models (i.e. BP), represents a key challenge for adapting this style of LCS to the GH problem. One likely reason for this difficulty, stems from two intertwined features (i.e. the decision list and default rule), common to both Pittsburgh systems, designed to speed up and improve classification accuracy and agent compactness [14]. By ordering the rules of an agent (via the decision list), the explicit pressure on rule-sets to be accurate, intuitively drives the evolution of a default rule (a rule that is entirely general, or a "catch-all"), which implicitly covers all un-matched instances, or in the case of an explicit default rule (as found in GAssist), covers the "default-class". While this is an effective method for condensing rule-set size, the default rule (either evolved, or explicitly included) essentially eliminates the specification of predictive rules for an entire class, reducing rule diversity, and likely having a direct impact on the power estimation methods intended to evaluate LCSs.

The results of the parameter sweep serve not only to roughly optimize each LCS for a more thorough comparison, but demonstrate the importance of the accessory features available to each respective system. The evaluation of each system over the complete spectrum of simulated data suggests the following; (1) the selected rule representation allowed each system to evolve interpretable rules which model specific genotype combinations, (2) both GALE and GAssist were able to achieve significant power to identify at least one predictive rule from one of the two underlying models (> 0.8) for a number of challenging data configurations, as well as average testing/prediction accuracies significantly higher than 0.5, (3) both GALE and GAssist are computationally intensive, and would likely benefit from a greater number of learning iterations, and (4) the small rule-sets evolved by GALE and GAssist offer the potential for users to identify underlying GH. As an example of such interpretability, refer to the agent depicted in Table 2 where the underlying GH is correctly characterized by the first and third rules which specify two high risk epistatic genotype combinations involving two separate pairs of attributes. In this example the correct underlying models involve attributes/SNPs (1, 2) and (3, 4) respectively.

The results of this study support the employment of LCSs to address the detection, modeling and characterization of attributes associated with common complex disease given the complicating presence of underlying GH and epistasis. However, these findings do not necessarily indicate that a given LCS will perform "better" or "worse" within any other problem domain to which they might be applied. The collective findings of [10] and the present study will be used to direct the development of a problem-specific LCS algorithm, aimed at (1) improving the power to detect underlying attributes, (2) being able to distinguish distinct underlying models constituting the GH, and (3) enhancing the overall interpretability of the evolved LCS rule-population solution.

Acknowledgments. This work was supported by the William H. Neukom 1964 Institute for Computational Science at Dartmouth College along with NIH grants AI59694, LM009012, and LM010098. We would like to thank Xavier Llora, and Jaume Bacardit for making open source Java implementation of their respective LCS algorithms freely available.

References

1. Moore, J., Asselbergs, F., Williams, S.: Bioinformatics challenges for genome-wide association studies. Bioinformatics 26(4), 445 (2010)
2. Thornton-Wells, T., Moore, J., Haines, J.: Genetics, statistics and human disease: analytical retooling for complexity. Trends in Genetics 20(12), 640–647 (2004)
3. Sing, C., Boerwinkle, E., Moll, P.: Strategies for elucidating the phenotypic and genetic heterogeneity of a chronic disease with a complex etiology. Progress in Clinical and Biological Research 194, 39 (1985)
4. Ritchie, M., Hahn, L., Moore, J.: Power of MDR for detecting gene-gene interactions in the presence of genotyping error, missing data, phenocopy, and genetic heterogeneity. Genetic Epidemiology 24(2), 150–157 (2003)
5. Ritchie, M., Edwards, T., Fanelli, T., Motsinger, A.: Genetic heterogeneity is not as threatening as you might think. Genetic Epidemiology 31(7), 797 (2007)
6. Digna, T., Dudek, R., Ritchie, M.: Exploring the performance of multifactor dimensionality reduction in large scale SNP studies and in the presence of genetic heterogeneity among epistatic disease models. Hum. Hered. 67, 183–192 (2009)
7. Ritchie, M., Hahn, L., Roodi, N., Bailey, L., Dupont, W., Parl, F., Moore, J.: MDR reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics 69(1), 138–147 (2001)
8. Cordell, H.: Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans. Human Molecular Genetics 11(20), 2463 (2002)
9. Moore, J., Gilbert, J., Tsai, C., Chiang, F., Holden, T., Barney, N., White, B.: A flexible computational framework for detecting, characterizing, and interpreting statistical patterns of epistasis in genetic studies of human disease susceptibility. Journal of Theoretical Biology 241(2), 252–261 (2006)
10. Urbanowicz, R., Moore, J.: The Application of Michigan-Style Learning Classifier Systems to Address Genetic Heterogeneity and Epistasis in Asssociation Studies. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. ACM, New York (in Press, 2010)
11. Urbanowicz, R., Moore, J.: Learning Classifier Systems: A Complete Introduction, Review, and Roadmap. Journal of Artificial Evolution and Applications (2009)
12. Llora, X., Garrell, J.: Automatic Classification and Artificial Life Models. In: Proceedings of Learning Workshop IEEE and Univesidad Carlos III (2000)
13. Llora, X., Garrell, J., et al.: Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Third Genetic and Evolutionary Computation Conference, pp. 461–468. Morgan Kaufmann, Citeseer (2001)
14. Bacardit, J.: Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time. PhD thesis, Enginyeria i Arquitectura La Salle, Ramon Llull University, Barcelona, European Union, Catalonia, Spain (2004)
15. De Jong, K., Spears, W.: Learning concept classification rules using genetic algorithms. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, Citeseer, pp. 651–656 (1991)

Threshold Selection, Mitosis and Dual Mutation in Cooperative Co-evolution: Application to Medical 3D Tomography

Franck P. Vidal[1], Evelyne Lutton[2], Jean Louchet[3], and Jean-Marie Rocchisani[4]

[1] Department of Radiation Oncology, University of California, San Diego, CA
franck.p.vidal@gmail.com
[2] INRIA - Saclay-Île-de-France, AVIZ team, Orsay, France
Evelyne.Lutton@inria.fr
[3] Artenia, Châtillon, France
Jean.Louchet@gmail.com
[4] Paris XIII University, UFR SMBH & Avicenne hospital, Bobigny, France
jean-marie.rocchisani@avc.aphp.fr

Abstract. We present and analyse the behaviour of specialised operators designed for cooperative coevolution strategy in the framework of 3D tomographic PET reconstruction. The basis is a simple cooperative co-evolution scheme (the "fly algorithm"), which embeds the searched solution in the whole population, letting each individual be only a part of the solution. An individual, or fly, is a 3D point that emits positrons. Using a cooperative co-evolution scheme to optimize the position of positrons, the population of flies evolves so that the data estimated from flies matches measured data. The final population approximates the radioactivity concentration. In this paper, three operators are proposed, threshold selection, mitosis and dual mutation, and their impact on the algorithm efficiency is experimentally analysed on a controlled test-case. Their extension to other cooperative co-evolution schemes is discussed.

1 Introduction

Evolutionary algorithms have been proven efficient to solve the inverse problem of 3D data reconstruction in tomography [1], and particularly of positron emission tomography (PET) reconstruction in nuclear medicine [2,3,4].

In PET, a positron emitter is used as radionuclide for labelling. Positrons generally lead to an annihilation reaction, that emits two photons of 511 keV in opposite directions. 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 joining the detectors that have been activated for a given pair of photons is called "line of response" (LOR). An overview of reconstruction methods in nuclear medicine can be found in [5].

In previous work, we showed that a cooperative coevolution strategy (or Parisian evolution) called "flies algorithm" [6] could be used in Single-Photon

Emission Computed Tomography (SPECT) reconstruction [1], and also in PET reconstruction in 2D-mode [2,3], and in Fully-3D-mode [4]. The marginal fitness was used to propose new operators, i) the threshold selection, and ii) the mitosis, but no performance analysis of these operators has been performed so far.

This paper addresses this deficiency and it analyses the impact of each operator on the performance of a PET reconstruction algorithm on a controlled test-case. A new operator (namely the dual mutation) and a pre-initialisation of the flies' position using back-projection are also described and analysed. Standard PET reconstruction algorithms are reviewed in Section 2. It is followed by an overview of the fly algorithm for PET reconstruction. The three operators that control a varying population size scheme are presented in Section 4, as well as an alternate initialisation process. Experimental setup and analysis are given in Section 5, before presenting some conclusions and future work in Section 6.

2 Standard PET Reconstruction Algorithms

Tomography reconstruction algorithms can be divided into two main categories.

On the one hand, there are analytical methods. These are based on a continuous modelling and the reconstruction consists in the inversion of measurement equations, such as the well known Filtered Back-Projection (FBP). This method is now rarely used due to strong artefacts in the reconstructed data (see Fig. 5) and also because the correction of imaging physics effects need to be undertaken before the reconstruction, leading to a systematic positive bias in the reconstructed volume.

On the other hand, there are iterative methods. This class of methods can be split into two kinds. Algebraic methods are used in X-ray Computed Tomography (CT); statistical methods are used in nuclear medicine for both SPECT and PET [7]. They take into account noise, and the correction of imaging physics can be applied during the reconstruction in the iterative steps. Iterative methods are relatively easy to model. In practice, the volume is usually discretised into voxels. Each voxel intensity is treated as an unknown. A system of linear equations is defined according to the imaging geometry and physics: $\mathbf{p} = \mathbf{R}\,\mathbf{f}$, with $\mathbf{f}$ the volume to recover, $\mathbf{p}$ the measured data, $\mathbf{R}$ the system model. Imaging physics, such as non-uniform attenuation, scatter, etc. can be modelled in $\mathbf{R}$, whereas they are difficult to handle in an analytic algorithm. The system of equations is finally solved using the iterative algorithm.

There are different ways to implement these iterative methods. The main differences are about the computation of the projections, the physics corrections (scattering, random, attenuation, etc.) are applied, and how the error corrections are applied in the estimated projections.

The Maximum Likelihood - Expectation Maximisation (ML-EM) (or 'EM") is a common algorithm in SPECT and PET. It assumes Poisson noise is present in the projection data. ML-EM does not produce artefacts seen in FBP reconstructions, and it has a better signal-to-noise ratio in region of low concentration. However, the algorithm converges slowly.

The Ordered Subset - Expectation Maximization (OS-EM) has been proposed to speed-up convergence of the EM algorithm. Its principle is to reduce the number of projections used at each iteration of the EM algorithm. Projections are grouped in K sub-groups. The projections of a sub-group are uniformly distributed around the volume to reconstruct.

3 PET Reconstruction Using the Fly Algorithm

The fly algorithm for tomography reconstruction follows the iterative paradigm. The steps of the iterative method can be described as follows:

1. Each individual, or fly, corresponds to a 3D point. Initially, the flies' position is randomly generated in the volume within the scanner. The population of flies corresponds to the tracer density in the patient.
2. To produce estimated projection data, each fly mimics a radioactive emitter, i.e. a stochastic simulation of annihilation events is performed. For each annihilation event, a photon is emitted in a random direction. A second photon is then emitted in the opposite direction. If both photons are detected by the scanner, the corresponding LOR is recorded. The scanner properties (e.g. detector blocks and crystals positions) are modelled, and each fly is producing an adjustable number of annihilation events.
3. The optimisation is performed using genetic operations. The fitness function used during the selection operation takes into account the comparison between the estimated projections and the measured projections.
4. Using genetic operations to optimise the position of radioactive emitters, the population of flies evolves so that the population total pattern matches measured data.
5. Instead of a "generational" evolutionary strategy, in which at each loop every individual (fly) will be eliminated and replaced with a new fly, we chose a "steady state" evolutionary strategy.

Note that in classical evolutionary approaches, each individual in the population is a potential solution; in the Fly approach, a subset of the evolving population itself is the representation of the solution. After convergence, the "good" flies (see Section 4.1) are then extracted to form the reconstructed volume.

4 Varying Population Size Scheme in a Cooperative Co-evolution Algorithm

Cooperative co-evolution strategies rely on a "social" formulation of the optimisation problem, where individuals collaborate or compete in order to collectively build a solution. The fly algorithm is a mono-population strategy (Parisian approach): all flies contribute independently and collectively to build the solution. In [8] a variable sized population Parisian GP strategy has been successfully used on a cooperative co-evolution, based on adaptive population deflating and inflating schemes. We test in this paper an "inflating-only" strategy, the mitosis, described below, to gradually increase the precision of the reconstructed data.

4.1 Marginal Fitness

In this application, the similarities or discrepancies between the estimated projection data and the measured projection data provided by the imaging system have to be assessed. We chose City Block distance (also known as Manhattan distance) as the fitness metrics to measure the distance between two LOR sets. It provides a good compromise between speed and accuracy. Eq. 1 provides the global fitness, i.e. the population's cost:

$$\mathrm{dist}(LOR_m, LOR_e) = \sum_{i}^{M} \sum_{j}^{M} |LOR_m(i,j) - LOR_e(i,j)| \tag{1}$$

with $\mathrm{dist}(LOR_m, LOR_e)$ the City Block distance between LOR_m and LOR_e, the set of LORs for the measured data and the estimated data respectively, $LOR(i,j)$ is the number of counts of a LOR between the photon detectors i and j, M is the total number of photon detectors within the imaging system. LOR sets are efficiently implemented using triangular sparse matrices to reduce the amount of memory needed to store the data. The smaller global fitness is, the closer the simulated data will be to the measured data.

In [1], 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. To address this, we introduced marginal evaluation to assess a given fly. We use a similar approach in PET:

$$\mathrm{F_m}(x) = \mathrm{dist}\left(LOR_e - \{LOR_x\}, LOR_m\right) - \mathrm{dist}\left(LOR_e, LOR_m\right) \tag{2}$$

with $\mathrm{F_m}(x)$ the marginal fitness of Fly x, and $LOR_e - \{LOR_x\}$ is the set of LORs simulated by the whole population without Fly x. In practice, each fly needs to keep a record of its simulated LORs.

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.

4.2 Threshold Selection

The fly to be killed is randomly chosen by the "selection" operator, with a bias towards killing "bad" individuals. On the other hand, if the new fly is to be created by mutation of another fly, this fly is randomly chosen by the "selection" operator, with a bias towards reproducing "good" individuals. Classical selection operators are ranking, roulette wheel and tournament [9]. In our algorithm, as each fly's fitness is the value of its (negative or positive) contribution to the quality of the whole population, we managed to simplify and speed up the selection process by using a fixed fitness threshold. Any "bad" fly (its fitness is negative) is a candidate for death, and any "good" fly (its fitness is positive) is a candidate for mutation.

4.3 Mitosis

When the number of flies with a negative fitness decreases, the threshold selection fails to provide flies to be killed in an acceptable time. It also means that the reconstruction is optimum at the current resolution. If the resolution is acceptable, i.e. there are enough flies to approximate the radio-tracer concentration, then the algorithm can stop and the reconstructed volume is extracted using flies with a positive fitness. If not, a mitosis operator is triggered to gradually increase the population size. Each fly is split into two new flies to double the population size. One of the two flies is then mutated.

4.4 Dual Mutation

To optimise the flies' position, our algorithm takes advantage of a mutation operator. When a new fly (b) is created by mutation of an old "good" fly (a), the position of Fly b is first initialised to the same position as Fly a. The new fly is then stochastically translated in any direction, and LORs are randomly generated from that fly. The length of the translation vector is a random variable that follows a Gaussian law whose mutation variance is σ^2. It needs first to be set to a large value to better explore the search space. However, a constant large mutation variance will lead to blurred reconstructed volumes. σ has therefore to be gradually reduced.

The use of adaptive mutations in evolutionary algorithms is an ancient idea, directly inspired by natural adaptive phenomena, e.g. mutations simulated by stress [10]. In artificial evolution, various adaptive schemes have been considered for mutation [11], depending of the parameter to be adapted (standard deviation, σ [12], or mutation law [13] for continuous mutation, mutation probability for discrete mutations [14]). Regarding the adaptation of σ, one can distinguish several strategies :

– σ is directly adapted to local measurements, like fitness [15] or local regularity [16],
– σ is tuned depending on some success measurement: in this category fall the famous $1/5^{th}$ rule proposed by Schewefel [17,18],
– σ is subject to an adaptive pressure itself, it is self-adapted [19]: σ is considered as an additional parameter in the genome, and a log-normal Gaussian law is used to control the "mutation over the mutation".

These techniques have been proven efficient in various cases, depending on the fitness function and the genetic engine. It has however to be noticed that the sophistication of a mutation operator has a computational cost, and that some very rough schemes may perform better due to their capability to rapidly test numerous sample points [20].

Concurrent testing with various subpopulation has been also considered for mutation law adaptation [13].

Here we propose an adaptive mutation scheme based on the concurrent testing of two alternative σ values (σ_{low} and σ_{high}, with $k\ \sigma_{low} = \sigma_{high}$). The update rule is multiplicative as for the $1/5^{th}$ rule. If σ_{high} gives the best results during the previous period, then both mutation variances are increased by a predefined factor (pf, with $pf > 1$). If σ_{low} gives better results, then the variances are multiplied by $\frac{1}{pf}$. The major advantage of this dual mutation scheme is to provide a fully automatic method to adapt the mutation variance, whilst keeping the administration cost of the algorithm relatively light. Additionnally, this scheme does not need to make any assumption on the ideal success rate of the mutation as in the $1/5^{th}$ rule. In practice, the global fitness is recorded after each mutation. The cumulative difference of the global fitness ($\Delta(\sigma)$) before and after mutations is computed to determine which σ value provides the best performance over a given period of time. To prevent oscillation of σ values, a criteria can be added to avoid changes when both σ_{high} and σ_{low} provide relatively similar results, e.g. when the absolute difference between $\Delta(\sigma_{low})$ and $\Delta(\sigma_{high})$, relative to the current global fitness, is below a given threshold (t_{mut}).

4.5 Initialisation of Flies on LORs

Iterative reconstruction methods generally make use of a constant volume as an initial estimate of the volume (see Fig. 5(a)).

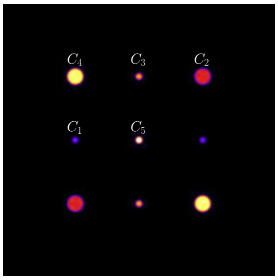
However, to speed-up the reconstruction process, a volume is first reconstructed using a fast analytical algorithm, the simple back-projection, that we implemented on the graphics card using OpenGL. The algorithm consists in back-projecting each LOR into the volume space. Pixels along the path of a LOR are updated uniformly. This operation is fast and provides the evolutionary algorithm with an initial guess of the volume (see Fig. 5(c)). For each voxel of the initial estimate, a given number of flies is assigned depending on the voxel intensity (see Fig. 5(b)).

5 Results

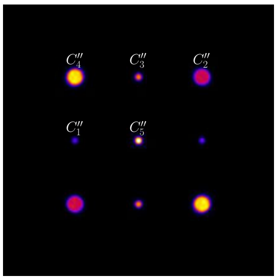
The validity of the reconstruction method has been addressed in [4]. In this paper, we focus on the evaluation of the performance of the new genetic operators. For each test case, 750000 new individuals have been created. For each tested configuration, the reconstruction has been repeated 20 times, and the final global fitness was recorded. For every test, unless specified, the dual variance and the threshold selection operators have been enabled. Results are presented using box plots (also called box-and-whisker diagrams).

5.1 Experimental Setup

Here, a single ring PET system is considered. Its radius is about 430mm. The ring is made of 72 linear blocks that include 8 crystals each. The width of a crystal is about 4.5mm. Fig. 1(a) shows the reference image. It includes 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) Reference image.



(b) Reconstructed image.

Fig. 1. Slices (512 × 512 pixels) through the cylinders

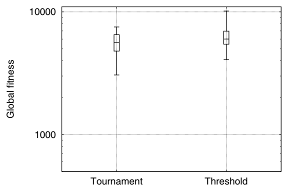


Fig. 2. Performance of the threshold selection and of the tournament selection

5.2 Threshold Selection

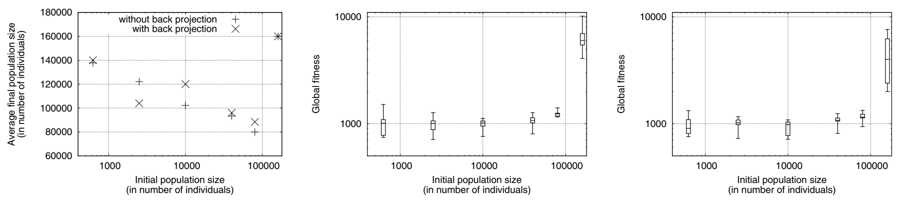
The size of the population is fixed (160000 flies), i.e. no mitosis has been triggered. The performance of the threshold selection and the tournament selection are presented in Fig. 2. Both operators provide similar performance. The threshold selection is then preferred because of the additional information that it brings: enable mitosis, and provide a convergence criteria at a given resolution (i.e. for a given size of population).

5.3 Mitosis

Two variables have to be assessed at the end of the reconstruction: the current size of the population, and the global fitness. The larger the final population, the better the image resolution can be obtained. The smaller the global fitness, the closer the estimated data to the measured data is.

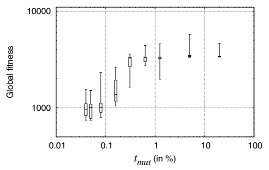
Fig. 3(a) shows the average number of flies in the final population depending on the initial size of the population (625, 2500, 10000, 40000, 80000, and 160000 flies). When the size of the population is 160000 flies, no mitosis has been triggered. Fig. 3(b) shows the corresponding global fitness.

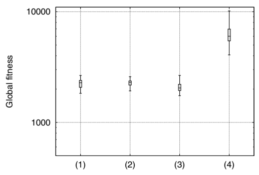
Similar performance in term of global fitness is obtained when the initial population size is below 10000 flies. The highest final population size is obtained with the smallest initial population size. Then, the reconstruction converges much faster when the initial population is small (smaller global fitness and bigger final population size). These results validate the efficiency of the mitosis operator.



(a) Average final population size.
(b) Mitosis without initialisation of the flies' position.
(c) Mitosis with initialisation of the flies' position using Fig. 5(c)

Fig. 3. Performance of the mitosis operator using variable initial population sizes





(a) When the initial population size is 625 flies, and depending on t_{mut} value.

(b) When the initial population size is 160000 flies with: (1) constant $\sigma = 0.1$mm, (2) constant $\sigma = 0.01$mm, (3) $t_{mut} = 0.0\%$, and (4) $t_{mut} = 0.05\%$.

Fig. 4. Performance of the dual variance operator

5.4 Dual Mutation

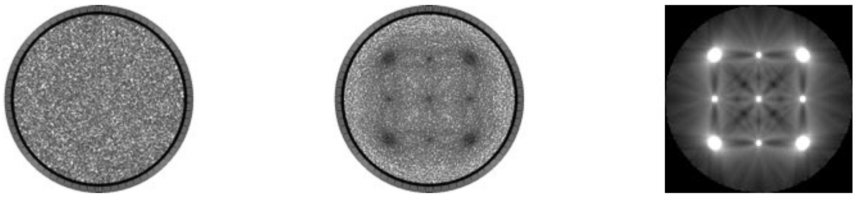
The initial σ_{low} value in this test is 35mm, pf is equal to $\sqrt[3]{2}$, and σ_{high} is equal to $\sqrt[3]{2}\sigma_{low}$. Different threshold values (t_{mut}) have been tested to limit oscillations of σ values (Fig. 4(a)). Larger values not only prevent oscillations, they also prevent any change of σ values, leading to unsatisfactory results.

Fig. 4(b) shows the global fitness obtained i) using a constant variance (see (1) and (2)), or ii) enabling dual mutation operator (see (3) and (4)). The best results are observed using the dual mutation operator with a very low t_{mut} value.

5.5 Initialisation of Flies on LORs

Fig. 5(a) and Fig. 5(b) show two possible initial estimates of the radio-active concentration. In the first case, flies are uniformly located within the space in the imaging system. In the latter case, the position of flies is initialised using the simple back projection. Fig. 3 shows the performance of both strategies when the mitosis operator is enabled.

When the initial size of the population is relatively large, the algorithm converges much faster using this initialisation step. This is not the case when the initial size of the population is relatively small. It may be due to the fact that



(a) Uniform distribution of the flies.

(b) Initialisation of the flies' position using Fig. 5(c)

(c) Image reconstructed using the simple back projection algorithm.

Fig. 5. Initial estimates of the reconstructed image

the algorithm converges fast enough when only a few flies are used. When the initial number of flies is slightly higher, the reconstruction converges faster when the position of flies are initialised using the back projection.

6 Conclusion and Futher Works

We have presented new operators in cooperative co-evolution and validated their efficiency using a controlled test-case in PET reconstruction. Both the threshold selection, mitosis and dual mutation operators have shown their usefulness and ability. Experimental statistics show that the threshold selection perform as well as the tournament selection, but it has the great advantage of bringing a convergence criterion related to the current resolution. Additionally, it allows to trigger an automatic mitosis, i.e. doubling the population size, to improve the resolution. Best performance, both in term of final resolution and convergence, are obtained using small initial population size. The dual mutation operator provides an adaptive mutation variance that has proven to be better than using fixed mutation variances.

Such operators can be used in any other cooperative co-evolution schemes as soon as a marginal fitness can be considered as beneficial, that obviously depends on the computation cost of the marginal fitness. For instance, threshold selection, mitosis and dual selection will be considered as further work for the original fly algorithm on a stereo-vision application ([6]). The marginal fitness will also be considered for developing a "deflating operator". This additional mechanism for controlling the population size may be interesting in the case of applications whose resolution does not depend on the size of the final population. Further work will also include the correction of photon attenuation and Compton scattering, and a concurrent study with the OS-EM algorithm.

References

1. 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)

2. Vidal, F.P., Lazaro-Ponthus, D., Legoupil, S., Louchet, J., Lutton, E., Rocchisani, J.: Artificial evolution for 3D PET reconstruction. In: Collet, P., Legrand, P. (eds.) EA 2009. LNCS, vol. 5975, pp. 37–48. Springer, Heidelberg (2010)
3. Vidal, F.P., Louchet, J., Lutton, E., Rocchisani, J.M.: PET reconstruction using a cooperative coevolution strategy in LOR space. In: IEEE Nuclear Science Symposium Conference Record, Orlando, Florida, pp. 3363–3366. IEEE, Los Alamitos (October 2009)
4. Vidal, F.P., Louchet, J., Rocchisani, J.M., Lutton, E.: New genetic operators in the fly algorithm: application to medical PET image reconstruction. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A.I., Goh, C.-K., Merelo, J.J., Neri, F., Preuß, M., Togelius, J., Yannakakis, G.N. (eds.) EvoApplicatons 2010. LNCS, vol. 6024, pp. 292–301. Springer, Heidelberg (2010)
5. Zeng, G.L.: Image reconstruction – a tutorial. Comput. Med. Imaging Graph. 25(2), 97–103 (2001)
6. Louchet, J.: Stereo analysis using individual evolution strategy. In: Proc. ICPR 2000, p. 1908 (2000)
7. Vandenberghe, S., D'Asseler, Y., Van de Walle, R., Kauppinen, T., Koole, M., Bouwens, L., Van Laere, K., Lemahieu, I., Dierckx, R.A.: Iterative reconstruction algorithms in nuclear medicine. Comput. Med. Imaging Graph. 25, 105–111 (2001)
8. Barriere, O., Lutton, E.: Experimental analysis of a variable size mono-population cooperative-coevolution strategy. In: Proc. NICSO 2008 (2008)
9. Baeck, T., Fogel, D.B., Michalewicz, Z. (eds.): Evolutionary Computation 1: Basic Algorithms and Operators. Taylor & Francis, Abington (2000)
10. Rosenberg, S.M.: Evolving responsively: adaptive mutation. Nat. Rev. Genet. 2, 504–515 (2001)
11. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE T. Evolut. Comput. 3(2), 124–141 (1999)
12. Bäck, T., Schwefel, H.P.: An overview of evolutionary algorithms for parameter optimization. Evol. Comput. 1(1), 1–23 (1993)
13. Chellapilla, K.: Combining mutation operators in evolutionary programming. IEEE T. Evolut. Comput. 2(3), 91–96 (1998)
14. Ochoa, G.: Setting the mutation rate: Scope and limitations of the 1/L heuristic. In: Proc. GECCO 2002, pp. 495–502. Morgan Kaufmann Publishers Inc., San Francisco (2002)
15. Lis, J.M., Orlowska-Kowalska, T.: Application of evolutionary algorithms with adaptive mutation to the identification of induction motor parameters at standstill. In: Proc. EUROCON 2007, pp. 1786–1791 (2007)
16. Lutton, E., Lévy Véhel, J.: Pointwise regularity of fitness landscapes and the performance of a simple ES. In: Proc. CEC 2006, pp. 16–21 (2006)
17. Schwefel, H.P.: Numerical optimization of computer models. Wiley, Chichester (1981)
18. Beyer, H.G., Schwefel, H.P.: Evolution strategies - a comprehensive introduction. Nat. Comput. 1(1), 3–52 (2002)
19. Bäck, T.: Self-adaptation in genetic algorithms. In: Proceedings of the First European Conference on Artificial Life, pp. 263–271. MIT Press, Cambridge (1992)
20. Collet, P., Lutton, E., Louchet, J.: Issues on the optimisation of evolutionary algorithm code. In: Proc. CEC 2002 (2002)

Comparative Analysis of Search and Score Metaheuristics for Bayesian Network Structure Learning Using Node Juxtaposition Distributions

Yanghui Wu[1], John McCall[1], and David Corne[2]

[1] IDEAS Research Institute, Robert Gordon University, Aberdeen, UK
{y.wu3,j.mccall}@rgu.ac.uk
[2] School of Mathematics and Computer Science, Heriot-Watt University, Edinburgh, UK
dwcorne@macs.hw.ac.uk

Abstract. Learning Bayesian networks from data is an NP-hard problem with important practical applications. Metaheuristic search on the space of node orderings combined with deterministic construction and scoring of a network is a well-established approach. The comparative performance of different search and score algorithms is highly problem-dependent and so it is of interest to analyze, for benchmark problems with known structures, the relationship between problem features and algorithm performance. In this paper, we investigate four combinations of search (Genetic Algorithms or Ant Colony Optimization) with scoring (K2 or Chain). We relate node juxtaposition distributions over a number of runs to the known problem structure, the algorithm performance and the detailed algorithmic processes. We observe that, for different reasons, ACO and Chain both focus the search on a narrower range of orderings. This works well when the underlying structure is compatible but poorly otherwise. We conclude by suggesting future directions for research.

Keywords: Ant Colony Optimization, Genetic Algorithm, Bayesian Network Structure Learning, Node Ordering, Chain Model, K2.

1 Introduction

Bayesian networks (BNs) are probabilistic graphical models which are used to represent knowledge about uncertain domain. The network consists of a directed acyclic graph (DAG) whose nodes represent random variables, and whose edges represent the direct dependencies between these variables, and a joint probability distribution (JPD) over the random variables. The JPD factorises according to the DAG structure. In many domains, the BN structure and parameters must be learned from data. Learning BN structure is a NP hard problem. It is known that the number of possible structures grows super-exponentially with the number of nodes [1], and so evaluating all possible structures is infeasible in most practical domains, where the number of variables is typically large. The process of finding cheaper approaches for learning the structure of BNs from large datasets is now a very research active area.

A well-established approach to learning BN structure uses metaheuristic search on the space of node orderings combined with deterministic construction and scoring of a network. The comparative performance of different search and score algorithms is highly problem-dependent and so it is of interest to analyze, for benchmark problems with known structures, the relationship between problem features and algorithm performance. In this paper, we investigate combinations of two metaheuristic search techniques, Genetic Algorithms (GA) and Ant Colony Optimisation (ACO) with two scoring approaches, K2 and Chain. All are previously published algorithms for which empirical trade-offs between computational expense and structural accuracy with a high degree of problem dependency have been observed [2,3,4].

In this paper, we attempt to understand this problem dependency. We explore the distributions of nodes juxtapositions in the best solutions found over a number of runs and relate this to the known problem structure, the algorithm performance and the detailed algorithmic processes.

The remainder of this paper is organized as follows: in section 2, we briefly describe search and score approaches for BN structuring learning. In section 3 we describe experiments used to generated node juxtaposition distributions. Results are discussed in section 4, and conclusions presented in section 5.

2 Background

2.1 Bayesian Network Structure Learning Using Search and Score

Search and score approaches attempt to search for the BN structure which best fits the data according to a scoring function. A range of well-known search techniques have been applied in search and score, including Hill Climbing [5], Genetic Algorithms [6], Simulated Annealing [7], Particle Swarm Optimization (PSO) [8], and Ant Colony Optimization (ACO) [9,10]. The most common scoring functions used in these algorithms include the K2-CH metric [11], BDeu [12], BIC [13], and Minimum Description Length (MDL) [14].

2.2 K2 Algorithm and K2-Based Search and Score

K2 is a well-known greedy algorithm that constructs and evaluates a BN from a database of cases [11]. K2 assumes that an ordering on the variables is available and that, a priori, all structures are equally likely. Moreover, it assumes a maximum number of parents a node can have. It starts by assuming that all nodes in the DAG are without parents (i.e. no edges). At each step, edges are added where doing so increases the joint probability of the resulting structure. K2 stops when no further edges can be added.K2 can thus be used as a scoring approach by applying it to an ordering selected by a metaheuristic search. Several K2-based search and score algorithms have been proposed. Here, we briefly introduce two relevant to this paper: K2GA and K2ACO. K2GA [2] uses a GA to search the space of node orderings. The fitness of each ordering is evaluated by running the K2 search algorithm on each ordering evaluated and returning the score of the network structure found. Standard ordering based operators are applied. Similarly, in K2ACO [4], node orderings are generated by a colony of ants. Evaluations from K2 are then used for pheromone update.

2.3 Chain Based Search and Score

Chain scoring for BN structure learning is first proposed in [3]. It is based on the hypothesis that an initial search phase of evaluating fixed chain structures imposed on orderings provides a sufficiently good scoring function to locate high scoring regions of the space of node orderings. A second phase then follows where K2 is applied directly to the best orderings found. Given a node orderingX_1,X_2,...,X_n, we define the chain structure by adding edges between successive nodes. Thus X_i is the sole parent of X_{i+1}. E_i is the edge from X_i to X_{i+1} Figure 1.

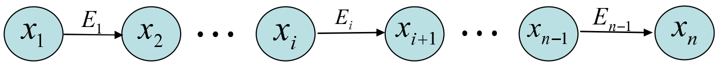


Fig. 1. Chain structure on an ordering

In our previous work, GA (ChainGA) [3] and ACO (ChainACO) [4] are developed as Chain-based search heuristics. At each evaluation step, a chain structure of the given ordering is constructed and evaluated using the K2-CH [11] score metric evaluations. At the end of evaluation, the ordering corresponding to the best fitness score is then produced for K2 algorithm to construct the BN structure. This is a relatively cheap evaluation in terms of the number of K2-CH factor evaluations needed. Our previous results have shown that the Chain structure model can get a significant reduction in computational cost for large data sets. The pseudocode for ChainACO and ChainGA is given in Tables 1 and 2 respectively.

3 Experiments

The aim of our experiments is to investigate the behaviour of GA and ACO metaheuristics searching the space of node orderings using the Chain and K2 evaluation methods. We try to explore the relationship between arcs derived from node juxtapositions in the best orderings found (Figure 1) and arcs in the original structure. We therefore make runs of each metaheuristic with each evaluation method, which we denote ChainACO, ChainGA, K2ACO and K2GA. Note that in these experiments, we only require to run the first phase of ChainACO and ChainGA algorithms to obtain the best node orderings, as the search of ordering

Table 1. Pseudocode of ChainACO Algorithm

```
Initialize pheromone
Initialize heuristic information
Loop
   Each ant is positioned on a starting node
   Loop
     Each ant applies a state transition rule to incrementally
     build a solution and a local pheromone updating rule
   Until all ants have built a complete solution
   A global pheromone updating rule is applied
Until termination criterion is met
Implement K2 Algorithm on best solution to learn the best
structure.
```

Table 2. Pseudocode of ChainGA Algorithm

```
Initialize population
Repeat
   Select best-ranking individuals to reproduce
   Apply crossover operator
   Apply mutation operator
Until termination criterion is met
Implement K2 Algorithm on best solution to learn the best
structure.
```

space ends at this point - the second phase is deterministic. However we are not comparing the computational efficiency or the scores of final networks produced as that has been covered in our earlier work [3,4].

Four well known benchmark problems have been selected in our research: Asia, Car, Insurance and Alarm. The Asia network is a simple network with 8 binary nodes and 8 edges. It is a diagnostic demonstrative Bayesian network [15]. The Car Diagnostic Network consists of 18 nodes and 17 edges. It can be applied to diagnose malfunctioning of self-propelling vehicles [3] . The Insurance network contains 27 nodes and 52 arcs, is a network for evaluating car insurance risks [16] . The Alarm network is a medical diagnostic system for intensive care patient monitoring consisting of 37 nodes and 46 edges [17]. All the data cases are sampled using the Netica tool [18]. In this paper, the dataset sizes for Asia, Car, Insurance and Alarm are 5000, 10000, 5000 and 3000 cases respectively.

In all cases, the scoring metric used to evaluate the node ordering is the K2-CH metric. For ChainACO and ChainGA we carry out 200 experimental runs each. For K2ACO and K2GA we carry out only 50 runs each due to time complexity. The parameters used for ACO and GA based algorithms in this paper are the same as those used in [3,4].

4 Results and Discussion

Table 3 presents the distribution of node juxtapositions recorded from the best ordering found in each of 200 experimental runs of ChainACO on the Asia network. The row index indicates the first node in a juxtaposition, the column index indicates the second. For example, the Table shows that in 65 of the runs, the node juxtaposition 1-2 appeared 65 times, and the node juxtaposition 2-1 appeared 135 times. This means that, in all runs of ChainACO on Asia, nodes 1 and 2 were juxtaposed in the best ordering found, with a 135:65 preference for node 2 preceding node 1. In all of these cases, the ordering will have been evaluated using a chain structure inserting a directed edge between these nodes. It is not of course necessary that any particular juxtaposition will appear in all experimental runs. Each ordering found will contain $n-1$ juxtapositions where n is the number of nodes. The sum of entries in the ordering distributions table will therefore in general be $r \cdot (n-1)$, where r is the number of runs. In this case, the entries sum to $1400 = 200 \times 7$.

Table 3. Node Juxtaposition Distribution for 200 runs of ChainACO on Asia

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	0	65	0	135	0	0	0	0
2	135	0	19	21	9	7	0	0
3	10	76	0	3	12	4	0	4
4	44	0	4	0	0	145	0	0
5	0	9	65	0	0	0	2	31
6	0	3	0	37	0	0	158	2
7	0	0	0	0	0	42	0	158
8	0	2	3	0	155	0	40	0

It is noticeable from Table 3 that the distribution of node juxtapositions is concentrated on a relatively small subset of possible node juxtapositions. This indicates that ChainACO is highly consistent in the node orderings it produces and suggests a strong convergence property of the search. In Figure 2, we present a visual representation of the node distributions produced by all four algorithms on Asia and Car. Here, the instance counts have been replaced by a normalized grayscale representation running from white (juxtaposition occurs on 0% of runs) through to black (juxtaposition occurs on 100% of runs). It is easy to observe from Figure 2a that there is a marked difference in distribution between ChainACO (top-left) and K2GA (bottom right). ChainACO produces a high contrast image consisting of mostly very dark or very light pixels whereas the K2GA image is much more diffuse. It is hard to visually detect much of a difference in contrast between K2ACO and ChainGA other than that they lie somewhere in between the other two. Moreover the dark areas for ChainACO do not particularly coincide with those for K2GA. However results in [4] show that each algorithm reliably reproduces the Asia network.

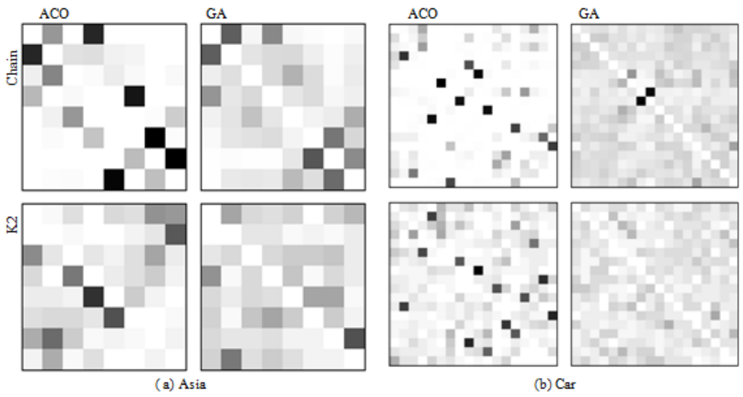


Fig. 2. Grayscale Grids of the Edges Occurrences in Asia and Car Networks within the Four Algorithms

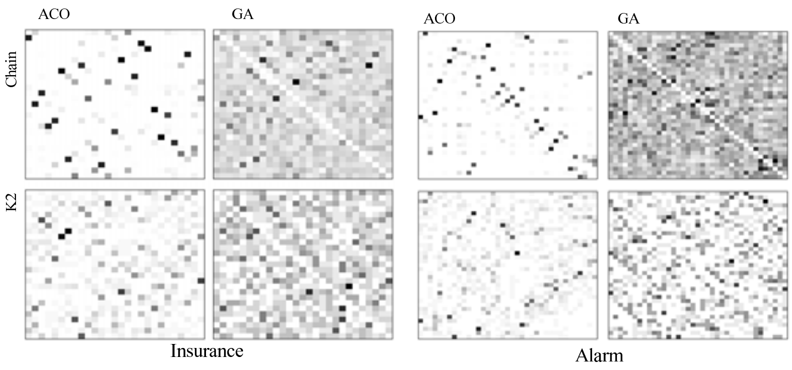


Fig. 3. Grayscale Grids of the Edges Occurrences in Insurance and Alarm Networks within the Four Algorithms

In Figure 2b, there are more possible node juxtapositions and the visual contrast is more marked. In order from highest to lowest visual contrast, the images are ordered ChainACO, K2ACO, ChainGA and K2GA. This ordering is consistent with a hypothesis that both the Chain scoring approach and the ACO metaheuristic result in more concentrated distributions than the K2 scoring approach and the GA metaheuristic respectively. Finally, the equivalent diagrams for Insurance and Alarm are shown in Figure 3. As these networks have many more possible node juxtapositions the diagrams have a finer granularity but the same effects are observable.

We present node juxtaposition frequencies for all four algorithms for the Insurance and Alarm problems as Box plots in Figure 4. These essentially show the same information. Here the effect of the Chain approach manifests as a low

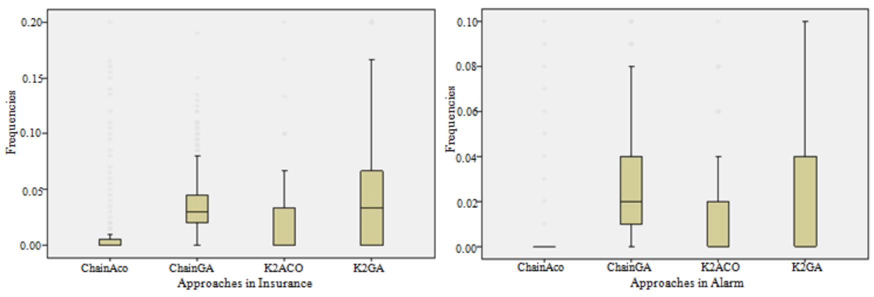


Fig. 4. Comparison of Frequencies of Each Edges Found in Insurance and Alarm Networks within four Algorithms

median frequency for most possible juxtapositions with a small number of high frequency outliers. This is particularly noticeable for ChainACO in both Figures. Conversely, for K2 approaches, and K2GA in particular, there is a higher median node distribution frequency and a large distribution of frequencies in the interquartile range, corresponding to the more diffuse visual pattern observed earlier.

Overall, there appears to be a small reduction in variability of the final ordering produced deriving from the use of ACO rather than GA, but the dominant difference in behaviour derives from the choice of scoring approach. We observe that the Chain scoring approach concentrates the search on a smaller set of node juxtapositions, and hence node orderings than the K2 scoring approach does. This is because, for any particular ordering, Chain only inserts edges between juxtaposed nodes whereas K2 may insert an edge between any two nodes. Thus it is possible to discover valuable interactions from a wider range of orderings with K2 than with Chain. Conversely, it takes longer to evaluate orderings with K2 because a large number of possible edges have to be considered in turn for each ordering. Therefore, the relative merits of Chain and K2 for any particular problem lie in how amenable the dependencies in the data are to discovery using the Chain approach.

Figure 5 and Figure 6 are diagrams of the known true structures for Asia and Car respectively, annotated with the best ordering found by ChainACO. For each node juxtaposition occurring in the best ordering that corresponds with an edge in the true structure, an arrow is added in the middle of the edge in the direction of the node juxtaposition. If there is no edge in the true structure corresponding to the node juxtaposition, a dotted arrow is added to the diagram. Solid directed edges with no central arrow therefore represent edges that occur in the true structure but are not represented by node juxtapositions in the best ordering found. We also annotate each node juxtaposition with the overall percentage of runs in which it appeared in the best ordering for that run.

Figure 5 shows that all but one of the node juxtapositions in the best ordering found by ChainACO for Asia coincide with true arcs. Of these three out of six are correct and three reversed. The solution adds one node juxtaposition

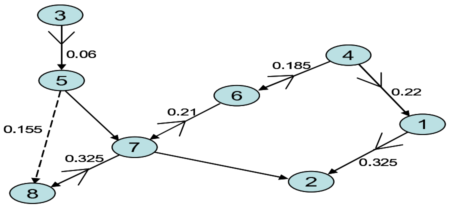


Fig. 5. Asia - annotated by best ChainACO ordering 3-5-8-7-6-4-1-2

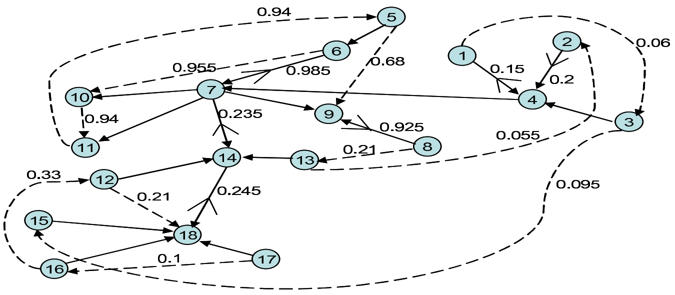


Fig. 6. Car - annotated by best ChainACO ordering 17-16-12-18-14-7-6-10-11-5-9-8-13-2-4-1-3-15

corresponding to a spurious arc and omits two arcs. However the nodes corresponding to the two omitted arcs are correctly ordered and so could be discovered by K2 in the second phase of the algorithm. This analysis shows that it is possible to create chains closely aligned to the structure. This explains why ChainACO and ChainGA perform well on Asia.

Figure 6 shows that only six of the seventeen node juxtapositions in the best ordering found by ChainACO for Car coincide with true arcs. Of these only one out of six is correct and five are reversed. The solution adds eleven node juxtapositions corresponding to spurious arcs and omits ten arcs. It is noticeable that the true Car structure contains nodes such as 4, 7, 14 and 18 each of which is a hub for a cluster of tightly-bound nodes. The binding between these clusters is loose. Such a topology is not amenable to the construction of chains where many node juxtapositions correspond to a true arc independent parents. For example, only one node can be positioned before node 18, which immediately excludes at least three true arcs in any ordering. Therefore in this case, the Chain approach finds spurious arcs that can coexist as node juxtapositions in a single ordering and give a better score than orderings that include correct node juxtapositions. This inherent difficulty in aligning chains of node juxtapositions with the true structure explains why ChainACO and ChainGA perform poorly on Car.

Table 4 contains for each problem and for the best node ordering found over all runs by ChainACO, the numbers: C, of node juxtapositions corresponding to correct arcs; R to reversed arcs; A to additional, spurious, arcs; and O is the

Table 4. Alignment of best ChainACO ordering with true structures

	C	**R**	**A**	**O**	**T**
Asia	3	3	1	2	8
Car	1	5	11	11	17
Insurance	10	8	8	34	52
Alarm	14	6	15	26	46

number of arcs in the true structure to which no node juxtapositions correspond; T represents the total arcs in true structure. The table shows that C and R dominate for those problems (Asia and Alarm) where ChainACO performs well. On the other hand, A and O dominate for those problems (Car and Insurance) where ChainACO performs poorly.

5 Conclusions

In this paper we have conducted experiments to investigate the behaviour of GA and ACO metaheuristics searching the space of node orderings using the Chain and K2 evaluation methods. We have explained problem-dependent performance trade-offs between cost and structure quality in terms of the relationship between the Chain scoring mechanism, which relies on ordering node juxtapositions and true arcs in the original structure. In all problems investigated, the Chain scoring approach focused the search on a narrower range of orderings than did the K2 scoring approach. A lesser effect was also observed in that ACO-based methods appeared to concentrate search more than GA-based methods.

The major conclusion of our analysis of node juxtaposition statistics is that in problems where the true structure of the data is amenable to alignment of node juxtapositions in a single ordering, the Chain scoring approach is able to yield high quality solutions with significantly less computational effort than the K2 scoring approach. In problems where such alignment is not possible, the Chain scoring approach is likely to be unsuccessful in producing high quality structures and so the relative benefit of reduced computational time is lost.

Finally our results suggest a possible direction for future work. A generalization of the Chain scoring approach that could detect shorter series of well-aligned node juxtapositions combined with a coarser-grain version of the K2 algorithm could potentially assemble high quality structures at reduced cost, even in situations where a single ordering would not admit a set of well-aligned arcs. ACO is a promising approach for more generalized construction approaches.

References

1. Robinson, R.W.: Counting labeled acyclic digraphs. In: Harary, F. (ed.) New Directions in the Theory of Graphs, pp. 239–273. Academic Press, New York (1973)
2. Larraaga, P., Kuijpers, C., Murga, R.: Learning Bayesian network structures by searching for the best ordering with genetic algorithms. IEEE Transactions on System, Man and Cybernetics 26, 487–493 (1996)

3. Kabli, R., Herrmann, F., McCall, J.: A chain-model genetic algorithm for Bayesian network structure learning. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, IEEE World Congress on Computational Intelligence 2007, London, England, July 7–11 (2007)
4. Wu, Y., McCall, J., Corne, D.: Two Novel Ant Colony Optimization Approaches for Bayesian Network Structure Learning. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010 (2010)
5. Tsamardinos, I., Brown, L.E., Aliferis, C.F.: The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning 65(1), 31–78 (2006)
6. Larraaga, P., Murga, R., Poza, M., Kuijpers, C.: Structure learning of Bayesian networks by hybrid genetic algorithms. In: Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics, pp. 310–316 (1995)
7. Wang, T., Touchman, J., Xue, G.: Applying two- level simulated annealing on Bayesian structure learning to infer genetic networks. In: Proceedings of the IEEE Computational Systems Bioinformatics Conference, pp. 647–648 (2004)
8. Cowie, J., Oteniya, L., Coles, R.: Particle Swarm Optimization for learning Bayesian Networks. In: Proceedings of the World Congress on Engineering 2007, WCE 2007, London, UK, July 2-4, vol. I (2007)
9. de Campos, L.M., Huete, J.F.: A new approach for learning Bayesian networks using independence criteria. International Journal of Approximate Reasoning 24, 11–37 (2000)
10. de Campos, L.M., Gmez, J.A., Puerta, J.M.: Learning Bayesian network by ant colony optimization: Searching in two different spaces. Mathware and Soft Computing IX (2-3), 251–268 (2002)
11. Cooper, G.F., Herskovits, E.: A Bayesian Method for the Induction of Probabilistic Network from Data. Machine Learning 9(4), 309–347 (1992)
12. Buntine, W.: Theory refinement on Bayesian networks. In: UAI-17, pp. 52–60. Morgan Kaufmann, San Francisco (1991)
13. Schwartz, G.: Estimating the dimensions of a model. Ann. Statist. 6(2), 461–464 (1978)
14. Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian networks: The combination of knowledge and statistical data. In: KDD Workshop, pp. 85–96 (1994)
15. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society 50(2), 157–224 (1988)
16. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning 29(2), 213–244 (1997)
17. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Proceedings of the Second European Conference on Artificial Intelligence in Medical Care, pp. 247–256. Springer, Berlin (1989)
18. Netica. Netica Bayesian network software from Norsys, http://www.norsys.com

Analyzing the Credit Default Swap Market Using Cartesian Genetic Programming

Laleh Zangeneh and Peter J. Bentley

Department of Computer Science
University College London, London WC1E 6BT, UK
{L.Zangeneh,P.Bentley}@cs.ucl.ac.uk

Abstract. The credit default swap has become well-known as one of the causes of the 2007-2010 credit crisis but more research is vitally needed to analyze and define its impact more precisely and help the financial market transparency. This paper uses cartesian genetic programming as a discovery tool for finding the relationship between credit default swap spreads and debts and studying the arbitrage channel. (Arbitrage is the practice of taking advantage of a price difference between markets.) To our knowledge this work is the first attempt toward studying the credit default swap market via an evolutionary process and our results prove that cartesian genetic programming is human competitive and it has the potential to become a regression discovery tool in credit default swap market.

Keywords: Cartesian Genetic Programming, Credit Default Swap, Regression.

1 Introduction

In the last two years, the world economy has been faced with one of the biggest crises ever seen, throwing most countries into recession. The causes of the financial meltdown are numerous, but it is widely accepted that one significant factor was the "Credit Default Swap" market. Trading of this complex financial product was unpredictable, out of control, and badly priced, leading to fortunes being made and lost [11].

To understand what a Credit Default Swap (CDS) really is, consider the following example. Imagine North bank made a five-year $10 million loan to West Airways. North bank is concerned about West Airways performance and not being able to pay back the loan (possible default). Therefore, in order to protect itself and reduce the risk of not getting its loaned money back, North bank can buy a kind of insurance (known as "protection") on West Airways from a insurance seller (a protection seller), which in this case might be East bank. The insurance is based on a West Airways-issued bond (a debt security which represents a formal contract to repay borrowed money with interest at fixed intervals). This protection (insurance) contract is called a CDS contract and East bank is then able to trade its CDS contracts with other banks, buying them when they cost less and selling them when they are worth more, in order to make profit. The price of CDS contract changes according to the success or failure of the business of West Airways (i.e., the credit quality of West Airways). If the West Airways credit quality decreases (risk of default increases), the CDS price will increase.

One of the major reasons why this product help cause the financial crisis is because CDS contracts were often very poorly costed-although they were supposed to represent a kind of insurance against a loan, their prices often showed little relationships to the true ability of the companies to repay those loans. Thus when companies unexpectedly defaulted on their loans (or unexpectedly paid the loans back), a bank that bought a CDS contract at a very high price might suddenly find it was worth very little, and it would lose money. Thus the pricing of CDS contracts is of enormous concern and consequence. In this paper, the first ever study is performed into CDS pricing by using Cartesian Genetic Programming to analyse the relationship between price, debt and equity information. We show that CGP can completely outperform the standard pricing model, and we provide some analysis of the CGP solution, as well as the ability of CGP to cope with this complex financial data.

More details of CDS contract pricing is provided in the following section. This is followed by a description of our CGP model in section three. Section four explains the datasets, CGP settings and experiment objectives. Results are provided in the fifth section and we conclude in section six.

2 Credit Default Swap Background

A CDS contract is a kind of credit derivative. Credit derivatives are over-the-counter (OTC)[1] financial contracts that allows one to take or reduce credit exposure, commonly on bonds or loans of corporate entity and it reflects the risk of a default in a corporation. This risk is expressed through the CDS price. A CDS is an agreement between two parties to exchange the credit risk of a reference entity, also called an issuer (West Airways, in our previous example), without directly involving the issuer [1]. The protection buyer (North bank, in our example) pays a periodic fee and receives compensation if the reference entity has a credit event. A credit event includes bankruptcy, failing to pay outstanding debt obligations, or restructuring of a bond or loan. The protection seller (East bank) collects the periodic fee and profits if the credit of the reference entity remains stable or improves while the swap is outstanding [2]. Figure 1 illustrates the terminology and mechanism of the CDS. The CDS is uniquely defined by four key parameters [3,2,1]:

1. **Issuer:** CDS contracts specify a reference bond or loan which defines the issuing entity through the bond prospectus (e.g. West Airways).

[1] The phrase "over-the-counter" can be used to refer to stocks, debt securities and other financial instruments such as derivatives, which are traded through a dealer as opposed to on a centralized exchange (e.g. London Stock Exchange).

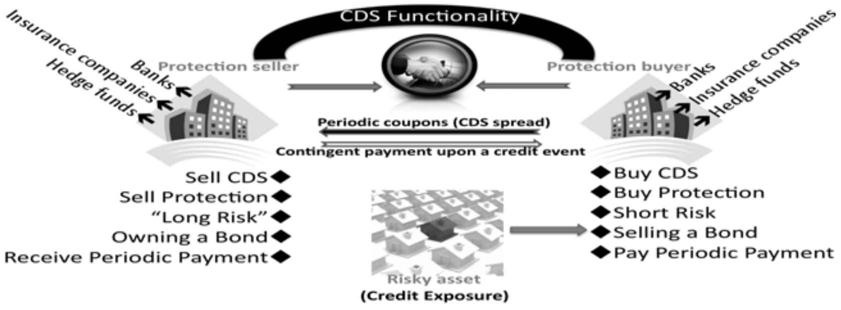


Fig. 1. Single Name CDS Functionality

2. **Notional amount:** Notional amount is the amount of credit risk being transferred between protection buyer (North bank) and protection seller (East bank).
3. **Spread:** A spread (also called coupon, or price) specifies the annual payments which are quoted in basis point[2]. These payments are paid quarterly (e.g. from North bank to East bank).
4. **Maturity Date:** The expiration of the contract. The most liquid[3] maturity term for CDS contract is 5 years.

Over the last few years, the credit derivatives market has grown significantly and exceeds both equity derivatives and corporate bond markets. The largest participants in the credit derivatives market are banks, insurance and securities companies. According to the British Bankers Association (BBA) report, the most important and widely used products in credit derivatives are CDS (42% of notional principal outstanding in 2006) [2]. The reality shows that the CDS market suffers from a lack of any comprehensive study. The lack of sufficient data had been a major problem for a broad empirical testing of CDS pricing models (as seen in [8,7]) until few years ago.The last few years with increased bond market liquidity and a well-developed CDS market provide more sufficient data for investigation.

2.1 CDS Pricing Challenge

In the credit risk literature, there are two broad approaches to modelling corporate default risk (e.g. the risk that West Airways defaults on a loan): the structural approach and the reduce-form model. In the structural model the evolution of the company's assets follows the diffusion process. In other words the default occurs when the value of the firm assets becomes lower than its debts; because the assets can be continuously assessed, downwards trends can be spotted and so the risk of default should never be a surprise. In contrast to the

[2] A basis point (often denoted as bp) is a unit relating to interest rates that is equal to 1/100th of a percentage point per annum (pa).

[3] Liquid means easily converted into cash (e.g. a bond which can be sold quickly).

Table 1. History Track of Credit Risk Approaches

Structural Model		**Reduced-form Model**	
Investigators	Date	Investigators	Date
Black & Scholes	1973	Geske, Ingersoll, Merton	1977
Merton	1974	Smith & Warner	1979
Black & Cox	1976	Cooper & Mello	1991
Longstaff & Schwarts	1995	Hull & White	1992
		Abken	1993
		Duffie & Singleton	1995

structural approach, the reduce-form approach assumes that there is no relation between value of the company and risk of default. In this approach defaults are seen as an unpredicted Poisson events involving a sudden loss in market value and therefore firms never default gradually. See [11,4,6,5] for more details on credit risk literature. Table 1 presents the history of credit risk approaches by referring to the investigators who contributed to the field.

As discussed, default risk is expressed through the CDS spread. While pricing of this CDS is a challenging open problem that is a very quantitative and qualitative field involving estimations of default, timing of default and balance sheet value fluctuations (see: [8,9,12]), the Duffie approach provides a method to evaluate the correct pricing of the CDS spread through the simple relationship.

$$CDSspread = RiskFreeRate - DebtReturn \tag{1}$$

Where Risk Free Rate refers to Interest Rate and Debt Return refers to Bond Yield. This paper uses Duffie regression model as the regression benchmark model. This relationship is observed in the market and if it breaks down significantly, traders will buy and sell the instruments to return the relationship close to parity. But observing the CDS spread, debt return and risk free rate in the financial market shows that this relation does not hold exactly. An example of this observation is illustrated in figure 2. The reason of the gap between the Duffie theory and market data is the cost of arbitrage or what is also known as the arbitrage channel [1]. The arbitrage cost comes from the range of market mechanics to borrow, sell and buy instruments to profit from the CDS spread, inaccurately estimating the risk of a default event [8]. Therefore, the challenge of the CDS pricing is narrowed to studying of this arbitrage channel and reduce it in order to find a regression model which can match the market data. This paper is focussed on investigating the relationship between CDS spread, debt and equity

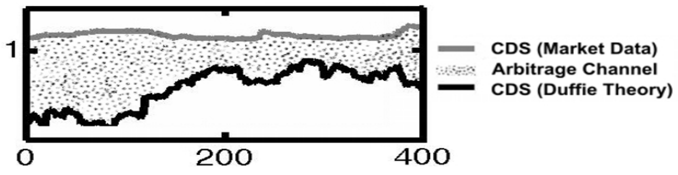


Fig. 2. Duffie Theory Vs. Market Data

information using CGP. In analyzing the results of CGP for a common group of financial corporations we expect to discover and propose a new regression model which can estimate the CDS price better than current regression benchmark.

3 CGP Model

CGP was developed from methods developed for the automatic evolution of digital circuits by Miller and Thomson [10] for the purpose of evolving digital circuits. Unlike traditional Genetic Programming (GP), CGP represents a program as a directed (that for feed-forward functions is acyclic) graph. The significance of the difference between CGP and Linear GP has been established in the means of restricting interconnectivity of nodes [16]. In CGP, the genotype is a fixed length representation and consists of a list of integers, which encode the function and connections of each node in the directed graph [15]. The number of nodes in the graph is bounded but it can be varied, as CGP uses a genotype phenotype mapping that allows the existence of unconnected nodes in the genotype which produce inactive sub-genotypes that have no effect on the phenotype. This leads to an effect on search called neutrality, a CGP feature that has been found to be tremendously valuable to the evolutionary process on the problems studied [14]. Each of the nodes is encoded by a number of genes representing a particular function and the inputs that each node has. The nodes take their inputs in a feed forward manner from either the output of a previous node or from one of the initial program inputs (terminals).

CGP has been applied to a growing number of domains and problems such as digital circuit design, digital filter design, image processing, artificial life, bio-inspired developmental models, evolutionary art and has been adopted within new evolutionary techniques such as cell-based optimization and social programming. To our knowledge it has not been investigated in the financial field so far while GP is widely used in financial fields such as: Stock markets, Game theory, Betting, Foreign exchange, Arbitrage and Studying markets. See [13] for comprehensive overview of GP and its applications.

3.1 CGP Model Modification

For the purpose of this paper objectives the basic CGP model[4] is modified in order to provide more information from CGP result. Our CGP model has the following features:

1. **Training/Test Dataset:** Our version of CGP divides the data X_a into two datasets where X_i randomly chosen data points are considered as the test set and the X_{a-i} remaining data points are considered as the training set. The training set used by CGP as input data, and the best result at the end of the evolutionary run is tested.

[4] Visit http://www.cartesiangp.co.uk for CGP related information and CGP model.

2. **CGP Output:** The original CGP model provide fitness report as the CGP output. For the purpose of analysis our version of CGP reports on training and test sets results which makes the data comparison possible.
3. **Chromosome Translator:** An important feature of this work is the evolved equation (not just the fitness values). Hence we also have created a solution parser that translates the chromosome into an understandable mathematical equation, which we can then study for insights into the solution.

4 Experimental Datasets, Settings and Objectives

The Centrica Plc company is chosen for our experiments. The Centrica Plc is a large multinational utility company. It is listed on the London Stock Exchange and also listed on FTSE 100 Index[5]. Data is collected from 5th January 2004 till 25th Jun 2009 (which of course includes the recent highly turbulent nature of the markets). Table 2 illustrates a sample of our database including the company CDS spread, debt and equity information. Two datasets are specified for the system. The first dataset contains three inputs: CDS spread (bp), bond yields (ask and bid price) and Bank of England base rate. The second dataset includes the all available information, eight inputs as shown in table 2. In the rest of this paper we refer to the first dataset as the CDS-Debt dataset and the second dataset is called CDS-Debt-Equity dataset. Each dataset contains 1400 data points (X_a) where 400 randomly chosen data points (X_i) are considered as the test set and the 1000 remaining data points (X_{a-i}) are considered as the training set. The test set is the same for all runs.

Table 2. Centrica plc Database (CDS, Debt and Equity Information)

Date	Spread(bp)	Bid yield	Ask yield	Base rate	Bid PX	Ask PX	E. Volatility	E. Weight	E. PX(High)
05/01/2004	0.2900	5.448	5.376	3.75	102.908	103.408	7.8840	187.7381	188.81
. . . .	. . . .	. . . .	. . . .	. . . .	. . . . .	. . . . .	. . . . .	. . . . . .	. . . .
24/06/2009	0.6667	4.172	3.990	0.50	105.188	105.768	28.996	226.5237	230.75
25/06/2009	0.6652	4.157	3.972	0.50	105.232	105.822	28.840	227.5276	231.00

Table 3 shows the experimental setup. We run all experiments with the same settings. We vary the mutation rates and the number of nodes (which in CGP affects the overall size of solutions and thus the complexity of equations that can be evolved) in order to monitor CGP behaviour. A simple function set is chosen, containing only fundamental operators as listed in table 3. In addition to our financial inputs, three constant integers (1,2 and 3) have been given as constant inputs to the model as well.

The fitness is calculated for each datapoint by defining the error rate, calculated as the absolute value of difference between the CGP–Output and the actual data: $Error = |CGP_{Output} - Data|$ and converting this result to a number between 0 and 1 where this number demonstrate the portion of the number of actual values that is predicted correctly by CGP: $DataPointFitness = \frac{1.0}{1.0+Error}$. The

[5] FTSE 100 index is a share index of the 100 most highly capitalised UK companies listed on the London Stock Exchange.

Table 3. CGP Settings

General Setting		Function	Symbol
Population size:	5	Add	+
Mutation rate:	0.20, 0.50, 0.70	Subtract	-
No. of generations:	200000	Multiplication	*
No. of runs:	20	Division	/
No. of rows:	1	Power	Pow
No. of cols:	250 or 500	Square root	Sqrt
Levels back:	250 or 500		

fitness of the whole dataset is equal to sum of all data points' fitnesses and the best dataset fitness is equal to number of data points. Thus, a higher fitness means a better result as it shows the smaller error rate. We follow two main objectives in our experiments:

1. **Monitoring CGP behavior under different settings.** Therefore, the first experiments are ran on the CDS-Debt dataset with different combinations of nodes (500,250) and mutation rates (0.20, 0.50, 0.70) to see how these two factors will affect the results. Following these experiments, the ability of CGP to deal with and distinguish between relevant and irrelevant inputs is examined by using the second dataset (CDS-Debt-Equity dataset) containing more data attributes.
2. **Assessing the CGP reliability as regression discovery tool.** Of interest, is to observe, whether CGP can come up with a regression model that can price CDS better than the regression benchmark model (Duffie Theory) and to understand something of how that model works.

Each experiment was run for 20 times and 200,000 generations.

5 Results

Figure 3 shows the CGP fitness report on CDS-Debt dataset. According to the result, although the number of nodes (graph C and D) and mutation rate (graph A and B) affect the CGP performance in terms of reaching a better fitness in early generations, but it does not have a big impact on the average fitness. This means the better fitness dose not always rely on a larger number of nodes and higher mutation rate (graph B and C). For CDS-Debt dataset, the best fitness was accived by 500 nodes and mutation rate of 0.50.

As we discussed in section 2 one of the important issues of CDS pricing is to reduce the arbitrage channel. Figure 4 and table 4 show the results in terms of accuracy of CDS pricing. In our experiments, CGP discovered a new relationship between bond yield and risk free rate which creates a very accurate prediction of the real CDS price in the market. It also demonstrates the arbitrage gap which exists in Duffie theory is significantly reduced (see figure 4a and 4b). The result shows that the trend of CDS price has been predicted correctly.

In the experiment using the larger dataset (CDS-Bond-Equity dataset), the results show that the extra inputs helped CGP to reduce the arbitrage channel

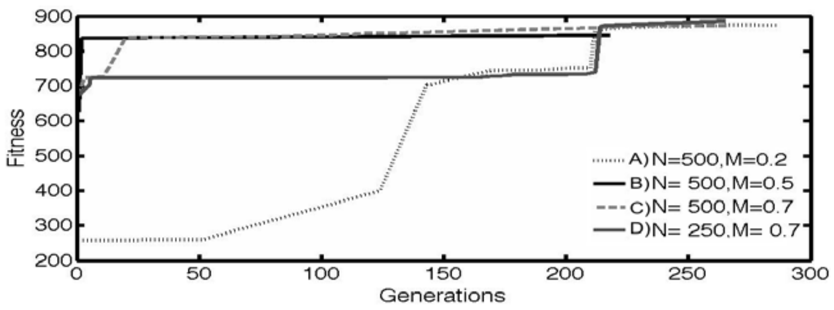
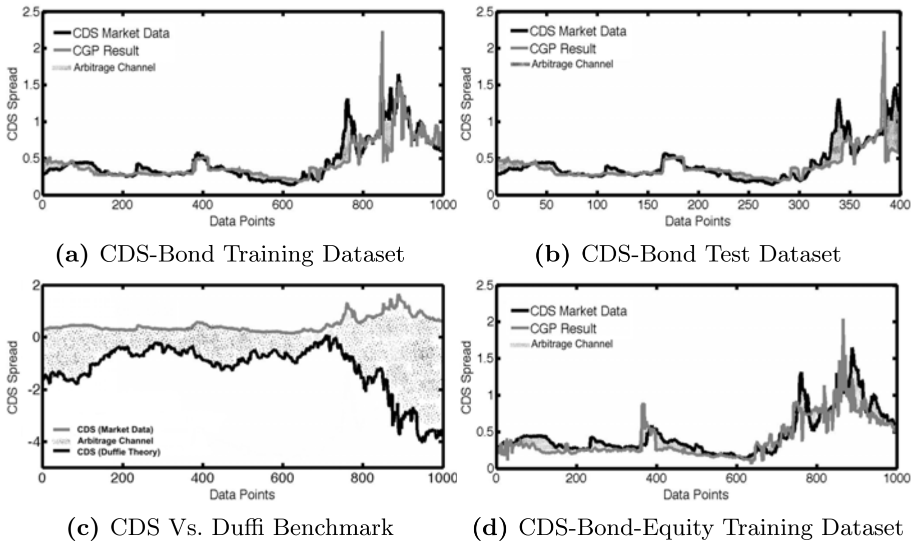


Fig. 3. CGP Behavior Under Different Settings



(a) CDS-Bond Training Dataset **(b)** CDS-Bond Test Dataset

(c) CDS Vs. Duffi Benchmark **(d)** CDS-Bond-Equity Training Dataset

Fig. 4. Experiments Result

in some parts but it had a negative effect on other parts (see figure 4c and 4d), so the overall error increased, see table 4.

The inability of CGP to perform effective feature selection using this larger number of attributes may be partly because of the complexity of this problem. Some of the additional variables may be useful some of the time and detrimental at other times, meaning that CGP (and indeed any evolutionary algorithm)

Table 4. Error Report

Model	Input	Training Set Error	Test Set Error
Duffie	CDS-Bond	167.84928%	134.6789%
CGP	CDS-Bond	8.9769295%	9.7533925%
CGP	CDS-Bond-Equity	10.1807889%	10.0130045%

would find it hard to eliminate them. The complexity of all the relationships, means that an incremental change from a complex solution using more variables into a simpler solution using fewer variables may be impossible without encountering extremely unfit variants, thus making the search unlikely to be successful. Nevertheless, the results are fascinating for they indicate that good accuracy for this problem can be obtained with fewer variables and simpler corresponding equations.

$$\mathbf{CDSspread} = \frac{\left(\frac{-X_1^2 X_3}{3^{X_3}} - 1\right)^{X_2 * (X_2 - X_3)}}{\sqrt{\left|\frac{X_2}{Z_1^{3^{\frac{3}{X_1^2 * (-X_3)}}}}\right|} + \left(2 * D_5^{\sqrt{\left|\left(\frac{S_1}{D_6^{\left(\frac{Z_1}{-X_1^2 X_3}\right)}}\right)^{D_6}\right|}}\right) * (Z_1 - M_1))} \tag{2}$$

Where

$$D_2 = (4 - X_1) * (-X_3^{X_2 - 1} - M_1), \quad D_3 = X_3 - D_2 - 3^{X_3}, \quad M_1 = X_2^3 + \frac{X_2^3}{X_3}, \quad S_1 = X_2 * (X_2 - X_3)^2$$

$$D_5 = \frac{D_3}{\frac{X_1^2}{X_1 - X_2 - 1}} * D_4, \quad D_6 = X_1 - \frac{3}{-X_1^2 X_3}, \quad D_4 = \frac{D_3}{Z_1 * 3^{\frac{3}{-X_1^2 X_3}}} * S_1 \quad Z_1 = \frac{\frac{X_2^3}{X_3}}{X_1^2 - X_3}$$

Not surprisingly, CGP has evolved completely different equation in each run. Equation (2) shows one of the best evolved solutions. Analysis of all evolved equations reveals that some components are repeated in all solutions. For instance the component $(\frac{X_2}{X_3})$ has been found in 12 best solutions. X_2 is buying price and X_3 is selling price of bond yield. Moreover, the component $(X_2 - X_3)$ which shows the difference amount between sell price and buy price of bond yield is several regions of the equations of 7 best solutions. CGP has discovered these possible relationships between X_2 and X_3 (buy and sell prices). To understand the significance of these relationships, we test the affect of these two components by reducing the sell price and buy price difference.

$$\lim_{(X_2 - X_3) \to 0} CDSspread \qquad and \qquad \lim_{(\frac{X_2}{X_3}) \to 1} CDSspread \tag{3}$$

The computational results show that the error rate significantly increases by ignoring the difference amount between X_2 and X_3 but the theoretical regression benchmark ignored these relationships completely by using the average value of buy and sell price or just one of them.

6 Conclusions

CDS pricing is highly significant, not just for finance, but for the world economy. This is one of the first ever investigations into the CDS market using machine learning. In this work we used Cartesian Genetic Programming to derive new relationships between variables in order to produce a dramatically more accurate model for CDS pricing compared to the standard Duffi approach. We demonstrated the effectiveness of this bio-inspired evolutionary method for a complex real-world financial problem. Our data included the highly turbulent behaviour of the markets in the last two years, with no loss of accuracy - a significant improvement over the Duffi method which showed a serious fall in accuracy. We also demonstrated the sensitivity of CGP parameters and showed that CGP was able to provide more consistent results using fewer attributes. Future research will focus on performing more experiments on a comprehensive financial database and more comparison analysis.

Although this may be the first use of CGP in finance, the results are highly significant and revealing. This suggests that other bio-inspired methods designed for noisy, unpredictable and unknown data may also be able to illuminate some of the hitherto murky waters of financial trading. We anticipate with tools such as these, future financial crises may be less likely to occur.

Acknowledgments

We would like to acknowledge Julian F. Miller for patiently supporting us over CGP model and also special thanks to Ajay Rathor for providing financial data and kindly sharing his professional financial knowledge and experience.

References

1. Credit Derivative Credit Derivative Handbook. Merrill Lynch (April 2003)
2. Credit Derivatives Handbook. J.P. Morgan-Corporate Quantitative Research (2006)
3. O'Kane, et al.: The Lehman Brothers Guid to Exotic Credit Derivatives, Risk. Lehman Brothers (2003)
4. Black, F., Cox, J.: Valuing corporate securities: Some effects of bond indenture provisions valuing corporate securities: Some effects of bond indenture provisions valuing corporate securities: Some effects of bond indenture provisions. Finance (1976)
5. Elizalde, A.: Credit default swap valuation: An application to spanish firms (10.1.1.139.5416) (2005)
6. Hofberger, B., Wagner, N.: Pricing cdx credit default swaps using the hull-white model (2007)
7. Houweling, P., Vorst, T.: Pricing default swaps: Empirical evidence. International Money and Finance 24(8), 1200–1225 (2005)
8. Hull, J., White, A.: Valuing credit default swaps: No counterparty default risk. Journal of Derivatives 8(1), 29–40 (2000)
9. Joro, T., Na, P.: Simulation-based first to default(ftd) credit default swap(cds) pricing approach under jump diffusion. In: Proceedings of the 36th Conference on Winter Simulation, pp. 1632–1636 (2004)
10. Miller, J.F., Tomson, P.: Cartesian genetic programming. In: Proc. of the European Conference on Genetic Programming, pp. 121–132 (2000)

11. Navneet, A., Bohn, J., zhu, F.: Reduced form vs. structural models of credit risk: A case study of three models. Investment Managment 3(4), 43–67 (2005)
12. Group of Twenty Finance Ministers of Central Bank Governors. Declaration of the Summit on Financial Markets and the World Economy
13. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: Genetic programming an introductory tutorial and a survey of genetic programming an introductory tutorial and a survey of techniques and applications. Technical Report CES-475 (October 2007), `http://www.essex.ac.uk/csee/research/publications/technicalreports/2007/ces475.pdf`
14. Walker, J.A., Miller, J.F.: Evolution and acquisition of modules in cartesian genetic programming. In: Keijzer, M., O'Reilly, U.-M., Lucas, S., Costa, E., Soule, T. (eds.) EuroGP 2004. LNCS, vol. 3003, pp. 187–197. Springer, Heidelberg (2004)
15. Walker, J.A., Miller, J.F., Cavill, R.: A multichromosome approach to standard and embedded cartesian genetic programming. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 903–910 (2006)
16. Wilson, G., Banzhaf, W.: A comparison of cartesian genetic programming and linear genetic programming. In: O'Neill, M., Vanneschi, L., Gustafson, S., Esparcia Alcázar, A.I., De Falco, I., Della Cioppa, A., Tarantino, E. (eds.) EuroGP 2008. LNCS, vol. 4971, pp. 182–193. Springer, Heidelberg (2008)

A Memetic Cooperative Optimization Schema and Its Application to the Tool Switching Problem

Jhon Edgar Amaya[1], Carlos Cotta[2], and Antonio J. Fernández Leiva[2]

[1] Universidad Nacional Experimental del Táchira (UNET)
Laboratorio de Computación de Alto Rendimiento (LCAR), San Cristóbal, Venezuela
jedgar@unet.edu.ve

[2] Dept. Lenguajes y Ciencias de la Computación, ETSI Informática,
University of Málaga, Campus de Teatinos, 29071 - Málaga, Spain
{ccottap,afdez}@lcc.uma.es

Abstract. This paper describes a generic (meta-)cooperative optimization schema in which several agents endowed with an optimization technique (whose nature is not initially restricted) cooperate to solve an optimization problem. These agents can use a wide set of optimization techniques, including local search, population-based methods, and hybrids thereof, hence featuring multilevel hybridization. This optimization approach is here deployed on the Tool Switching Problem (ToSP), a hard combinatorial optimization problem in the area of flexible manufacturing. We have conducted an ample experimental analysis involving a comparison of a wide number of algorithms or a large number of instances. This analysis indicates that some meta-cooperative instances perform significantly better than the rest of the algorithms, including a memetic algorithm that was the previous incumbent for this problem.

1 Introduction and Related Work

Collaborative optimization models constitute a very appropriate framework for integrating different search techniques. Each of these techniques has a different view of the search landscape, and by combining the corresponding different exploration patterns, the search can benefit from an increased capability for escaping from local optima. Of course, this capability is more useful whenever the problem tackled poses a challenging optimization task to the individual search algorithms. Otherwise, computational power is diversified in unproductive explorations.

Different schemes have been proposed for cooperating metaheuristics. For example, Toulouse *et al.* [1] considered using multiple instances of tabu search (TS) running in parallel, eventually exchanging some of the attributes stored in tabu memory. Crainic and Gendreau [2] presented a cooperative parallel TS method that was shown to outperform independent search strategies. Crainic *et al.* [3] also proposed a method for asynchronous cooperative multi-search using variable neighborhood search (VNS). Pelta *et al.* [4] presented a cooperative multi-thread search-based optimization strategy, in which several solvers were controlled by a higher-level coordination algorithm which collected information on their search performance and altered the behavior of the solvers accordingly. Milano and Roli [5] developed a multi-agent system called MAGMA (MultiAGent Metaheuristic Architecture) in which metaheuristics are used at different levels (creating solutions, improving them, defining the search strategy, and coordinating lower-level agents). Recently, Amaya et al. [6] have proposed agent topologies equipped with local search (LS) methods based on simple structures of communication similar to those used in the computer networks. More specifically, [6] proposed four different cooperative models (i.e., Ring,Broadcast, Random, and a so-called Ring SDI model) to handle the uniform tool switching problem (ToSP).

In this paper we go a step beyond and have generalized the first model (i.e., Ring) described in [6]; the result is a generic schema whose instances produce meta-cooperative architectures in which one or more agents can also be loaded with a cooperative optimization technique. This schema is not specific for a particular optimization problem and thus can be applied to many combinatorial optimization problems. To demonstrate both the adequacy of the schema and the goodness of its instances (as meta-cooperative algorithms) we have also conducted an empirical evaluation on the ToSP.

2 Ring-Based (Meta-)Cooperative Model

Let us denote $\mathbb{N}_n^+ = \{1, \cdots, n\}$. The optimization architecture proposed is shown in Algorithm 1. As it can be seen, it features an architecture R with n agents connected in form of a ring; each agent a_i ($0 \leqslant i \leqslant n-1$) in R consists of an optimization method (e.g., any metaheuristic, such as a local searcher, a population-based method, or even a hybrid thereof). Observe that there exists a circular list of agents in which each node only sends (resp. receives) information to its successor (resp. from its predecessor). The agents in the architecture engage in periods of isolated exploration followed by synchronous communication. We denote as $cycles_{\max}$ the maximum number of such exploration/communication cycles in a certain cooperative model. Also, let S_i be the set of candidates solutions managed by agent a_i; note that the nature of S_i is variable (e.g., if a_i is a population-based method this means that S_i is its corresponding population whereas if a_i is a loaded with a trajectory-based method, then S_i just contains one candidate).

Firstly all the agents are initialized with a set of initial candidate solutions (lines 1-3, function GenerateCandidateSet). The size of this set depends on the technique endowed in the agent (e.g., it might be a population or just one single candidate). Then, the algorithm is run for a maximum number of iterations cycles (lines 5-16) where, in each cycle, an optimization phase of the specific candidate set kept in each agent is done (lines 6-9) and the best solution obtained in each agent (line 8, function BestCandidateIn) is sent to its successor agent if this solution is better than the best one obtained in the successor agent (lines 10-14). Note that an agent only accepts an incoming solution if it is better than the

Algorithm 1. Ring-based (Meta-)Cooperative Model$_n$

```
for $i \in \mathbb{N}_n^+$ do
    $S_i \leftarrow$ GenerateCandidateSet();
endfor
$cycles \leftarrow 1$;
while $cycles \leqslant cycles_{\max}$ do
    for $i \in \mathbb{N}_n^+$ do
        $S_i \leftarrow a_i(S_i)$;
        $b_i \leftarrow$ BestCandidateIn($S_i$);
    endfor
    for $(i,j) \in \{(i, i(n)+1) \mid i \in \mathbb{N}_n^+$ and $i(n)$ denotes $i$ modulo $n\}$ do
        if fitness($b_i$) < fitness($b_j$) then
            Replace worst candidate in $S_j$ with $b_i$;
        endif
    endfor
    $cycles \leftarrow cycles + 1$
endw
return $\max^{-1}\{$fitness(BestCandidateIn($S_i$)) $\mid i \in \mathbb{N}_n^+\}$;
```

best one kept in its candidate set (lines 11-13) . Observe also that, for a maximum number of evaluations $E_{\max}$ and for a specific number of cycles $cycles_{\max}$, each cycle in our cooperative algorithms spends $E_{cycle} = E_{\max}/cycles_{\max}$ evaluations, and the specific optimization method of any agent takes E_{cycle}/n evaluations at most.

Multiple variants of this cooperative schema can be devised from the general schema shown above as no specific mention is done about the type of the agents involved in the architecture.

3 Experimental Results

To test the feasibility of the proposed architecture we consider the uniform tool switching problem (ToSP) as a benchmark. This section is thus devoted to present the problem and provide a formal description of it. Then, an experimental analysis that includes a wide number of optimization techniques is shown.

3.1 The ToSP

The uniform tool switching problem (ToSP) is a hard combinatorial optimization problem that can be found in flexible manufacturing systems (FMSs) and diverse areas such as electronics manufacturing, metalworking industry, computer memory management, and aeronautics, among others [7,8,9,10]. This problem occurs in a single machine that has several slots into which different tools can be loaded. Each slot just admits one tool, and each job executed on that machine requires a particular set of tools to be completed. Jobs are sequentially executed, and therefore each time a job is to be processed, the corresponding tools must be loaded in the machine magazine. The ToSP consists of finding an appropriate job sequence in which jobs will be executed, and an associated sequence of tool loading/unloading operations that minimizes the number of tool switches in the magazine. Therefore management tool directly affects the efficiency of FMS. The ToSP has been tackled through different methods such as exact methods [8], LS methods, population-based optimization methods [11], and even cooperative models [6]. [12] and [13] proved formally that the ToSP is NP-hard for $C > 2$ and thus exact methods are inherently limited.

Following the previous description of the uniform ToSP, there are two major elements in the problem: a machine M and a collection of jobs $J = \{J_1, \cdots, J_n\}$ to be processed. Regarding the latter, the relevant information for the optimization process is the tool requirements for each job. We assume that there is a set of tools $T = \{\tau_1, \cdots, \tau_m\}$, and that each job J_i requires a certain subset $T^{(J_i)} \subseteq T$ of tools to be processed. As to the machine, we will just consider one piece of information: the capacity C of the magazine (i.e., the number of available slots). Given the previous elements, we can formalize the ToSP as follows: let a ToSP instance be represented by a pair, $I = \langle C, A \rangle$ where C denotes the magazine capacity, and A is a $m \times n$ binary matrix that defines the tool requirements to execute each job, i.e., $A_{ij} = 1$ if, and only if, tool τ_i is required to execute job J_j. We assume that $C < m$; otherwise the problem is trivial. The solution to such an instance is a sequence $\langle J_{i_1}, \cdots, J_{i_n} \rangle$ (where $i_1, \ldots, i_n$ is a permutation of numbers $1, \ldots, n$) determining the order in which the jobs are executed, and a sequence $T_1, \cdots, T_n$ of tool configurations ($T_i \subset T$) determining which tools are loaded in the magazine at a certain time. Note that for this sequence of tool configurations to be feasible, it must hold that $T^{(J_{i_j})} \subseteq T_j$.

We will index jobs (resp. tools) with integers from $\mathbb{N}_n^+$ (resp. $\mathbb{N}_m^+$). An ILP formulation for the ToSP is shown below, using two sets of zero-one decision variables – x_{jk} ($j \in \mathbb{N}_n^+$, $k \in \mathbb{N}_n^+$), and y_{ik} ($i \in \mathbb{N}_m^+$, $k \in \mathbb{N}_n^+$) – that respectively indicate whether a job j is executed at time k or not, or whether a tool i is in the magazine at time k or not. Notice that since each job makes exclusive use of the machine, time-step k can be assimilated to the time at which the kth job is executed. Processing each job requires a particular collection of tools loaded in the magazine. It is assumed that no job requires a number of tools higher than the magazine capacity, i.e., $\sum_{i=1}^{m} A_{ij} \leqslant C$ for all $j \in \mathbb{N}_n^+$. Tool requirements are reflected in Eq. (5). Following [8], we assume the initial condition $y_{i0} = 1$ for all $i \in \mathbb{N}_m^+$. This initial condition amounts to the fact that the initial loading of the magazine is not considered as part of the cost of the solution (in fact, no actual switching is required for this initial load). The objective function $F(\cdot)$ counts the number of switches that have to be done for a particular job sequence:

$$\min \ F(y) = \sum_{k=1}^{n} \sum_{i=1}^{m} y_{ik}(1 - y_{i,k-1}) \tag{1}$$

This general definition shown above corresponds to the *uniform ToSP* in which each tool fits in just one slot. The uniform ToSP considered the cost of switching

$$\forall j \in \mathbb{N}_n^+ : \sum_{k=1}^{n} x_{jk} = 1 \qquad (2)$$

$$\forall k \in \mathbb{N}_n^+ : \sum_{j=1}^{n} x_{jk} = 1 \qquad (3)$$

$$\forall k \in \mathbb{N}_n^+ : \sum_{i=1}^{m} y_{ik} \leqslant C \qquad (4)$$

$$\forall j,k \in \mathbb{N}_n^+ \ \forall i \in \mathbb{N}_m^+ : \ A_{ij}x_{jk} \leqslant y_{ik} \qquad (5)$$

$$\forall j,k \in \mathbb{N}_n^+ \ \forall i \in \mathbb{N}_m^+ : \ x_{jk}, y_{ij} \in \{0,1\} \qquad (6)$$

a tool constant (the same for all tools) and computing the cost of a job sequence by means of a greedy procedure termed *Keep Tool Needed Soonest* (KTNS) [8,9]. The importance of this policy is that given a job sequence KTNS obtains its optimal number of tool switches in polynomial time. Therefore, we can concentrate on determining the sequence of jobs, and use KTNS as a subordinate procedure to decide where (i.e., in which slot) to place each tool.

3.2 Computational Results

As far as we know, no standard data instance exists for this problem (at least publicly available) so that we have selected a wide set of problem instances that were attacked in [8,14,15,16]; more specifically, 16 instances were chosen with values for the number of jobs, number of tools, and machine capacity ranging in [10,50], [9,60] and [4,25] respectively. Table 1 shows the different problem instances chosen for the experimental evaluation where a specific instance with n jobs, m tools and machine capacity C is labeled as $C\zeta_n^m$.

Table 1. Problem Instances considered in the experimental evaluation. The minimum and maximum of tools required for all the jobs is indicated in second and third rows respectively. Fourth row shows the work from which the problem instance was obtained.

	$4\zeta_{10}^{10}$	$4\zeta_{10}^{9}$	$6\zeta_{10}^{15}$	$6\zeta_{15}^{12}$	$6\zeta_{15}^{20}$	$8\zeta_{20}^{15}$	$8\zeta_{20}^{16}$	$10\zeta_{20}^{20}$	$24\zeta_{20}^{30}$	$24\zeta_{20}^{36}$	$30\zeta_{20}^{40}$	$10\zeta_{30}^{25}$	$15\zeta_{30}^{40}$	$15\zeta_{40}^{30}$	$20\zeta_{40}^{60}$	$25\zeta_{50}^{40}$
Min.	2	2	3	3	3	3	3	4	9	9	11	4	6	6	7	9
Max.	4	4	6	6	6	8	8	10	24	24	30	10	15	15	20	20
Source	[14]	[8]		[8]			[8]	[8]	[8]	[8]						
	[15]	[16]	[16]	[16]	[14]	[15]	[16]	[16]	[16]	[16]	[16]	[15]	[14]	[15]	[14]	[15]

Five different datasets[1] (i.e., incident matrixes or relations among tools and jobs) were generated randomly per instance. Each dataset was generated with the restriction, already imposed in previous works such as [14], that no job is *covered* by any other job in the sense that $\forall i,j \in \mathbb{N}_n^+$, $i \neq j$, $T^{(J_i)} \not\subseteq T^{(J_j)}$. The reason to enforce this constraint is to avoid the simplification of the problem by preprocessing techniques as done for instance in [8] and [16].

[1] All datasets are available at *http://www.unet.edu.ve/~jedgar/ToSP/ToSP.htm*

The experiments have been performed using a wide set of different algorithms. In particular we have considered deterministic methods, local search (LS) techniques, cooperative methods, and a number of meta-cooperative algorithms devised from the schema shown in Algorithm 1. From these, a number of variants have also been considered. For instance, as deterministic method we have considered the beam search (BS) algorithm presented in [16]; this algorithms admits a parameter β (termed beam-widht) for which we have considered five different values $\beta \in \mathbb{N}_5^+$. As to LS methods, we have considered two of them: (1) The tabu search versions (TSP and TSF) specialized for the ToSP and described in [6]; here *P and *F is used to indicate the algorithmic variant in which the neighborhood is partially or fully explored respectively (the interested reader is referred to [6] for more details), and (2) the steepest-ascent Hill Climbing (HC) method presented in [11]; from this we have also devised two versions HCP and HCF following the same principles of partial/full exploration mentioned previously.

As cooperative techniques we have considered the memetic algorithm (MA) presented in [11] (denoted as MAHCP because it is a combination of a genetic algorithm (GA) and the method HCP mentioned previously). In [6] we shown that this MA was a killer approach for the ToSP (beating to a number of cooperatives models in which all agents where loaded with LS techniques). We have also included in the comparison a new MA denoted as MATSP because it is a combination of a GA and the TSP method mentioned previously (the parameters were the same as those indicated in [11] for the MAHCP). In these two MAs the LS techniques were always applied to each offspring generated after the mutation step. Other parameters are: $popsize = 30$, $p_X = 1.0$, and $p_M = 1/n$ where n is the number of jobs, with binary tournament selection; alternating position crossover (APX) is used [17], and mutation is done by applying the random block swap as operator (see [11] for more details).

Regarding the meta-cooperative model, we have devised 10 different instances from the schema shown in Algorithm 1 where $n = 3$ and $cycles_{\max} \in \{4, 5\}$; GenerateCandidateSet represents a random initialization, and fitness is defined as the KTNS method described in Section 3.1. In all the instances, at least one agent has been loaded with a cooperative optimization technique, in particular with one of the two MAs mentioned above (i.e., MAHCP or MATSP). In the rest of the paper we have used the notation U(*dd*,*ee*,*ff*,*xx*) to represent an instance of three agents loaded with techniques *dd*, *ee*, *ff* and where *xx* is the number of cycles considered. All algorithms were run 10 times (per instance and dataset) and a maximum of $E_{\max} = \varphi n(m - C)$ evaluations, cf. [6]. Regarding the BS algorithm, because of its deterministic nature, just one execution per dataset (and per value of beam width) was run and the algorithm was allowed to be executed until exhaustion (i.e., until completing the search).

Due to space limitations we will not present all the obtained results for each of the instances and for all the algorithms involved in the comparison, and will use a rank-based approach in order to analyze the significance of the results. To do so, we have computed the rank r_j^i of each algorithm j on each instance i (rank 1 for the best, and rank k for the worst, where $k = 21$ is the number

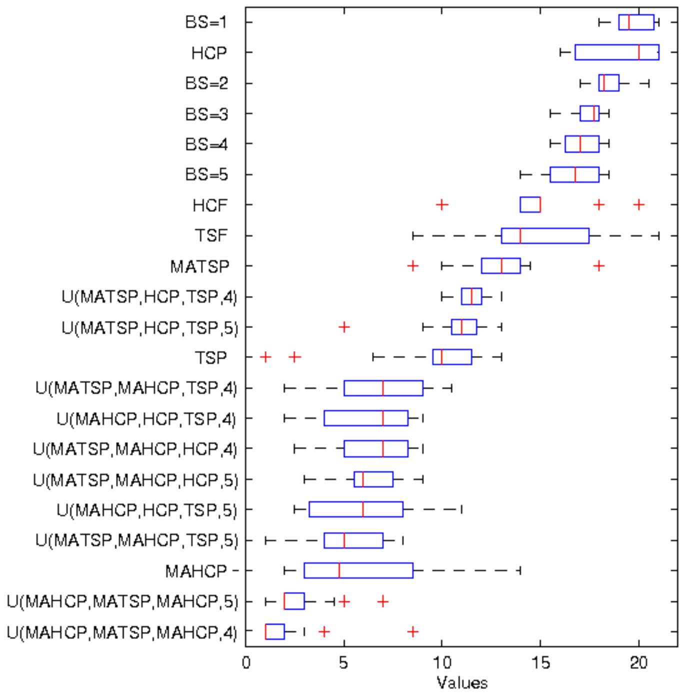


Fig. 1. Rank distribution of each algorithm across all instances. As usual, each box comprises the second and third quartiles of the distribution, the median is marked with a vertical line, whiskers span 1.5 times the inter-quartile range, and outliers are indicated with a plus sign.

of algorithms; in case of ties, an average rank is awarded). The distribution of these ranks is shown in Fig. 1.

Next, we have used two well-known non-parametric statistical tests [18] to compare ranks, namely Friedman test [19] and Iman-Davenport test [20]. The results are shown in Table 2. As seen in the first row, the statistic values obtained are clearly higher than the critical values, and therefore the null hypothesis, namely that all algorithms are equivalent, can be rejected. Since there are algorithms with markedly poor performance, we have repeated the test with the top 4 algorithms (i.e., U(MAHCP,MATSP,MAHCP,4), U(MAHCP,MATSP,MAHCP,5), MAHCP,, and U(MATSP,MAHCP,TSP,5)). Again, it can be seen that the statistical test is passed, thus indicating significant differences in their ranks at the standard $\alpha = 0.05$ level.

Subsequently, we have focused in these top 4 algorithms, and performed Holm's test [21] in order to determine whether there exists significant differences with respect to a control algorithm (in this case U(MAHCP,MATSP,MAHCP,4), the algorithm with the best mean rank). The results are shown in Table 3. The test indicates there exists a significant difference between the control

Table 2. Results of Friedman and Iman-Davenport tests

	Friedman value	critical χ^2 value	Iman-Davenport value	critical F_F value
all	277.30	31.41	97.41	1.61
top 4	41.44	7.81	94.71	2.81

Table 3. Results of Holm's test using U(MAHCP,MATSP,MAHCP,4) as control algorithm

i	algorithm	z-statistic	p-value	$\alpha/(k-i)$
1	U(MAHCP,MATSP,MAHCP,5)	1.369	0.0855	0.017
2	MAHCP	3.834	$6.3e-5$	0.025
3	U(MATSP,MAHCP,TSP,5)	4.108	$1.9e-5$	0.050

Table 4. Computational results. Best results (in terms of the best solution average) are underlined and in boldface. U_1 = U(MAHCP,MATSP,MAHCP,4), U_2 = U(MAHCP,MATSP,MAHCP,5), U_3 = U(MATSP,MAHCP,TSP,5) and MA = MAHCP. $\overline{x}$, σ and b denote the mean, standard deviation and best values respectively.

		$4\zeta^{10}_{10}$	$4\zeta^{9}_{10}$	$6\zeta^{15}_{10}$	$6\zeta^{12}_{15}$	$6\zeta^{20}_{15}$	$8\zeta^{15}_{20}$	$8\zeta^{16}_{20}$	$10\zeta^{20}_{20}$	$24\zeta^{30}_{20}$	$24\zeta^{36}_{20}$	$30\zeta^{40}_{20}$	$10\zeta^{25}_{30}$	$15\zeta^{40}_{30}$	$15\zeta^{30}_{40}$	$20\zeta^{60}_{40}$	$25\zeta^{40}_{50}$
U1	$\overline{x}$	**8.78**	**7.9**	13.82	**15.92**	22.98	**22.66**	26.96	**30.18**	24.42	**44.82**	**40.6**	**64.2**	**99.1**	**95.08**	**205.78**	**143.22**
	σ	1.72	0.81	2.01	1.86	1.92	3.22	2.07	2.16	3.48	8.46	4.34	2.23	12.39	8.09	7.82	11.46
	b	7	7	11	13	20	17	22	26	19	35	32	60	80	82	194	125
U2	$\overline{x}$	8.86	7.98	13.76	16.12	22.84	22.9	**26.78**	30.26	**24.34**	44.92	41.04	64.3	98.64	95.46	206.0	144.72
	σ	1.71	0.79	2.11	1.82	2.04	3.37	1.96	2.38	3.1	8.02	4.66	1.93	12.41	7.62	7.92	11.67
	b	7	7	11	12	20	18	23	25	21	35	31	60	79	83	192	128
MA	$\overline{x}$	8.94	8.1	13.89	16.26	23.18	22.86	27.24	30.53	24.78	44.87	41.3	64.32	99.7	95.86	206.3	144.18
	σ	1.62	0.75	1.99	1.79	1.96	3.41	2.22	2.49	3.29	7.55	4.41	2.4	12.82	7.52	8.81	11.94
	b	7	7	11	12	20	17	22	26	20	35	31	59	80	80	193	122
U3	$\overline{x}$	8.86	7.98	**13.7**	16.28	**22.82**	23.02	27.08	30.48	24.84	45.2	41.52	65.52	100.06	97.1	207.38	145.48
	σ	1.69	0.73	2.06	1.77	2.17	3.72	2.12	2.74	3.13	8.49	4.69	2.86	12.77	7.73	9.89	11.72
	b	7	7	11	13	20	17	22	25	21	33	31	59	81	85	191	127

algorithm and both MAHCP and U(MATSP,MAHCP,TSP,5), but not with respect to U(MAHCP,MATSP,MAHCP,5) (at the 0.05 level; the p-value is only slightly above this value though).

Also, analyzing the obtained results, grouped by problem instances (see Table 4 for the results of these top 4 algorithms), one can observe that the two best meta-cooperative models (i.e., U_1 and U_2) outperform MAHCP (the previous incumbent for this problem) in all the problem instances.

4 Conclusions

In this work we have proposed a memetic cooperative architecture where several agents endowed with MAs and other techniques cooperate in solving a certain optimization problem. This model takes advantage of maintaining a high diversity of possible solutions as well as providing a certain degree of independence in the exploration of different regions of the search space as in island model-based evolutionary systems (although the former is much more flexible since it does not depend solely on population-based algorithms, and tries to exploit the synergy between different search techniques).

The results obtained show the effectiveness of the model on the ToSP, a very hard combinatorial problem related to flexible manufacturing. As expected, the experimentation indicates the choice of heuristic combinations, as well as the number of cycles used in the meta-cooperative model, are crucial parameters. Combinations including several memetic algorithms endowed with both TS and HC have been shown to provide the best results, with statistical significance with respect to other models (including a single MA that was the previous incumbent for this problem). Determining the proper values of some of the parameters (such as the number of agents, number of cycles for communication, the probability of acceptance of solutions, communication topology) in the ToSP and other problems is a line of future work.

Acknowledgements

We thank the reviewers for their valuable comments and suggestions. The second and third authors were partially supported by Spanish MICINN project under contract TIN2008-05941 (NEMESIS project).

References

1. Toulouse, M., Crainic, T.G., Sanso, B., Thulasiraman, K.: Self-organization in co-operative tabu search algorithms. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, October 11-14, vol. 3, pp. 2379–2384 (1998)
2. Crainic, T.G., Gendreau, M.: Cooperative parallel tabu search for capacitated network design. Journal of Heuristics 8(6), 601–627 (2002)
3. Crainic, T.G., Gendreau, M., Hansen, P., Mladenović, N.: Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10(3), 293–314 (2004)
4. Pelta, D., Cruz, C., Sancho-Royo, A., Verdegay, J.: Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. Information Sciences 176, 1849–1868 (2006)
5. Milano, M., Roli, A.: Magma: a multiagent architecture for metaheuristics. IEEE Transactions on Systems, Man, and Cybernetics, Part B 34(2), 925–941 (2004)
6. Amaya, J.E., Cotta, C., Fernández, A.J.: Hybrid cooperation models for the tool switching problem. In: Pelta, D., al., e. (eds.) International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Granada, Spain. Studies in Computational Intelligence. Springer, Heidelberg (in press, 2010)
7. Belady, L.: A study of replacement algorithms for virtual storage computers. IBM Systems Journal 5, 78–101 (1966)
8. Bard, J.F.: A heuristic for minimizing the number of tool switches on a flexible machine. IIE Transactions 20(4), 382–391 (1988)
9. Tang, C., Denardo, E.: Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. Operations Research 36(5), 767–777 (1988)

10. Shirazi, R., Frizelle, G.: Minimizing the number of tool switches on a flexible machine: an empirical study. International Journal of Production Research 39(15), 3547–3560 (2001)
11. Amaya, J., Cotta, C., Fernández, A.: A memetic algorithm for the tool switching problem. In: Blesa, M.J., Blum, C., Cotta, C., Fernández, A.J., Gallardo, J.E., Roli, A., Sampels, M. (eds.) HM 2008. LNCS, vol. 5296, pp. 190–202. Springer, Heidelberg (2008)
12. Oerlemans, A.: Production planning for flexible manufacturing systems. Ph.d. dissertation, University of Limburg, Maastricht, Limburg, Netherlands (October 1992)
13. Crama, Y., Kolen, A., Oerlemans, A., Spieksma, F.: Minimizing the number of tool switches on a flexible machine. International Journal of Flexible Manufacturing Systems 6, 33–54 (1994)
14. Hertz, A., Laporte, G., Mittaz, M., Stecke, K.: Heuristics for minimizing tool switches when scheduling part types on a flexible machine. IIE Transactions 30, 689–694 (1998)
15. Al-Fawzan, M., Al-Sultan, K.: A tabu search based algorithm for minimizing the number of tool switches on a flexible machine. Computers & Industrial Engineering 44(1), 35–47 (2003)
16. Zhou, B.H., Xi, L.F., Cao, Y.S.: A beam-search-based algorithm for the tool switching problem on a flexible machine. The International Journal of Advanced Manufacturing Technology 25(9-10), 876–882 (2005)
17. Larrañaga, P., Kuijpers, C., Murga, R., Inza, I., Dizdarevic, S.: Genetic algorithms for the travelling salesman problem: A review of representations and operators. Articial Intelligence Review 13, 129–170 (1999)
18. Lehmann, E., D'Abrera, H.: Nonparametrics: statistical methods based on ranks. Prentice-Hall, Englewood Cliffs (1998)
19. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of the American Statistical Association 32(200), 675–701 (1937)
20. Iman, R., Davenport, J.: Approximations of the critical region of the Friedman statistic. Communications in Statistics 9, 571–595 (1980)
21. Holm, S.: A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 6, 65–70 (1979)

Ownership and Trade in Spatial Evolutionary Memetic Games

Juan C. Burguillo and Ana Peleteiro

Telematics Engineering Department
University of Vigo, 36310-Vigo, Spain
{J.C.Burguillo,apeleteiro}@det.uvigo.es

Abstract. In this paper we consider several strategies to compete in a spatial version of the Iterated Prisoner's Dilemma (IPD). The cell interaction is modeled by a two-dimensional square lattice, where each cell only locally interacts with its neighbors. Cell actions are public and therefore can be imitated by neighbors. The main contribution of the paper is the framework for the memetic analysis of the population evolution in this extended version of the spatial prisoner's dilemma. Among the classical strategies, cooperate (C) and defect (D), we consider two other strategies based on the property of resources: Possession (P), as the right to possess what one owns, and Trade (T), as the right to buy and sell ownership. This work also includes a set of simulation results showing how ownership and trade emerge from anarchy, as evolutionary stable strategies, to enable the peaceful resolution of property conflicts under certain environment conditions.

Keywords: Evolutionary Game Theory, Iterated Prisoner's Dilemma, Spatial Games, Conflict Resolution.

1 Introduction

Game Theory [2] provides useful mathematical tools to understand the possible strategies that self-interested agents may follow when choosing a course of action. The context of cooperative games and cooperation evolution has been extensively studied seeking general theoretical frameworks like the Prisoner's Dilemma (PD) [1]. An interesting spatial version of the PD was suggested and deeply analyzed by Nowak and other authors [11,10,7] trying to understand the role of local interactions in the maintenance of cooperation.

Evolutionary Game Theory (EGT) [14] models the application of interaction dependent strategies in populations along generations. EGT differs from classical game theory by focusing on the dynamics of strategy change more than in the properties of strategy equilibrium. In evolutionary game theory participants do not posses unfailing Bayesian rationality. Instead, they play with limited computing and memory resources. All the requirement is that the players learn by trial and error, incorporate what they learn in future behavior, and die or somehow 'change' if they do not.

In The Selfish Gene [6], Dawkins proposes that social ideas, what he calls 'memes', are a non-organic replicator form. His examples of memes include tunes, catch-phrases, taboos, and fashions among others. In Dawkins' view, the fundamental characteristics of life are replication and evolution. In biological life, genes serve as the fundamental replicators; while in human culture, memes are the fundamental ones. Both genes and memes evolve by mutation-coated replication and natural selection of the fittest. The memetic theme has found popularity in several fields [12,4,3].

In the memetics model, less successful individuals and groups within a population imitate the behavior of the more successful ones in order to improve their competence for resources. Accordingly, the more above average an individual is, the more others copy his behavior. As a result, the population establishes and self-enforces over time standards of normal behavior. Normal behavior may either be time-independent or it may cycle through a range of behaviors. Memetics belong to evolutionary games because the evolutionary process is essentially a scenario of replication dynamics based on survival of the fittest [9].

This paper develops an evolutionary spatial memetic game model of property ownership and trade, as a way to analyze the conditions for the peaceful resolution of property conflicts. The work is based on [13], but here the framework is an spatial grid distribution where cells play the Iterated Prisoner's Dilemma (IPD) with their neighbors, using a defined 2x2 pay-off matrix.

The remainder of the paper is structured as follows. Section 2 introduces the game model and the basic strategies to be performed by the cells. Section 3 presents the main results obtained in the simulations, and finally; Sect. 4 draws the main conclusions obtained by this work.

2 Spatial Memetic Game Model

The approach we follow in this paper is a composite spatial game where actions are effectively simultaneous, but every cell interacts with several neighbors at a time, depending on the neighborhood radio. Considering the action selected by a cell A, and depending on the action chosen by a neighbor B, cell A receives a pay-off. We consider a spatial structure of the population, i.e., the interaction among cells are locally restricted to their neighbors and obtained by repeating one-to-one games. The background of the work presented here take its roots from [10,11,7] and specially from [13].

2.1 Spatial Distribution

If we let every node in the system to interact with the remaining (N-1) nodes, we have a panmictic population, and this is done with a theoretical analysis at [13]. But, in many real contexts like geography, biology, or Mobile Area Networks (MANETs) [5]; each node interacts mainly with its neighbors. Therefore, for the spatial distribution of the cells we consider a two-dimensional square lattice consisting of N nodes. Figure 1 shows a cell A and two possible neighborhoods

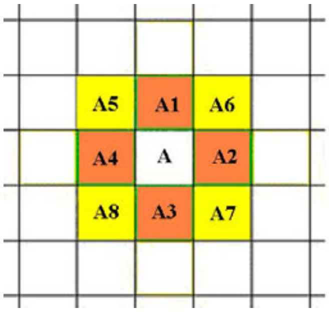


Fig. 1. Cell (A) and two neighborhoods: first with 4 cells (A1,...,A4), and second with 8 cells (A1,...,A8)

depending on the neighborhood radio: the first with a radio 1.0 includes 4 neighbors, while the second has a radio 1.5 and includes 8 neighbors. To determine if a cell is within the neighborhood of cell A, we consider the euclidean distance between the center of both cells.

2.2 Game Basic Strategies

In this section we describe the different actions, and basic strategies available in the game. We start considering the famous Hawk-Dove game [8], whose payoffs are depicted in the left side of Fig. 2. In this game, two equally matched parties compete for a resource, worth by V by each, and we consider only two basic strategies: Hawk and Dove. Since both parties are equal, in a fight between two hawks, each one has only a one-half chance of winning the asset, so in the immediate payoff we consider the average per each, dividing the whole payoff by two. We also include an expected total cost to each participant of h, to model all other costs, p.e., the expected energy expenditure. Doves retreat when confronted by a Hawk. If two Doves meet, a random one of the two Doves retreats and leaves the other to the spoils; so in the immediate payoff matrix we again divide V by two, without any extra cost in this case.

The Prisoner's Dilemma Game: In neoclassical game theory, when ($V/2 > h$) we have the Prisoners Dilemma (PD), while when ($V/2 < h$) we have the Chicken Game (see [2]). The general form of the Prisoner's Dilemma matrix appears at the right side of Fig. 2, and the matrix payoffs must fulfill the following two inequalities:

$$T > R > P > S$$
$$2R > S + T$$

In any one round (or "one-shot") game, choosing defection (D) is a Nash equilibrium, because it rewards the higher payoff for cell A whether the opponent chooses cooperation (C) or defection (D). At the same time, the combined payoff for both cells A and B is maximized if both cooperate. A simple analysis shows

$$\begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\textbf{B}} \\ & & \textbf{Hawk} & \textbf{Dove} \\ \hline \textbf{A} & \textbf{Hawk} & \left(\frac{V}{2}-h, \frac{V}{2}-h\right) & (V,0) \\ \hline & \textbf{Dove} & (0,V) & \left(\frac{V}{2}, \frac{V}{2}\right) \\ \hline \end{array} \qquad \begin{array}{cc|c|c|} & & \multicolumn{2}{c}{\textbf{B}} \\ & & \textbf{C} & \textbf{D} \\ \hline \textbf{A} & \textbf{C} & (\mathrm{R}, \mathrm{R}) & (\mathrm{S}, \mathrm{T}) \\ \hline & \textbf{D} & (\mathrm{T}, \mathrm{S}) & (\mathrm{P}, \mathrm{P}) \\ \hline \end{array}$$

Fig. 2. (left) Hawk-Dove game matrix, and (right) general Prisoner's Dilemma matrix where (R: Reward, S: Sucker's Payoff, T: Temptation and P: Punishment)

that defection is an ESS in a one-shot PD (see [2,9]). This conclusion holds for a so called panmictic population, but here we are interested in the spatial games, so each cell A interacts only with the m cells of its neighborhood. In evolutionary game theory this is called a m-person game, where n = m + 1 in the given case. Each game is played between n players simultaneously, and the payoff for each player depends on the actions taken by the other cells in its neighborhood. Besides, we also consider the iterated version of the PD, i.e., the famous Iterated Prisoner's Dilemma (IPD) [1], and we use the same values for the IPD matrix as in [11], i.e., T = 3.5, R = 3, P = 0.5, S = 0.

Introducing Ownership: *Finders keepers* and *first come, first serve* are not only basic thumb rules in playground citizenship, they are powerful norms that have been recognized by the courts and applied widely in several settings (see [13]). The Possessor strategy models the practice of ownership, and unlike co-operators or defectors, possessors observe convention based on their status; i.e., their behavior depends on whether they are the owner or the intruder of any particular resource.

To model this norm [13] introduces the **Possessor (P)** strategy:

$$P \equiv \begin{cases} D & if\ current\ owner \\ C & if\ current\ intruder \end{cases}$$

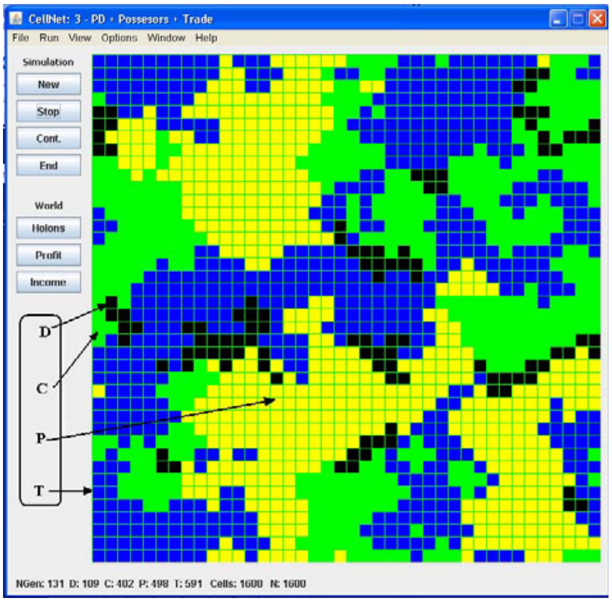
To model ownership in our spatial lattice, we consider that in any encounter between cell A and any of its neighbors B, there is a probability of possession *ProbP*, that makes B to consider A as the owner, i.e., the neighbor always behaves as an intruder.

Introducing Trading: A trader (T) is a possessor who is willing to sell or buy a property when dealing with a fellow trader. In particular, when both owner and intruder of a particular encounter are traders, and the intruder values the property by V, which is more than the owner's value v, then the intruder purchases the property at a value of x, where $v < x < V$. Reference [13] models this norm introducing the **Trader (T)** strategy:

$$T \equiv \begin{cases} if\ neighbor\ is\ not\ T & behave\ as\ P \\ if\ neighbor\ is\ T & \begin{cases} sell\ for\ x & if\ owner\ and\ v < x < V \\ buy\ for\ x & if\ intruder\ and\ v < x < V \\ behave\ as\ P & otherwise \end{cases} \end{cases}$$

2.3 Memetics

The payoff obtained by the neighbors of a cell is public, and every new round every cell imitates the action previously done by the most successful peer in its neighborhood, i.e., the one with the highest payoff. Therefore, the strategy management is memetic, as it imitates the most successful peers.



NGen: 131 D: 109 C: 402 P: 498 T: 591 Cells: 1600 N: 1600

Fig. 3. Snapshot of the CellNet simulator with all the strategies represented by different colors: black (D), green (C), red (P) and blue (T)

3 Games and Results

This section presents some key results from an extensive set of simulations done with the game model and the strategies introduced in the previous sections; attending to the variation of several simulation parameters. All simulations have been done on a PC (Pentium VI dual-core with a memory of 3 GB) and usually taking less than 1 minute per execution (from a hundred to less than a thousand rounds to achieve stability). The simulations has been performed ten times, and the figures in the next subsections correspond to an average result.

In all simulations we consider a square lattice of 40 x 40 = 1600 square cells. When starting, every cell randomly selects its strategy, that may change latter by means of imitation. The IPD game matrix used in all these simulations is the same as in [11] with values: T = 3.5, R = 3, P = 0.5, S = 0. We also consider the effect of mutation to analyze the stability of the different strategies. For this purpose, in some simulations there is a probability that any cell changes its strategy into any other one. Every new round, the income of every cell in the lattice is reset to zero (no accumulation of payoffs). Figure 3 presents a snapshot of our simulator with the coexistence of the different populations (D, C, P, T).

3.1 IPD Game

Figure 4 presents the IPD game with the percentage of cooperators (C) and defectors (D) as a function of the neighborhood radio. As the reader may see, cooperators are more popular with lower radios, but as the radio value reaches 2.5 defectors become the most popular option. The explanation is simple: if there are many neighbors, defectors may attach many more cooperators and avoid their core effect (see [11]).

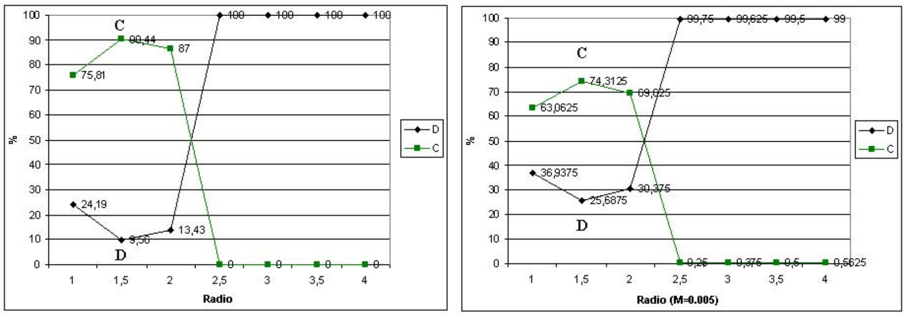


Fig. 4. Evolution of defectors (D) and cooperators (C) as a function of the radio without mutation (left) and with mutation $M = 0.005$ (right)

On the right side of the figure appears the same simulation, but in this case with a mutation probability ($M = 0.005$) per every cell. In this case the results are similar, but as an effect of the mutation, defectors are more popular in lower radio values. The reason is that clusters of cooperators can be altered by a mutation, and that makes cooperation more difficult.

3.2 IPD-Possessor Game

Figure 5 presents the more representative results of the IPD game, including the possessor strategy, for a neighborhood radio of 1. In both figures the x axis presents the probability of being an owner in a confrontation. As before, the picture on the left has no mutation, while the one at the right has it.

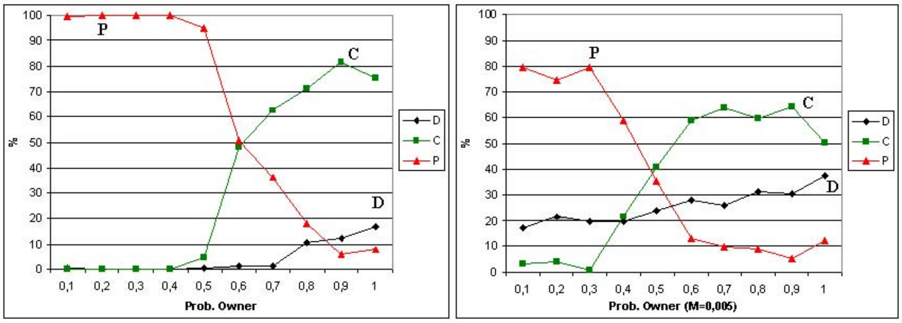


Fig. 5. Evolution of defectors (D) and cooperators (C) and possessors (P) as a function of the ownership probability

On the left figure, we see how the percentages of D, C and P strongly depends on the probability of ownership. Initially P is very popular, and its value decreases as soon as the probability of being a possessor gets closer to 0.5. At that moment, possessors decrease their popularity and cooperators emerge as the preferred option; and this causes that the percentage of defectors also raises as they can attack more cooperators. This seems counterintuitive, but the explanation is that as $ProbP$ gets closer to one, all possessor cells behave as defectors with their neighbors providing 3.5 units to them and 0 to the possessor neighbors in an encounter, this is clearly a lower payoff than when both are cooperators, that receive 3 units each. In fact, the success of P when $ProbP < 0.5$ is that the possessors behaves cooperatively when they are not owners. This is clear when reviewing Fig. 4 as in scenarios with ($Radio < 2.5$) cooperators are more popular.

Table 1. Relation among populations, neighborhood radio and ownership probability

Radio	Neighbors	Results (depending on ProbP)
1	4	P with $ProbP < 0.7$, then C
1.5	8	P with $ProbP < 0.7$, then C
2.0	12	P with $ProbP < 0.7$, then C
2.5	20	Mainly P
3.0	28	Alternation between P and D
3.5	36	Only D

On the right side of Fig. 5 appears the same simulation with a mutation effect $M = 0.005$, and we see how both curves are very similar. Nevertheless, compared to the figure in the left, the percentages of P and C are lower. The explanation is simple: defectors may now appear at any group of cells behaving as P or C, reducing their common success.

Finally, on Table 1 appears the relation among populations, neighborhood radio and ownership probability. We can see the effect described for Fig. 5, and how the defectors take control as soon as the neighborhood is too big (radio over 3.0); which is coherent with the results obtained in Fig. 4.

3.3 IPD-Possessor-Trader Game

Concerning the game with all the strategies, Fig. 6 presents a snapshot with radios 1 and 1.5, and with trading values, v and V, generated uniformly in the interval $[0.5, 3.0]$, which are the payoff values in the IPD matrix for P and R respectively. The graphics in the left of the figure do not use mutation, while the ones at the right do.

Concerning the case without mutation, the interesting effect is the alternation of the different populations depending on the owner's probability ($ProbP$). When this value is low, possessors are more popular, but as we raise the owner's probability the value of P decreases, while traders and cooperators start to become the most popular ones. This is clearly shown in lower left figure where we see that, with a radio of 1.5, there is a clear succession of the different strategies depending on the owner's probability: first possessor, then traders and finally cooperators.

On the right side of the figure, we see the effect of mutation ($M = 0.005$) that is similar than in the previous games, i.e., defectors are more popular in both

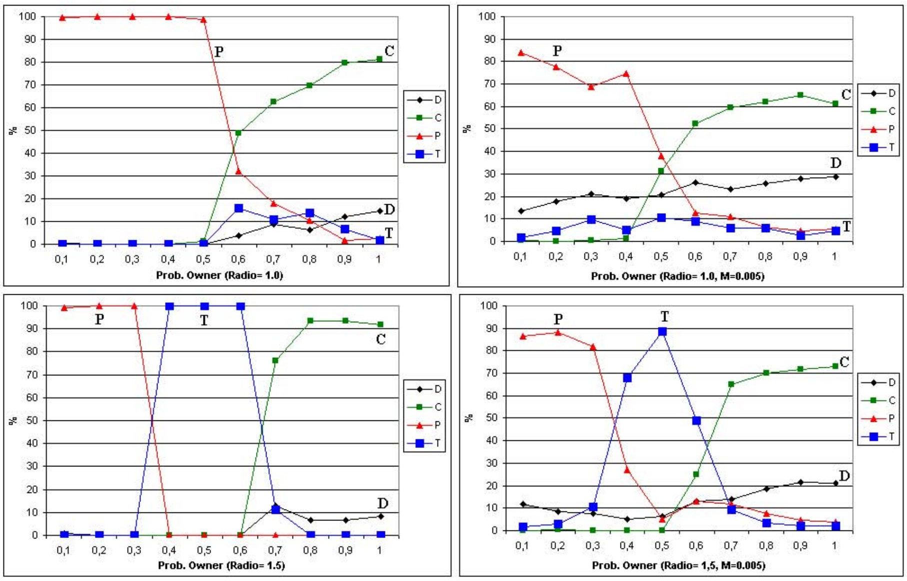


Fig. 6. Evolution of defectors (D), cooperators (C), possessors (P) and traders (T) as a function of the ownership probability for radios 1.0 (top figures) and 1.5 (bottom ones).

Table 2. Relation among populations, neighborhood radio and ownership probability

Radio	Neighbors	Results (depending on ProbP)
1	4	P with $ProbP < 0.5$, then C
1.5	8	P with $ProbP < 0.4$, then T, and C with $ProbP > 0.7$
2.0	12	P with $PropP < 0.3$, then T, and C with $PropP > 0.95$
2.5	20	P and D with $PropP < 0.4$, then T
3.0	28	Mainly D, sometimes T
3.5	36	Only D

figures than in the left ones without mutation. This causes that the rest of the strategies are attenuated, but the whole picture is the same.

Finally, in Table 2 we describe the relation among populations, neighborhood radio and ownership probability. We see how traders are more popular for intermediate values of $ProbP$ and radios between 1.5 and 2.5. This means that when the neighborhood is big enough and not everybody is an owner, trading can emerge. The raising of T can be configured by the trading values v and V, that we set in the interval $[0.5, 3.0]$. Nevertheless, if the neighborhood is too big, then defectors emerge disabling any possibility of cooperation or trading.

4 Conclusions

In this paper we consider several strategies to compete in a spatial version of the iterated prisoner's dilemma (IPD). Among the basic strategies, cooperate (C) and defect (D), we consider two other strategies based on the property of resources: Possession (P), as the right to occupy or possess what one owns; and Trade, as the right to buy and sell ownership. The simulations based on this model describe how evolutionary forces, depending on the simulation parameters, allow the emergence of the different type of populations (D, C, P or T).

On the one hand, deference by intruders to owners is evolutionarily preferred over non-status-based behavior because prior possession enables the resolution of possession conflicts. On the other hand, trade is the ability to buy and sell according to what optimizes personal gain; trading does not occur unless both parties gain. Accordingly, traders always benefit from trade and, when the conditions enable their appearance, they are evolutionarily preferred. But, as seen in the simulations, this preference is clearly dependent, in spatial games, on several parameters as the trading values, the neighborhood and the owner's probability.

The main conclusion is that when the radio is low, defectors can only effect local influences and do not succeed, being possessors the most common population. As soon as the radio raises, traders can appear, but if the radio is too big (bigger than 3.0 in our simulations) then defectors can affect many cells and become a majority. Finally, we also consider the effect of mutations, and we have seen that the results are similar (i.e., the strategies are evolutively stable) but that defectors become more popular as they can infiltrate clusters of cooperators, possessors and traders that would be isolated otherwise.

Future work will analyze the game model described here under different game matrix and trading values, and considering that the cells may learn (i.e., behave as adaptive agents) what is the best action to play against their neighbors.

References

1. Axelrod, R.: The evolution of Cooperation. Basic Books, New York (1984)
2. Binmore, K.: Game Theory. McGraw Hill, New York (1994)
3. Boyd, R., Richerson, P.J.: Not by Genes Alone: How Culture Transformed Human Evolution. Chicago University Press (2005)
4. Brodie, R.: Virus of the Mind: The New Science of the Meme. Integral Press, Seattle (1996)
5. Costa-Montenegro, E., Burguillo-Rial, J.C., Gonzàlez-Castaño, F.J., Vales-Alonso, J.: Agent-Controlled Sharing of Distributed Resources in User Networks. In: Lee, R.S.T., Loia, V. (eds.) Computational Intelligence for Agent-based Systems. Studies in Computational Intelligence, vol. 72, Springer, Heidelberg (2007)
6. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)
7. Langer, P., Nowak, M.A., Hauert, C.: Spatial invasion of cooperation. Journal of Theoretical Biology 250, 634–641 (2008)
8. Maynard Smith, J., Price, G.: The Logic of Animal Conflicts. Nature 246, 15–18 (1973)
9. Maynard Smith, J.: Evolution and the Theory of Games. Cambridge University Press, Cambridge (1982)
10. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
11. Schweitzer, F., Behera, L., Mühlenbein, H.: Evolution of Cooperation in a Spatial Prisoner's Dilemma. Advances in Complex Systems 5(2-3), 269–299 (2002)
12. Waldrop, M.: Complexity: the emerging science at the edge of order and chaos. Simon & Schuster, New York (1992)
13. Yee, K.K.: Ownership and Trade from Evolutionary Games. International Review of Law and Economics 23(2), 183–197 (2003)

A Hyper-Heuristic Approach to Strip Packing Problems

Edmund K. Burke, Qiang Guo, and Graham Kendall

School of Computer Science, University of Nottingham,
Nottingham, NG8 1BB, United Kingdom
{ekb,qxg,gxk}@cs.nott.ac.uk
http://www.asap.cs.nott.ac.uk/

Abstract. In this paper we propose a genetic algorithm based hyper-heuristic for producing good quality solutions to strip packing problems. Instead of using just a single decoding heuristic, we employ a set of heuristics. This enables us to search a larger solution space without loss of efficiency. Empirical studies are presented on two-dimensional orthogonal strip packing problems which demonstrate that the algorithm operates well across a wide range of problem instances.

Keywords: Hyper-heuristic, Strip Packing.

1 Introduction

Cutting and packing problems are a large family of problems arising in many industrial settings, from stock-cutting in the paper, metal, glass and wood industries to container, pallet loading, multi-processor scheduling and other resource allocation problems. Many heuristics have been devised for the problems, and their performance have been intensively studied [1,2]. For the offline version of the problems, where all pieces are known beforehand, results can usually be improved by a meta-heuristic search [3,4,5].

Coffman et al. [1] pointed out that the performance of a single heuristic, in terms of both worst-case and average-case, may vary depending on given instances. Their proofs presume a uniform distribution of item sizes and are applicable to one dimensional problems. The performance for other distributions and higher dimensional instances is not so well understood. The lack of insight into instance properties and heuristic behaviour causes difficulty in practical situations when we need to select an appropriate heuristic for the problem at hand. In addition, many heuristics are designed to guarantee feasible packings, especially in high dimensional cases, but may not be able to construct certain patterns, effectively stopping us being able to find the optimal solution [6]. This limitation is also inherited by any meta-heuristic which employs only one of these heuristics as the mapping function from representation space to solution space.

Hyper-heuristics are currently receiving some attention in the literature [7,8,9]. The approach is motivated by the goal of raising the level of generality of search methodologies [7] and they have been successfully applied to cutting and packing problems [10,11]. In this paper, we propose a novel hyper-heuristic approach which helps intelligently choose a suitable heuristic each time we need to place an item. The main differences from previous, standard meta-heuristic approaches is that a set of heuristics will be utilised. The heuristic set will map the representation space to the solution space so that the algorithm can avoid the shortcomings of only using one heuristic. To demonstrate the effectiveness of the approach, we propose a hyper-heuristic based on a genetic algorithm. The chromosome contains not only which item to pack, but also which heuristic(s) are available to pack that item. Therefore, we enhance the standard genetic algorithm (GA) encoding, where a chromosome is a permutation of items, by adding a set of heuristics together with probabilistic information. Such information will facilitate choice decisions to select heuristics rather than rely on a user's arbitrary judgement. Compared to the hyper-heuristic by Ross [12,10], the learning mechanism updates the probabilities of applying heuristics according to their historical performance, rather than through a learning classifier system.

2 Related Work

In classical two-dimensional orthogonal strip packing problems [6], we are given a container **C**, with width **W** and infinite height. We are required to pack into **C** a set of small rectangles $\mathbf{R} = \{r_1, r_2, ..., r_n\}$ with (w_i, h_i) denoting the width and height for each $r_i \in \mathbf{R}$. The objective is to minimise the total height of the packed rectangles. Typical assumptions, as summarised by Fekete and Schepers [13], are:

1. Each edge of the rectangles have to be parallel to one edge of the container (orthogonal);
2. We do not require guillotine cutting (free-form);
3. All rectangles must be within the container (closeness);
4. Rectangles must not overlap with each other (disjoint);
5. Rectangles cannot be rotated (fixed orientation).

The problem can be classified as two-dimensional regular open dimensional packing (2D-R-ODP) according to the typology proposed by Wäscher et al. [14].

Some heuristics for one-dimensional cases can be modified for the strip packing problem. Baker et al. [6] presented a bottom-up left-justified (BL) heuristic, which finds the lowest feasible space, similar to one dimensional First Fit (FF), and packs left justified. Another heuristic has been described by Liu and Teng [15]. Each rectangle is dropped from the top right corner of the container and moved down and then left until it settles at a stable position. The heuristic overlooks any holes formed by preceding rectangles in the partial packing. Therefore, it can be regarded as Next Fit (NF) [1] which never utilises empty spaces produced at an earlier stage. The Best Fit (BF) policy [1] fits a piece into the smallest feasible space. Hayek et al. [16] proposed a way to search for the smallest feasible space. Burke et al. [17] designed a Best Fit Decreasing Width

(BFDW) heuristic, which was later enhanced using a meta-heuristic as a post process operation [20].

Meta-heuristic approaches have also attracted much attention as they have been shown to produce good quality solutions for cutting and packing problems [18,19,20,21]. The most common way of implementing these methodologies for packing problems is to use a hybrid strategy which combines an iterative search component together with a placement heuristic. For example in a typical GA method, the GA searches for the best permutation of shapes that enables a decoder to return a good packing solution. Given that the different decoders and parameter sets perform differently, it normally relies on a user's decision (or even intuition) to make appropriate choices.

Being aware of the difficulties faced by heuristics and meta-heuristics, a natural question to ask is if we can develop an automated system which requires less human interaction and can deal with a wide range of problems? Ross et al. [12,10] presented two approaches for one-dimensional bin packing problems. They associate a set of packing heuristics with different packing statuses. A learning classifier system [12] and a genetic algorithm [10], acting as higher level managers, search for an appropriate packing heuristic to be employed at each step of the packing. In these approaches, a set of predefined problem statuses is used to describe the status of partially filled bins and the remaining pieces. In [12], a classifier system determines the problem status and decides which low level heuristic to call. This approach requires a good understanding between problems and low level heuristics. For many situations, such as in high dimensional cases, the task of gaining such understanding, and enumerating all possible situations, can be non-trivial. In [10], a GA is employed to detect the problem status and suitable heuristics to employ. As we demonstrate in the next section, some heuristics such as left-justify or right-justify, may not be relevant to the problem status, but are still crucial in some cases.

3 The GA-Based Hyper-Heuristic Approach

3.1 Overview

Our hyper-heuristic approach is based on a genetic algorithm for the over-riding search strategy (Fig. 1). To facilitate the choice of heuristics, the standard chromosomes are enhanced by combining the sequence of rectangles with heuristic-probability pairs. Section 3.2 provides more details on the chromosomes. Compared to the hyper-heuristic approach using a static learning classifier system [12,10], our approach adopts a roulette-wheel selection mechanism to choose a heuristic from the candidate set (step 2.3 in Fig. 1), along with an adaptive learning mechanism to intelligently recognise a suitable heuristic within a set (steps 2.5 to 2.7 in Fig. 1).

```
1.1 for each individual $c_{ind}$ random shuffle $\{r_1, r_2, \ldots, r_n\}$;
1.2 for each rectangle $r_i$ initialize a set of heuristic-probability pairs $(h_i^j, p_i^j)$;

2 while $Iteration \leq Iteration_{max}$
2.1 select parents $c_x, c_y$;
2.2 generate new child $c_{ind'}$ by crossover and mutation;

2.3 choose a heuristic $h_i^{j^*}$ according to its probability $p_i^{j^*}$;
2.4 pack $r_i$ with $h_i^{j^*}$;

2.5 $\Delta = (Height_{c_{ind}} - Height_{c_{ind'}})/Height_{c_{ind}}$;
2.6 $p_i^{j^*} = max\{0, p_i^{j^*} + \Delta\}$;
2.7 for each $j \in \mathbf{J} \setminus j^*$, update $p_i^j$;
```

Fig. 1. Pseudo-code of the GA-based hyper-heuristic framework

We also compare two alternative versions of the hyper-heuristic to decide the types of heuristic decoders:

Non-competing heuristic sets (NC-HH). The type of heuristics are fixed. They are all available to pack each shape even if the probability value approaches zero (see step 2.7 in Fig. 2). In this version, although the heuristics h_i^j are arbitrarily chosen and remain static, the probabilities p_i^j are updated adaptively and the search procedure is still a dynamic probability selection mechanism. Fig. 2 shows details of the refined procedures of steps 1.2 and 2.7 for this version of the hyper-heuristic.

```
1.2 for each $r_i$ initialize a set of $|\mathbf{J}|$ heuristics, and set each $p_i^j = \frac{1}{|\mathbf{J}|}$;
2.7 for each $j \in \mathbf{J} \setminus j^*$, $p_i^j = max\{0, p_i^j - \frac{\Delta}{|\mathbf{J}|-1}\}$;
```

Fig. 2. Refined step 1.2 and 2.7 for NC-HH

Competing heuristic sets (C-HH). The hyper-heuristic chooses initial heuristic sets, and it allows badly performing heuristics to be replaced (Fig. 3). When initialising, the hyper-heuristic randomly selects a subset of heuristics from all those available. During the updating process, if the probability of a heuristic drops below a threshold level, it will be replaced by another randomly chosen heuristic. Whenever replacement happens, the probabilities of the heuristics will be reset to allow the newly introduced heuristic a fair chance of competing with the surviving members that are already in the set. In effect, all heuristics are competing against each other in order to stay in the candidate set.

```
1.2 for each $r_i$ random select a set of $|\mathbf{J'}| < |\mathbf{J}|$ heuristics, and set each $p_i^j = \frac{1}{|\mathbf{J'}|}$;
2.7 if the incumbent heuristic $h_i^{j^*}$ has $p_i^{j^*} < Prob_{th}$ replace $h_i^{j^*}$ with a new heuristic, set $p_i^j = \frac{1}{|\mathbf{J'}|}$, reset other heuristics' probabilities;
otherwise for each $j \in \mathbf{J'} \setminus j^*$, $p_i^j = max\{0, p_i^j - \frac{\Delta}{|\mathbf{J'}|-1}\}$;
```

Fig. 3. Refined step 1.2 and 2.7 for C-HH

3.2 Chromosomes

We enhance the standard genetic algorithms' chromosome by including with each item (allele) some probabilistic information for heuristic selection. Each allele is denoted as a set of pairs of heuristic h_i^j and probability p_i^j, $i = 1, 2, ..., n$,where n is the number of items and j is a parameter defining the number of candidate decoding heuristics available to each rectangle. Fig. 4 shows a chromosome for the proposed hyper-heuristic methodology.

r_1	r_2	$\dots$	r_n
(h_1^1, p_1^1) (h_1^2, p_1^2)	(h_2^1, p_2^1) (h_2^2, p_2^2)	$\dots$	(h_n^1, p_n^1) (h_n^2, p_n^2)
$\dots$	$\dots$	$\dots$	$\dots$

Fig. 4. A hyper-heuristic GA chromosome

The values of the probabilities (initially set equal) will be updated through a learning mechanism. The choice of a heuristic for each piece will be rewarded or punished according to the results of the final packing height, i.e. the probabilities of incumbent heuristics will be increased if we obtain a better packing, and decreased otherwise. Therefore, the system learns from its interaction with the search problem. For example, a system may find it tends to apply rules finding lower positions for large pieces, while for small pieces there is less difference in heuristic probabilities. The hyper-heuristic uses this adaptive policy to learn how to utilise the heuristics.

3.3 Decoding Heuristics

Decoding heuristics for higher dimensional problems are concerned with two decisions: which space to select for the placement and where in the chosen space to place the item. For the first decision, we will use three categories of heuristics: First Fit, Next Fit and Best Fit. To implement these heuristics, we maintain a list of feasible spaces, initially containing one element of the size of the strip. We recalculate the list after placing each shape, similar to [16].

First Fit (FF) select the feasible space at the lowest level, break ties by choosing the left most space (equivalent to the bottom-up heuristic [6]);

Best Fit (BF) select the feasible space with the smallest area;

Next Fit (NF) spaces not exposed from the top of the partial packing will be removed from the list, then select the lowest feasible space (equivalent to bottom-left move with downward priority [15]).

For the second decision our hyper-heuristic will consider all four corners of a chosen space. Therefore, in our experiments, we have twelve different placement options for each item. The type and quantity of heuristics will affect the performance of the hyper-heuristic, possibly due to the larger the size of the pool, the potentially larger search space (see section 4.3). Therefore, we limit the candidate sets to a more manageable size of four rather than using all twelve heuristics.

3.4 Selection and Replacement Strategy

Parent selection is carried out by truncated selection. By experimentation we found that it is more effective to select parents from the top third, rather than using roulette-wheel selection from the entire population. A child chromosome replaces the worst member in the population, that is not a replica of an existing chromosome.

3.5 Recombination

The GA recombination operators are standard, involving a random two-point order-based crossover (2OX) [5] and mutation. In particular, when exchanging orders of items in a sequence the associated set of heuristics of each item will be exchanged as well. We have implemented two other operators, partial matching crossover (PMX) and single point crossover (1OX), which also guarantee feasibility. Compared with 2OX, PMX makes little difference and 1OX performs slightly worse. The detailed settings of parameters will be shown in Section 4.

4 Experimental Results

To examine the effectiveness of the proposed hyper-heuristics, we have created a set of instances to demonstrate that hyper-heuristics can explore a wider solution space (Section 4.1). We also compare the average performance with standard GAs (Section 4.2). It is also interesting to investigate the impact of the size of the heuristic sets, which is an important parameter affecting the size of the search space (Section 4.3).

The benchmark instances are taken from Burke et al. [17] and the OR-library `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/`. C1 to C7 are seven categories with three instances in each and N1a to N7e are 35 non-guillotine instances. N1 to N12 have a number of items ranging from 10 to 500, and the other two sets of instances have 16 to 197 items.

The algorithm was implemented in C++ and ran on a grid computer with 2.2GHz CPUs, 2GB memory and GCC compiler. To obtain statistics every experiment was run 100 times.

4.1 Feasibility and Optimality

The first set of experiments is designed to evaluate the effects of multiple decoders. We created some instances where gaps have to exist in the middle of patterns in the optimal solutions (as per Baker et al. [6]). Using only one heuristic will fail to achieve the optimal pattern. An example of such an instance is as follows. Nine Items: 60x60, 60x60, 50x50, 50x50, 40x40, 40x40, 10x10, 10x10, 31x30 are to be packed into a strip of width of 151. (Note if the last item was 30x30 and the strip has a width of 150, the shapes would fit perfectly.) The best results achieved by a meta-heuristic with a single placement heuristic (in our experiments GA+NFBL, GA+FFBL, GA+BFBL) and hyper-heuristics (both C-HH and NC-HH versions) are 120 and 110 respectively Fig. 5). It is simple to verify that 110 is the optimal. Assuming the optimal is less than 110, say 109, the whole area of strip needed (including any utilised and wasted areas) is 16,459 (151x109), which is less than the total area of all items 16,530, therefore it is impossible.

Other instances in our dataset are created by choosing a number of pieces and cutting at random points. The hyper-heuristics demonstrates stronger

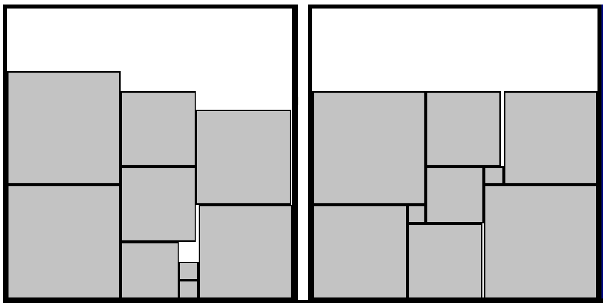


Fig. 5. Best result achieved by meta-heuristic is 120 and optimal achieved by hyper-heuristic is 110

Table 1. Average and best results of new instances

	instance 1		instance 2		instance 3		instance 4		instance 5		instance 6		instance 7		instance 8	
	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg	min	avg
Next Fit HH	**110**	**112.3**	**110**	**119.3**	110	**113.04**	**110**	120.8	111	**115.26**	120	121.0	112	**114.30**	**116**	**119.67**
GA	120	120.0	120	120.0	110	113.70	120	**120.7**	111	115.45	120	**120.5**	112	115.06	117	120.37
First Fit HH	**110**	**110.1**	**110**	**118.1**	110	**111.85**	**110**	**120.0**	111	**113.10**	120	**120.0**	**110**	**113.26**	116	**117.62**
GA	120	120.0	120	120.0	110	112.43	120	120.1	111	114.31	120	120.2	111	114.20	116	118.40
Best Fit HH	**110**	**110.0**	**110**	**116.3**	110	**111.77**	110	**119.5**	111	**112.43**	120	120.2	**110**	**112.61**	116	**118.50**
GA	120	120.0	120	120.0	110	111.91	110	119.6	111	113.75	**110**	**120.1**	111	113.42	**114**	118.82

performance (Table 1). Comparing best and average results to algorithms applying only one heuristic, hyper-heuristics are better on almost all cases. This experiment provides evidence that hyper-heuristics can avoid the drawbacks of applying only a single heuristic, and find more feasible solutions and, possibly, optimal solutions.

4.2 Performance

In this experiment we further compare our hyper-heuristics to standard meta-heuristics on well known benchmark instances for each category of decoders (FF, NF and BF). The hyper-heuristic (NC-HH) utilises four positioning heuristics while the standard GA uses only one. Table 2 shows that the hyper-heuristics produces superior solutions in more cases on the First Fit and Best Fit and equal solutions on Next Fit. The extra calculations to update the probabilities only causes a minor increase to the CPU time even for larger sized instances, as shown in Table 3 (0.5% on average).

Table 2. Average of all instances

dataset	number of	First Fit			Next Fit			Best Fit		
	instances	HH Wins	GA Wins	equal	HH Wins	GA Wins	equal	HH Wins	GA Wins	equal
n1-n12	12	5	5	2	2	9	1	9	2	1
c1-c7	21	12	7	2	8	12	1	11	10	0
n1a-n7e	35	16	19	0	21	14	0	18	17	0
new	8	8	0	0	6	2	0	7	1	0
total	76	41	31	4	37	37	2	45	30	1

Table 3. Average CPU time for 5000 evaluations (milliseconds)

Set size	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12
GA	174	508	924	1546	2224	2216	2820	4076	4786	9902	18358	62022
HH	176	518	906	1568	2230	2208	2824	4074	4856	10008	18542	61950

4.3 Effects of Number of Heuristics in a Set

In the next set of experiments we attempt to find a suitable trade-off between the size of the set (and thus computational time) and solution quality. In Table 4, we present a comparison between four runs of a hyper-heuristic (C-HH version) where heuristics are all randomly chosen and the set size varies between 4 and 8. It can be seen that many of the best results (highlighted) are produced with just four heuristics.

Table 4. Heuristic set size affects results

Set size	n1	n2	n3	n4	n5	n6	n7	n8	n9	n10	n11	n12
size of 4	40.00	51.51	**52.59**	**84.08**	**106.19**	104.78	**110.04**	**85.79**	**156.48**	**154.36**	155.88	317.26
size of 5	40.00	51.32	52.61	84.29	106.42	104.63	110.37	86.26	156.61	154.66	155.97	317.28
size of 6	40.00	**51.08**	52.65	84.39	106.50	104.70	110.20	86.02	156.70	154.80	155.80	317.20
size of 7	40.00	51.22	52.66	84.40	106.42	**104.55**	110.26	86.35	156.63	154.64	**155.75**	**317.13**

5 Conclusion and Future Work

In this paper we have proposed a hyper-heuristic approach to tackle cutting and packing problems. The idea is to combine a set of heuristic decoders with a high level search operator. Empirical studies have demonstrated that the hyper-heuristic approach is superior to standard meta-heuristics which use only one decoder. The potential benefits can be summarised as follows:

– Compared to standard approaches the hyper-heuristic is able to explore a larger solution space. Therefore, it has the potential to find the global optima or deliver better results than other meta-heuristic approaches.
– Its built-in learning mechanism is highly automated requiring less user judgement, as the hyper-heuristic itself will intelligently choose a suitable heuristic to pack a given item. It is also flexible for further expansion by having the option to add new heuristics into the candidate set.

In this paper, the hyper-heuristic utilises a GA as the search methodology and a number of well known heuristics as decoders. The hyper-heuristic is flexible to adopt other search engines, such as Tabu Search or Simulated Annealing. It is also possible to employ other more sophisticated low-level heuristics, such as those considering shared edges. There is scope for further improvement by integrating techniques such as a local search into the hyper-heuristic framework. Like most meta-heuristics, hyper-heuristics are usually computational intensive algorithms. Therefore, it is interesting to investigate if parallelization (e.g. an island model) could solve even lager instances. Further work is also required to understand the dynamics among different level operators and the evolution and interaction between the heuristic search space and solution space.

Acknowledgments. We would like to acknowledge the support of EPSRC (Engineering and Physical Sciences Research Council) for supporting this work (Ref: EP/D061571/1).

References

1. Coffman, E., Garey, M., Johnson, D.: Approximation algorithms for bin packing: a survey. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-hard Problems, pp. 46–93. PWS Publishing, Boston (1996)
2. Lodi, A., Martello, S., Monaci, M.: Two-dimensional packing problems: a survey. Eur. J. Oper. Res. 141, 241–252 (2002)
3. Aarts, E., Korst, J., Michiels, W.: Simulated annealing. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 7, pp. 187–210. Springer, Heidelberg (2005)
4. Gendreau, M., Potvin, J.: Tabu search. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 6, pp. 165–186. Springer, Heidelberg (2005)

5. Sastry, K., Goldberg, D., Kendall, G.: Genetic algorithms. In: Burke, E., Kendall, G. (eds.) Search Methodologies - Introductory Tutorials in Optimization, Search and Decision Support Methodologies, ch. 4, pp. 97–126. Springer, Heidelberg (2005)
6. Baker, B., Coffman, E., Rivest, R.: Orthogonal packings in 2 dimensions. SIAM Journal on Computing 9, 846–855 (1980)
7. Burke, E., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics 9, 451–470 (2003)
8. Burke, E., Hart, E., Kendall, G., Newall, P., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern research technology. In: Handbook of Metaheuristics, ch. 16, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)
9. Ross, P.: Hyper-heuristics. In: Burke, E., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, ch. 17, pp. 529–556. Springer Science, Heidelberg (2005)
10. Ross, P., Marín-Blázquez, J.G., Schulenburg, S., Hart, E.: Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. 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, Part II. LNCS, vol. 2724, pp. 1295–1306. Springer, Heidelberg (2003)
11. Dowsland, K., Gilbert, M., Kendall, G.: A local search approach to a circle cutting problem arising in the motor cycle industry. J. Oper. Res. Soc. 58, 429–438 (2007)
12. Ross, P., Schulenburg, S., Marín-Blázquez, J.G., Hart, E.: Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of Genetic and Evolutionary Computation - GECCO 2002, vol. 6, pp. 942–948 (2002)
13. Fekete, S., Schepers, J.: A combinatorial characterization of higher-dimensional orthogonal packing. Mathematics of Operations Research 29, 353–368 (2004)
14. Wäscher, G., Hauáner, H., Schumann, H.: An improved typology of cutting and packing problems. Eur. J. Oper. Res. 183, 1109–1130 (2007)
15. Liu, D., Teng, H.: An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles. Eur. J. Oper. Res. 112, 413–420 (1999)
16. El Hayek, J., Moukrim, A., Negre, S.: New resolution algorithm and pretreatments for the two-dimensional bin-packing problem. Computers & Operations Research 35, 3184–3201 (2008)
17. Burke, E., Kendall, G., Whitwell, G.: A new placement heuristic for the orthogonal stock-cutting problem. Operations Research 52, 655–671 (2004)
18. Hopper, E., Turton, B.: A review of the application of meta-heuristic algorithms to 2D strip packing problems. Artificial Intelligence Review 16, 257–300 (2001)
19. Alvarez-Valdes, R., Parreño, F., Tamarit, J.: Reactive grasp for the strip-packing problem. Computers & Operations Research 35, 1065–1083 (2008)
20. Burke, E., Kendall, G., Whitwell, G.: A Simulated Annealing Enhancement of the Best-Fit Heuristic for the Orthogonal Stock Cutting Problem. INFORMS Journal on Computing 21(3), 505–516 (2009)
21. Dowsland, K., Herbert, E., Kendall, G., Burke, E.: Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems. Eur. J. Oper. Res. 168, 390–402 (2006)

Asymptotic Analysis of Computational Multi-Agent Systems

Aleksander Byrski[1], Robert Schaefer[1,⋆], Maciej Smołka[2], and Carlos Cotta[3,⋆⋆]

[1] AGH University of Science and Technology, Kraków, Poland
[2] Jagiellonian University, Kraków, Poland
[3] University of Málaga, Málaga, Spain
{olekb,schaefer}@agh.edu.pl, smolka@ii.uj.edu.pl, ccottap@lcc.uma.es

Abstract. A stationary Markov chain model of the agent-based computation system EMAS is presented. The primary goal of the model is better understanding the behavior of this class of systems as well as their constraints. The ergodicity of this chain can be verified for the particular case of EMAS, thus implying an asymptotic guarantee of success (the ability of finding all solutions of the global optimization problem). The presented model may be further adapted to numerous evolutionary and memetic systems.

1 Motivation

Evolutionary algorithms (EAs) and multi-agent systems (MAS) are closely related paradigms. Among other similarities, they share conceptual elements such as the usage of a pool of entities (individuals in the case of EAs, agents in the case of MAS) which interact among themselves directly (via n-ary operators in EAs, and using autonomous, proactive or reactive behaviors in MAS) or indirectly (via modifications of the environment in MAS, and by e.g. coevolution, archive-based strategies, etc. in EAs). Not surprisingly, cross-fertilization of both paradigms has been attempted in the so-called agent-based computational systems (e.g. EMAS [3], AMAS [22], GCE [6]). In particular, EMAS (Evolutionary Multi-Agent System introduced by Cetnarowicz et al. in [5]) have been shown to be effective in solving difficult optimization tasks, e.g., optimization of neural-network architectures, multi-modal optimization, multi-criteria optimization.

The connection of MAS and EAs is particularly clear in the case of memetic algorithms (MA). It is customary – and in some sense based on good practical reasons – to consider that a MA is an EA hybridized with some form of local search (LS). This definition of MA was actually popularized by early works such as [17], and paved the way for the vigorous development of optimization algorithms based on this idea (exhibiting a remarkable record of success, check

⋆ R. Schaefer is partially supported by the Polish Ministry of Science and Higher Education under project no. 4057/B/T02/2009/37.

⋆⋆ C. Cotta is partially supported by MICINN under project NEMESIS (TIN2008-05941).

e.g. [7]). It is also true that seminal works on this topic had a wider perspective, in which EAs endowed with LS was rather an appropriate incarnation of a MA than a restrictive definition [13]. Under this wider interpretation of the MA paradigm, a stronger relationship with multi-agent systems emerges. Indeed, a MA has been sometimes defined (as early as in [16] – see also [14]) as a cooperative-competitive strategy of optimizing agents. The use of the term *agent* here tries to emphasize the fact that individuals are more than mere solution placeholders that passively suffer the application of different variation and selection operations on them [15]. On the contrary they can be regarded as active actors in the search process, intertwining periods of individual search/learning with periods of cooperation and competition. While this interpretation remains compatible with classical MA approaches, it also opens up the door to more complex strategies, e.g. individual roles [1], different recombination behaviors [2], etc.

The relationship between multi-agent systems and memetic algorithms is not limited to a simple source of algorithmic inspiration. On the contrary, it can provide a useful means for improving the theoretical understanding of these techniques. In this sense, and opposed to classical evolutionary strategies for which qualitative formal models were introduced and intensively studied (see e.g. [23], [20], [19]), it must be noted that there is still lack of them for most complex, biologically-inspired heuristics.

Based on the results presented in [4,21] we introduced a discrete, finite state-space Markov chain as a model for EMAS. We also proved ergodicity of such model. This feature in not so obvious in this case as in case of simple genetic mechanisms, where the passage between two arbitrary states is possible in a single step, if the mutation rate is strictly positive [23].

Such an analysis may ensure that the system is able to reach a population containing an arbitrary minimizer in a finite number of steps. Moreover the effective upper bound of step number may be evaluated. In addition, asymptotic guarantee of success is satisfied [8,18]. In the course of modelling a number of constraints were indicated, leading to better understanding of the functioning of these systems (e.g. synchronization schemes, probability distributions used and topology of connections).

2 EMAS Architecture and Behavior

We will focus on a EMAS systems solving global optimization problems consisting of finding all global minimizers $\arg\min\{FITN(x)\}, x \in U$ where $FITN : U \to \mathbb{R}_+$, and U is a finite genetic universum $\#U = r < +\infty$.

Computational EMAS agents belong to the predefined finite set Ag one-to-one mapped on set $U \times P$, where $P = \{1, \ldots, p\}$ and p is assumed to be the maximum number of agents contain the same genotype, so each agent $ag_{gen,n} \in Ag$ is uniquely represented by its signature $(gen, n) \in U \times P$.

Agents are assigned to locations $Loc = \{1, \ldots, s\}$. The locations are linked by channels along which agents may migrate from one location to another. The topology of channels is determined by the symmetric relation $Top \subset Loc^2$. We assume that the connection graph $\langle Loc, Top \rangle$ is coherent and does not change during the system evolution. Each agent possesses a variable parameter called energy, its value is quantized and belongs to $\{0, \Delta e, 2 \cdot \Delta e, 3 \cdot \Delta e, \ldots, m \cdot \Delta e\}$. The current value of the energy exhibits the maturity of agent in solving the optimization problem, affecting its abilities (reproduction, cloning, migration) (see [9]).

Let us introduce the set of three-dimensional, incidence and energy matrices $x \in X$ with s layers (corresponding to all locations) $x(i) = \{x(i, gen, n),\ gen \in U,\ n \in P\}$, $i \in Loc$. The layer $x(i)$ will contain energies of agents in i-th location. In other words, $x(i, gen, k) > 0$ means that the k-th clone of the agent containing the gene $gen \in U$ is active, its energy equals $x(i, gen, k)$ and it is located in i-th location.

We introduce the following coherency conditions:

– each layer $x(i)$ contains at most q_i values greater than zero, which denotes the maximum capacity of the i-th location, moreover, the quantum of energy Δe is lower or equal than total energy divided by the maximal number of individuals that may be present in the system $\Delta e \leq \frac{1}{\sum_{i=1}^{s} q_i}$ what allows to achieve maximal population of agents in the system,
– reasonable values of p should be greater or equal to 1 and less or equal to $\sum_{i=1}^{s} q_i$. We assume that $p = \sum_{i=1}^{s} q_i$ which assures that each configuration of agents in locations is available, respecting the constrained total number of active agents $\sum_{i=1}^{s} q_i$. Increasing p over this value does not enhance the descriptive power of the presented model,
– $(\cdot, j, k)$-th column contains at most one value greater than zero, which expresses that the agent with k-th copy of j-th genotype may be present in only one location at a time, whereas other agents containing copies of j-th genotype may be present in other locations,
– entries in the incidence and energy matrices are non-negative $x(i, j, k) \geq 0,\ \forall\ i = 1, \ldots, s,\ j = 1, \ldots, r,\ k = 1, \ldots, p$ and $\sum_{i=1}^{s} \sum_{j=1}^{r} \sum_{k=1}^{p} x(i, j, k) = 1$, which means that the total energy contained in the whole system is constant, equal to 1.

Gathering all these conditions, the set of three-dimensional incidence and energy matrices may be described in the following way:

$$X = \Big\{ x \in \{0, \Delta e, 2 \cdot \Delta e, 3 \cdot \Delta e, \ldots, m \cdot \Delta e\}^{s \cdot r \cdot p},\ \Delta e \cdot m = 1,$$
$$\sum_{i=1}^{s} \sum_{j=1}^{r} \sum_{k=1}^{p} x(i, j, k) = 1 \text{ and } \forall\ i = 1, \ldots, s\ \sum_{j=1}^{r} \sum_{k=1}^{p} [x(i, j, k) > 0] \leq q_i$$
$$\text{and } \forall j = 1, \ldots, r,\ k = 1, \ldots, p\ \sum_{i=1}^{s} [x(i, j, k) > 0] \leq 1 \Big\} \tag{1}$$

where $[\cdot]$ denotes the value of the logical expression contained in the parentheses.

Note that the formula (1) implies that there must exist at least one agent in the system i.e. at least one location is non-empty at a time.

EMAS may be modeled as the following tuple:

$$< U, Loc, Top, Ag, \{agsel_i\}_{i \in Loc}, locsel, \{LA_i\}_{i \in Loc}, MA, \omega, Act > \qquad (2)$$

where:

MA (master agent) is used to synchronize the work of the locations; it allows to perform actions in particular locations. This agent is also used to introduce necessary synchronization into the system.

$locsel : X \to \mathcal{M}(Loc)$ is the function used by MA to determine which location should be allowed to perform the next action.

LA_i (local agent) is assigned to each location; it is used to synchronize the work of computational agents present in its location, LA_i chooses the computational agent and lets it evaluate a decision and perform the action, at the same time asking MA whether this action may be performed.

$agsel_i : X \to \mathcal{M}(U \times P)$ is a family of functions used by local agents to select the agent that may perform the action, so every location $i \in Loc$ has its own function $agsel_i$. The probability $agsel_i(x)(gen, n)$ vanishes when the agent $ag_{gen,n}$ is inactive in the state $x \in X$ or it is present in other than i-th location,

$\omega : X \times U \to \mathcal{M}(Act)$ is the function used by agents for selecting actions from the set Act; both these symbols will be described later.

Act is a predefined, finite set of actions.

Here and later $\mathcal{M}(\cdot)$ stands for the space of probabilistic measures.

The population of agents is initialized by using introductory sampling. It may be explained as a one-time sampling from X according to the predefined probability distribution (possibly uniform) from $\mathcal{M}(X)$. Every agent starts its work in EMAS immediately after being activated. At every observable moment a certain agent on each location gains the possibility of changing the state of the system by executing its action.

The function $agsel_i$ is used by the Local Agent LA_i to determine which agent present on i-th location will be the next one to interact with the system. After being chosen, the agent $ag_{gen,n}$ chooses one of the possible actions according to the probability distribution $\omega(x, gen)$. Notice the relationship of this probability distribution with the concept of fine-grain schedulers introduced in the syntactic model for memetic algorithms in [11].

Next, the agent applies to LA_i for the permission to perform this action. When the permission is granted, $ag_{gen,n}$ checks whether the associated condition is true, and if so, the agent performs the action. The agent suspends its work in the system after performing the action which brings its energy to zero.

Master agent MA manages the activities of LA_i allowing them to grant permissions for their agents (thus relating to coarse-grain schedulers in [11]). Each action $\alpha \in Act$ is the pair of families of random functions $\{\delta_\alpha^{gen,n}\}_{gen \in U, n \in P}$ and $\{\vartheta_\alpha^{gen,n}\}_{gen \in U, n \in P}$ where

$$\delta_\alpha^{gen,n} : X \to \mathcal{M}(\{0,1\}) \qquad (3)$$

will denote the decision. The action α is performed with probability $\delta_\alpha^{gen,n}(1)$ by agent $ag_{gen,n}$ in state $x \in X$ i.e. when the decision is undertaken. Moreover

$$\vartheta_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{4}$$

defines the non-deterministic state transition caused by the execution of action α by agent $ag_{gen,n}$. The trivial state transition

$$\vartheta_{null} : X \to \mathcal{M}(X) \tag{5}$$

such that for all $x \in X$

$$\vartheta_{null}(x)(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

is performed with probability $\delta_\alpha^{gen,n}(x)(0)$, i.e. when decision δ_α is not undertaken ($\delta_\alpha^{gen,n}(x)$ is evaluated as zero).

The value of the probability transition function for action α for the agent containing the n-th copy of genotype gen being in the location l

$$\varrho_\alpha^{gen,n} : X \to \mathcal{M}(X) \tag{7}$$

for the arbitrary current state $x \in X$ and the next one $x' \in X$ is given by:

$$\varrho_\alpha^{gen,n}(x)(x') = \delta_\alpha^{gen,n}(x)(0) \cdot \vartheta_{null}(x)(x') + \delta_\alpha^{gen,n}(x)(1) \cdot \vartheta_\alpha^{gen,n}(x)(x') \tag{8}$$

Notice finally that it is formally possible to consider a very large (yet finite) set *Act*, comprising all actions up to a certain description length (using a Gödel numbering or any appropriate encoding). This implies that this set may be implicitly defined by such an encoding, allowing much flexibility in the set of actions available (a connection can be drawn with multimeme algorithms [10]).

The agents' actions may belong to one of two distinct types:

- global – they change the state of the system in two or more locations, so only one global action may be performed at a time,
- local – they change the state of the system inside one location respecting only the state of local agents, only one local action for one location may be performed at a time.

In the system governed by software agents there will be either a possibility of performing many local or one global action at a time.

3 EMAS Dynamics

At the observable moment at which EMAS takes state $x \in X$ all agents in all locations notify their local agents their intent to perform an action, all local agents choose an agent using the distribution given by the $agsel_i(x)$, $i \in Loc$ function and then notify the master agent of their intent to let perform an action by one of their agents. The master agent chooses the location using the probability distribution given by $locsel(x)$.

The probability that in the chosen location $i \in Loc$ the agent wants to perform a local action is as follows:

$$\xi_i(x) = \sum_{gen \in U} \sum_{n=1}^{p} agsel_i(x)(gen, n) \cdot \omega(x, gen)(Act_{loc}) \tag{9}$$

The probability that the master agent will choose the location with the agent intending to perform a local action is:

$$\zeta^{loc}(x) = \sum_{i \in Loc} locsel(x)(i) \cdot \xi_i(x) \tag{10}$$

Of course the probability of choosing a global action by the master agent is:

$$\zeta^{gl}(x) = 1 - \zeta^{loc}(x) \tag{11}$$

If a global action has been chosen, the state transition is as follows:

$$\tau^{gl}(x)(x') = \sum_{i \in Loc} locsel(x)(i)$$
$$\left(\sum_{gen \in U} \sum_{n=1}^{p} agsel_i(x)(gen, n) \cdot \left(\sum_{\alpha \in Act_{gl}} \omega(x, gen)(\alpha) \cdot \varrho_{\alpha}^{gen,n}(x)(x') \right) \right) \tag{12}$$

Let us state the set of action sequences containing at least one local action:

$$Act_{+1loc} = \left\{ (\alpha_1, \ldots, \alpha_s) \in Act^s;\ \sum_{i=1}^{s} [\alpha_i \in Act_{loc}] > 0 \right\} \tag{13}$$

Let us define now the family of coefficients $\{\mu_{\alpha_i, gen_i, n_i}(x)\}$, $i \in Loc$, $gen_i \in U$, $n_i \in P$, $x \in X$. If the location i is nonempty at the state x, then $\mu_{\alpha_i, gen_i, n_i}(x)$ is equal to the probability that in the i-th location agent ag_{gen_i, n_i} chooses action α_i:

$$\mu_{\alpha_i, gen_i, n_i}(x) = agsel_i(x)(gen_i, n_i) \cdot \omega(x, gen_i)(\alpha_i). \tag{14}$$

Of course $\mu_{\alpha_i, gen_i, n_i}(x) = 0$ if agent ag_{gen_i, n_i} does not exist in location i at state x, because $agsel_i(x)(gen_i, n_i) = 0$ in this case. Moreover, we set $\mu_{\alpha_i, gen_i, n_i}(x) = 1$ if location i is empty at state x. Next we introduce the multi-index:

$$ind = \big(\alpha_1, \ldots, \alpha_s; (gen_1, n_1), \ldots, (gen_s, n_s)\big) \in IND = Act_{+1loc}^{s} \times (U \times P)^s. \tag{15}$$

The probability that at state x, in consecutive locations agents ag_{gen_i, n_i} choose actions α_i is given by:

$$\mu_{ind}(x) = \prod_{i=1}^{s} \mu_{\alpha_i, gen_i, n_i}(x) \tag{16}$$

the transition function for the case of parallel executing of local actions is then:

$$\tau^{loc}(x)(x') = \sum_{ind \in IND} \mu_{ind}(x)(\pi_1^{ind} \circ, \ldots, \circ \pi_s^{ind})(x)(x') \tag{17}$$

where

$$\pi_i^{ind}(x) = \begin{cases} \varrho_{\alpha_i}^{gen_i, n_i}(x), & \alpha_i \in Act_{loc} \text{ and the location } i \text{ is nonempty} \\ \vartheta_{null}, & \alpha_i \in Act_{gl} \quad \text{or} \quad \text{the location } i \text{ is empty.} \end{cases} \tag{18}$$

It is possible to prove that the value of $(\pi_1^{ind} \circ, \ldots, \circ \pi_s^{ind})(x)(x')$ does not depend on the composition order because transition functions associated with local actions commutate pairwise. The proof of this property in the discrete model is similar to the proof in [21] for a continuous system state space, and is omitted here due to space constraints.

The commutativity of local action validates the following observation:

Observation 1. *The probability transition function for the parallel EMAS model is given by formula*

$$\tau(x)(x') = \zeta^{gl}(x) \cdot \tau^{gl}(x)(x') + \zeta^{loc}(x) \cdot \tau^{loc}(x)(x') \tag{19}$$

and formulas (9)–(18).

It is also easy to see that

Observation 2. *The stochastic state transition of EMAS given by formula (19) satisfies the Markov condition. Moreover, the Markov chain defined by these functions is stationary.*

4 Sample Actions and Asymptotic Behavior

Let us consider a sample EMAS with the following set of actions:

$$Act = \{get, repr, clo, migr\} \tag{20}$$

Due to space limitations we describe the actions informally, underlining only the necessary conditions for the subsequent analysis of the systems's ergodicity. Complete formal descriptions of these actions leading to the probability transition functions (3) and (4) may be found in [4]. In the following (gen, n) stands for the signature of a generic agent that attempts to execute the following actions:

get Decision $\delta_{get}^{gen,n}$ for energy transfer is positive when there is at least one agent more on the same location. Agent chooses randomly one of its neighbors and during the meeting, the energy is exchanged between agents, what may be considered somewhat as a tournament (see tournament selection [12]). The direction of the energy flow is determined by a probability distribution $CMP : U \times U \to \mathcal{M}(\{0, 1\})$ dependent on agents' fitnesses and the current state of the system. In the next state one of the agents receives a predefined part of energy Δe from its neighbor, which is assumed to satisfy $\Delta e \leq (\sum_{i=1}^{s} q_i)^{-1}$.

repr Decision $\delta_{repr}^{gen,n}$ for reproduction is positive when the energy of the agent performing the action is greater than a reproduction threshold e_{repr} and there is at least one agent more in the same location satisfying the same energy condition. We assume that $e_{repr} \leq 2\Delta e$. These agents create an offspring agent based on their solutions using a predefined mixing operator. Part of the parents' energy ($e_0 = n_0 \cdot \Delta e$, n_0 is even) is passed to the offspring.

clo Decision $\delta_{clo}^{gen,n}$ for cloning is based on checking the amount of agent's energy only. An agent with enough energy strictly greater than Δe, creates an offspring agent based on its solution (applying a predefined mutation operator $MUT : U \to \mathcal{M}(U)$). Part of the parent's energy Δe is passed to the offspring.

migr Decision $\delta_{migr}^{gen,n}$ is positive when an agent has enough energy greater than e_{migr} and there exists a location that is able to accept it (the number of agents there is lower than its capacity). When these conditions are met the agent is moved from its location to another. We assume, that $e_{migr} < s^{-1}$.

Theorem 1. *Given the following assumptions:*

1. *The capacity of every location is greater than one,* $q_i > 1, i = 1, \ldots, s$.
2. *The graph of locations is connected.*
3. *Each active agent can be selected by its local agent with strictly positive probability, so*
   $\exists\ \iota_{agsel} > 0;\ \forall\ i \in Loc, \forall\ gen \in U, \forall\ n \in P,\ \forall\ x \in \{y \in X; y(i, gen, n) > 0\}$,
   $agsel_i(x)(gen, n) \geq \iota_{agsel}$.
4. *The families of probability distributions being the parameters of EMAS have the uniform, strictly positive lower bounds:*
   $\exists\ \iota_\omega > 0;\ \forall\ x \in X,\ gen \in U,\ \alpha \in Act,\ \omega(gen, x)(\alpha) \geq \iota_\omega$,
   $\exists\ \iota_{CMP} > 0;\ \forall\ gen, gen' \in U,\ CMP(gen, gen') \geq \iota_{CMP}$,
   $\exists\ \iota_{mut} > 0; \forall gen, gen' \in U,\ MUT(gen)(gen') \geq \iota_{mut}$,
   $\exists\ 0 < \iota_{locsel} < 1; \forall\ x \in X, \forall\ j \in Loc,\ locsel(x)(j) \geq \iota_{locsel}$.

We can construct a finite sequence of transitions between two arbitrarily chosen system states which may be passed with strictly positive probability. Moreover we can deliver the upper bound of the number of such transitions, which can be effectively computed based on the system's parameters.

The proof of the Theorem 1 is omitted in this paper because of its length and strictly technical substance. It has already been completed and will be published in an extended version.

Assumptions 1 and 2 allow to migrate agents to all locations that are not overpopulated (with a positive probability). The positive probability of performing crucial actions (*get*, *clo*) changing energy and genotype is ensured by assumptions 3 and 4. The above stated properties make possible to define a generic path between two arbitrary states of the system.

Notice that verifying the ergodicity is different than usually done for classical genetic algorithms (see e.g. [23]), where all possible states of the system are reachable within a single step, because of the characteristics of the mutation operator.

Remark 1. Theorem 1 makes all states containing the extrema reachable in a finite number of states, thus EMAS satisfies an asymptotic guarantee of success [8], [18]. Moreover the Markov chain modelling EMAS (see equation (19)) is ergodic.

5 Conclusions

We presented a discrete version of EMAS model (following the continuous versions of the model published in [4,21]). The space of states of the system – Eq. (1) – and the probability transition function – Eq. (19) – constitute a stationary Markov chain.

Under assumptions of Theorem 1 an EMAS is able to reach the population containing an arbitrary minimizer in a finite number of steps. The effective upper bound for the number of steps required may also be evaluated. In addition, asymptotic guarantee of success is satisfied [8,18]. The properties mentioned above make the Markov chain modelling EMAS ergodic. The ergodicity in the case of EMAS is not as straightforward as in classical genetic algorithms (cf. the works of Vose [23]) where any possible state of the system may be reached in one step thanks to positive mutation rates.

In the course of modelling several constraints were indicated leading to better understanding of the functioning of agent-based memetic systems (e.g. synchronization schemes, probability distributions used and topology of connections).

References

1. Berretta, R., Cotta, C., Moscato, P.: Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm: Results on the number partitioning problem. In: Resende, M., Pinho de Sousa, J. (eds.) Metaheuristics: Computer-Decision Making, pp. 65–90. Kluwer Academic Publishers, Boston (2003)
2. Berretta, R., Moscato, P.: The number partitioning problem: An open challenge for evolutionary computation ? In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 261–278. McGraw-Hill, New York (1999)
3. Byrski, A., Kisiel-Dorohinicki, M.: Agent-based evolutionary and immunological optimization. In: Proceedings of Computational Science - ICCS 2007, 7th International Conference, Beijing, China, May 27-30. Springer, Heidelberg (2007)
4. Byrski, A., Schaefer, R.: Stochastic model of evolutionary and immunological multi-agent systems: Mutually exclusive actions. Fundamenta Informaticae 95(2-3), 263–285 (2009)
5. Cetnarowicz, K., Kisiel-Dorohinicki, M., Nawarecki, E.: The application of evolution process in multi-agent world (MAW) to the prediction system. In: Tokoro, M. (ed.) ICMAS 1996 Proceedings. AAAI Press, Menlo Park (1996)
6. Chira, C., Gog, A., Dumitrescu, D.: Exploring population geometry and multi-agent systems: a new approach to developing evolutionary techniques. In: Proceedings of the 2008 GECCO Conference Companion on Genetic and Evolutionary Computation, Atlanta, GA, pp. 1953–1960. ACM Press, New York (2008)

7. Hart, W.E., Krasnogor, N., Smith, J.E.: Memetic evolutionary algorithms. In: Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166, pp. 3–27. Springer, Heidelberg (2005)
8. Horst, R., Pardalos, P.: Handbook of Global Optimization. Kluwer, Dordrecht (1995)
9. Kisiel-Dorohinicki, M.: Agent-oriented model of simulated evolution. In: Grosky, W.I., Plášil, F. (eds.) SOFSEM 2002. LNCS, vol. 2540, pp. 253–261. Springer, Heidelberg (2002)
10. Krasnogor, N., Gustafson, S.: A study on the use of "self-generation" in memetic algorithms. Natural Computing 3, 53–76 (2004)
11. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
12. Michalewicz, Z.: Genetic Algorithms Plus Data Structures Equals Evolution Programs. Springer, New York (1994)
13. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Technical Report Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989)
14. Moscato, P.: Memetic algorithms: A short introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 219–234. McGraw-Hill, New York (1999)
15. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 105–144. Kluwer Academic Publishers, Boston (2003)
16. Norman, M.G., Moscato, P.: A competitive and cooperative approach to complex combinatorial search. Technical Report Caltech Concurrent Computation Program, Report. 790, California Institute of Technology, Pasadena, California, USA (1989)
17. Radcliffe, N.J., Surry, P.D.: Formal Memetic Algorithms. In: Fogarty, T.C. (ed.) AISB-WS 1994. LNCS, vol. 865, pp. 1–16. Springer, Heidelberg (1994)
18. Rinnoy Kan, A., Timmer, G.: Stochastic global optimization methods. Mathematical Programming 39, 27–56 (1987)
19. Rudolph, G.: Models of stochastic convergence. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computations. Oxford University Press, Oxford (1997)
20. Rudolph, G.: Stochastic processes. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computations. Oxford University Press, Oxford (1997)
21. Schaefer, R., Byrski, A., Smołka, M.: Stochastic model of evolutionary and immunological multi-agent systems: Parallel execution of local actions. Fundamenta Informaticae 95(2-3), 325–348 (2009)
22. Barkat Ullah, A.S.S.M., Sarker, R., Lokan, C.: An agent-based memetic algorithm (AMA) for nonlinear optimization with equality constraints. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, Trondheim, Norway, pp. 70–77. IEEE Press, Los Alamitos (2009)
23. Vose, M.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge (1998)

Path-Guided Mutation for Stochastic Pareto Local Search Algorithms

Madalina M. Drugan and Dirk Thierens

Department of Information and Computing Sciences, Utrecht University,
PO Box 80.089, 3508TB Utrecht, The Netherlands
{madalina,dirk}@cs.uu.nl

Abstract. *Stochastic Pareto local search* (SPLS) methods are local search algorithms for multi-objective combinatorial optimization problems that restart local search from points generated using a stochastic process. Examples of such stochastic processes are Brownian motion (or random processes), and the ones resulting from the use of mutation and recombination operators. We propose a path-guided mutation operator for SPLS where an individual solution is mutated in the direction of the path to another individual solution in order to restart a PLS. We study the exploration of the landscape of the *bi-objective Quadratic assignment problem* (bQAP) using SPLSs that restart the PLSs from: i) uniform randomly generated solutions, ii) solutions generated from best-so-far local optimal solutions with uniform random mutation and iii) with *path-guided mutation*. Experiments on a bQAP with a large number of facilities and high correlation between the flow matrices show that using mutation, and especially path-guided mutation, is beneficial for performance of SPLS. The performance of SPLSs is partially explained using their dynamical behavior like the probability of escaping the local optima and the speed of enhancing the Pareto front.

1 Introduction

Stochastic local search algorithms, SLS, are among the most popular techniques for solving combinatorial optimization problems in many areas from computer science, operations research, engineering, physics, etc. A local search algorithm iteratively moves from a current solution to a neighboring solution. The algorithm stops when no improving solution can be found in the neighborhood. Some of SLS's properties that make it so successful in all these areas are: simplicity, ease of understanding and in implementation, flexibility in design, and power of generalization. SLSs for multi-objective spaces [1] are called *stochastic Pareto local search* (SPLS) [2,3,4,5].

To enhance the efficiency of multi-start PLS, we want to exploit the structure of the landscape with suitable exploration operators like for instance local perturbation by mutation. The advantage of mutation-generated restarts is that the local search is restarted from nearby, and therefore most likely, correlated areas of the landscape. We call algorithms that mutate the local optima to restart Pareto local search, *iterated PLS* (IPLS), as they are a straightforward extension of *iterated local search* algorithms for single objective spaces [6,7]. In this paper we compare uniform random mutation with a path-guided mutation that generates solutions on the path linking two local optimal individuals. This operator resembles the path relinking operator [8], but the *path-guided mutation* generates individuals at a certain distance from a parent on the path to the second parent. In this paper, we want to investigate the use of path-guided mutation in multi-objective optimization by stochastic Pareto local search. Specifically, we compare PLS with IPLS and the path-guided pIPLS on a bi-dimensional quadratic assignment problem (bQAP) where the dimensions have common structure (i.e., the multi-dimensional QAPs have correlated flow matrices [9]). The solutions of QAPs can be represented by permutations of facilities to different positions. Consequently, the perturbation operators, mutation and recombination, we use here are tailored to this type of representation. We measure the performance of the three SPLS with the unary *hypervolume* indicator and *attainment* functions [10]. We also track the impact of the different mutation operators and different mutation rates on the search efficiency. We propose to connect the dynamical exploitation - the number of Pareto front enhancements - and exploration - the probability of escaping the local optima - of the landscape with the performance of SPLSs. As expected, the results show that multi-start PLS is easily outperformed by IPLS using either uniform random mutation or path-guided mutation. Furthermore, IPLS using path-guided mutation outperforms uniform random IPLS by exploiting the commonalities in the correlated search spaces.

In the next section, we describe the PLS algorithm and its use for multi-objective QAPs. In Section 3, we describe the IPLS algorithms and their use on multi-objective QAPs. In Section 4, we show experimental results and correlate the performance with the dynamical behaviour of SPLS algorithms. Section 5 concludes the paper.

2 The Multi-start Pareto Local Search Algorithm (PLS)

In [11,12], Paquete et al. introduce the Pareto local search algorithm (PLS). A PLS starts from a (randomly generated) initial solution and iteratively generates new solutions using a neighborhood function. In the multi-objective space, solutions can be better than other solutions in some dimensions but worse in other dimensions.

We denote with s a solution and $\mathcal{N}(s)$ a neighborhood of s. The solution s has attached a flag, $s.visited$ which is set to $true$ after evaluating the entire neighborhood of s and $false$ otherwise. We say that a solution s *dominates* another solution s' if s is at least as "good" on all objectives as well as being "better" on at least one. We say that s does *non-dominated* s' if there exists at least one dimension in which s is "worse" than s'. The set of solutions that dominates the other solutions in at least one dimension (or objective) is called the *non-dominated archive* (NDA). We say that s is dominated by the NDA, if it is dominated by at least one solution in the NDA. Finally, s is not dominated by the NDA, if it is not dominated by any solution in the NDA.

The NDA is initialized with a solution s that is generated randomly. Each iteration, a solution s with the visited flag set to false is randomly chosen from the NDA. All the neighborhood solutions, s', of s are evaluated. s' will be added to the NDA if s' is not dominated by any solution in the current NDA, and the solutions dominated by s' are removed. If s' is dominating all other solutions in the NDA, s' will be the only remaining solution in the NDA. The search continues until there are no solutions left in the NDA that have their visiting flag set to false. Note that PLS is a best improvement algorithm because it selects all non-dominated neighboring solutions. The PLS algorithm stops in a local optimal NDA set. To find the best optima of the search space, the *multi-restart PLS* algorithm is restarted multiple times from $M > 0$ uniformly random initial solutions.

2.1 PLS for Multi-objective Quadratic Assignment Problem (mQAP)

Single and multi-dimensional QAPs are NP-hard combinatorial optimization problems that model many real-time problems (i.e., computer aided design in the electronic industry, scheduling, vehicle routing, etc.). Intuitively, QAPs can be described as the (optimal) assignment of N facilities to N locations. A distance is specified between each pair of locations, and for each pair of facilities the amount of materials (or flows) transported between these facilities is given. The goal is to find the assignment of facilities to locations that minimizes the sum of the products between distances and flows.

We consider the multi-dimensional QAPs (mQAPs) introduced by Knowles and Corne [9]. These mQAPs have for each dimension different flow matrices and a single distance matrix. The flow matrices are correlated with some correlation ρ. Let us consider n facilities, a set $\Pi(n)$ of all permutations of $\{1, 2, \ldots, n\}$ and the $n \times n$ distance matrix $A = (a_{ij})$, where a_{ij} is the distance between location i and location j. We assume an m-dimensional space, and m flow matrices $B^k = (b^k_{ij})$, each with $n \times n$ elements, where b^k_{ij} represents the flow from facility i to facility j in the k-th objective dimension. The goal is to minimize for all dimensions the set of functions

$$C^k(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot b^k_{\pi(i)\pi(j)}$$

where $\pi(\cdot)$ is a permutation from $\Pi(n)$. It takes quadratic time to evaluate this function. QAPs are permutation problems, and a suitable neighborhood operator for PLS is the exchange operator that swaps the position of two or more facilities. For example, the 2-exchange swapping operator, swaps the position of two different facilities. The 2-exchange operator is attractive because of its linear time to compute the change in the cost function with the condition that all matrices A and B^k are symmetrical [2].

3 Iterated Pareto Local Search (IPLS)

Restarting PLS from randomly generated solutions is basically random sampling in the space of local optima. To improve the efficiency, we need to exploit the structure of the search space. Iterated PLS uses mutation operators to generate starting points for the PLS algorithm. First, PLS is restarted N times, $0 < N < M$, from uniform randomly generated points to construct a "good" and diverse initial NDA. Then, a solution is uniform randomly chosen from the current NDA and a new individual is generated with some mutation operator. PLS is restarted from this individual. This process of restarting PLS from mutations of NDA solutions is repeated until a stopping criterium is met. In Section 3.1, two IPLS instances for mQAPs are proposed.

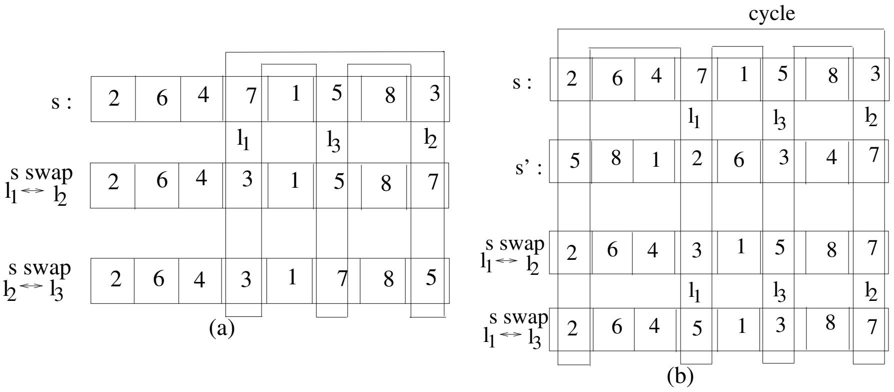

s :
s swap $l_1 \leftrightarrow l_2$
s swap $l_2 \leftrightarrow l_3$
l_1
l_3
l_2
(a)
cycle
s' :
s swap $l_1 \leftrightarrow l_3$
(b)

Fig. 1. (a) An example of 3-exchange mutation. Three positions l_1, l_2 and l_3 are chosen at random and shifted to the right. (b) An example of 3-exchange path mutation. The main parent s and s' form two cycles from which $cycle = \{2, 3, 5, 7\}$ is randomly chosen. The three positions l_1, l_2 and l_3 in $cycle$ are used to decrease the distance between s and s'.

3.1 Iterated PLS for Multi-objective QAPs

In permutation problems like (m)QAPs, the mutation operator interchanges facilities between different positions. When PLS uses the 2-exchange operator to generate a neighborhood, the mutation operator should exchange at least 3 facilities to escape from the region of attraction of the local optima. IPLS denotes the algorithms that use an m-exchange operator, $m > 2$, to restart PLS. The stopping criterium for IPLS is chosen to fairly compare its performance to PLS algorithms. The distance between two solutions is defined as the minimum number of exchanges to obtain one solution from another and is computed as in [13]. The distance between a solution and the solution obtained with 2-exchange mutation is 1, meaning that one swap is necessary. In general, the distance between a solution and its m-exchange solution is $m - 1$. The search in IPLS is halted when the same number of swaps is executed as with PLS. The difference in cost function, ΔC^k, for two solutions with distance m is linear in the number of facilities and the number of exchanges. Counting the number of swaps is equivalent to counting the number of function evaluations.

The m-exchange mutation uniform randomly selects without replacement m distinct locations in a solution s, $\{l_1, \ldots, l_m\}$, where $m > 2$. To generate a new solution, these locations are exchanged from left to right or from right to left with equal probability to not bias the generation of individuals. When positions are exchanged from right to left, a position l_i takes the value of its right neighbor l_{i+1}. Then

$$temp = l_1;\ \ l_1 = l_2;\ \ \ldots\ \ ; l_{m-1} = l_m;\ \ l_m = temp$$

where $temp$ is a buffer variable. An example of 3-exchange mutation is given in Figure 1(a). Note that s and the resulting solution form a cycle of size m that contains the mutated positions $\{l_1, \ldots, l_m\}$ and are m swaps apart.

The path-guided m-exchange mutation uses two uniform randomly selected solutions without replacement from the current NDA where the distance between them is at least m. One of the solutions, we call it the parent solution, will be mutated. The other solution is only used to construct the path between the two solutions. At first, the set of *cycles* common for the two solutions are identified. A cycle is a minimal set of facilities such that the set of locations associated with that set of facilities in both parents is the same. It is possible to switch that subset from one parent to the other one while keeping a valid permutation. For example, in Figure 1 (b) there are two cycles between s and s': i) $\{2, 7, 5, 3\}$ and ii) $\{6, 8, 4, 1\}$. Next, a cycle, c, with a size larger than 1 is randomly chosen. If the size of c, ℓ_c, is larger than m, $\ell_c > m$, we uniform randomly select one position, i, in the cycle. For $m - 1$ times, the i-th value in the parent solution is exchanged with the j-th value of the same solution, where the j-th position in the second solution has the same value as the i-th position in the parent solution. If the size of c is smaller or equal than m, $\ell_c \leq m$ the whole cycle is swapped with the second child. This process of randomly selecting a cycle and swap locations is repeated untill the distance between the parent s and its child is m. The generated solution s'' is at $m - 1$ distance from the parent solution s and $\ell_c - m + 1$ distance to the second solution s' since the positions that were exchanged have the same value as the second parent. An example of 3-exchange path mutation is given in Figure 1(b).

This path mutation resembles path relinking and cycle crossover. In path relinking, all the individuals on the path will be generated. In cycle crossover, entire cycles are swapped with some probability. Here we study the search efficiency of different m-exchanges on bi-dimensional QAPs (bQAP) with correlated flow matrices.

4 Experimental Results

Although the principles discussed and the algorithms proposed in this paper are general, we limit our experiments to two objectives because is easier to visualize the results of the algorithms.

The tested problems. We compare the mentioned SPLS algorithms on bQAP instances generated using the software of Knowles and Corne [9]. This bQAP has high positive correlations $\rho = \{0.75\}$ and a large number of facilities $n = \{50\}$. To facilitate comparisons, we used the same problems as the unstructured bQAP instances in Paquete's study [2] (*http://eden.dei.uc.pt/paquete/qap/*). For QAPs with a large number of facilities and high positive correlation Paquete reported a poor performance of multi-restart PLSs. In the following, we show that restarts generated with m- exchange mutation and path mutation outperform PLS by exploiting the structure in the search space.

The four tested algorithms. The multi-restart PLS, *PLS*, with a best improvement 2-exchange neighborhood is restarted M times. The number of swaps S is counted. For *IPLS*, at first, N PLSs are uniform randomly restarted. Next, until S swaps are reached, PLSs are restarted from a mutated solution from the current NDA with an m-exchange operator, $m > 2$. In *pIPLS*, after randomly restarting N PLSs, the algorithm is run S swaps where the restarts are generated with the m-exchange path guided mutation. In *rIPLS*, after randomly restarting N PLSs, the algorithm is run S swaps and the restarting

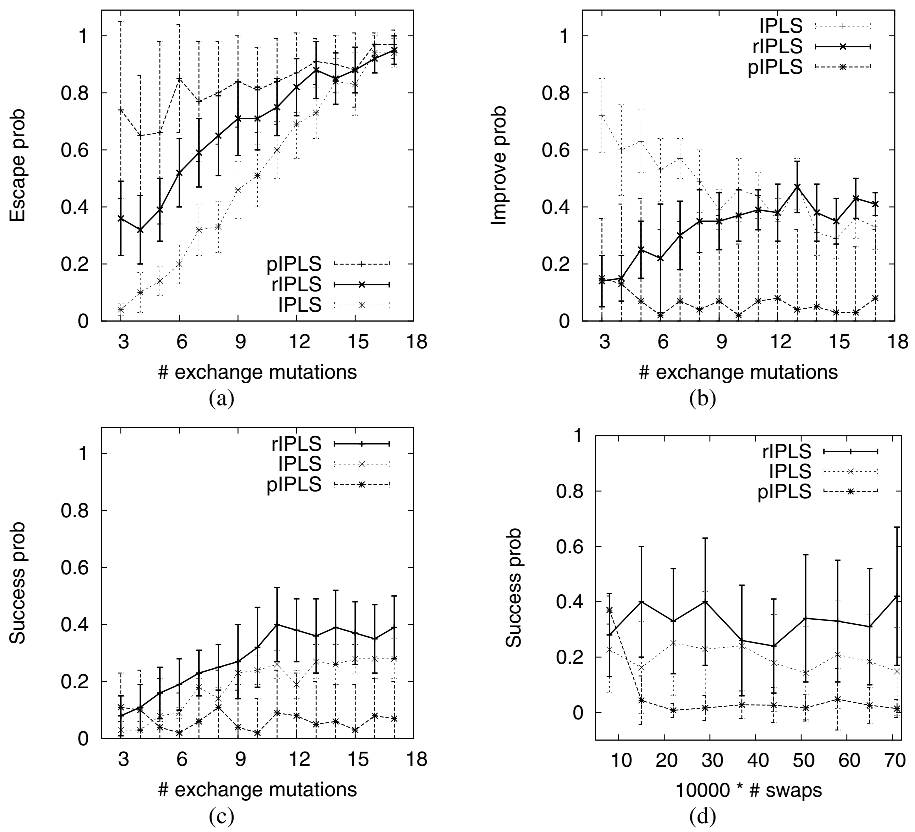


Fig. 2. The (a) escape, (c) success and (b) improve probabilities for 15 values of m for *IPLS*, *pIPLS* and *rIPLS* with $N = 10$ and $M = 20$. (d) The success probabilities with the number of swaps for $m = 10$ for the three algorithms.

points are generated 50% with m-exchange mutation and 50% with the m-exchange path-guided mutation.

rIPLS is a combination of *IPLS* and *pIPLS* algorithms, because *pIPLS* can be stuck in a number of situations: i) the NDA size is 1, ii) all solutions in NDA have distances smaller than m and iii) exploring the restarts on the path between NDA solutions does not generate new non-dominated solutions. The three IPLS algorithms (*IPLS*, *pIPLS* and *rIPLS*) are run 15 times with m-exchange mutations between 3 and 17. For high exchange rates the m-exchange operator is almost equivalent with a random generator. Each instance of the four algorithms is run 50 times.

The exploitation and exploration characteristics are measured by the success, improvement and escape probability as shown in Figure 2. The *escape probability* is the probability that the solution after mutation does not belong to the basin of attraction of its parent (= $\#escapes/\#restarts$, where $\#restarts$ are the number of times *PLS* is restarted in a run). The *success probability* is the probability that the new starting

Table 1. *PLS* vs *IPLS* and *rIPLS* using several measurements like: i) the hypervolume unary indicator (hypervolume), ii) the average number of restarts for PLS (#restarts), iii) The average number of neighborhood search per PLS (#neigh search/PLS), and iv) the average number of NDA enhancements (#NDA enhancements). The table contains the maximum, minimum and average values over all value of m-exchanges for *IPLS* and *rIPLS* algorithms.

	PLS	*IPLS*			*rIPLS*		
		max	min	aver	max	min	aver
hyper	0.45 ± 0.05	0.81 ± 0.13	0.48 ± 0.07	0.62 ± 0.12	0.89 ± 0.14	0.54 ± 0.08	0.70 ± 0.14
restart	100 ± 0	971 ± 204	184 ± 7	445 ± 241	1973 ± 445	254 ± 31	699 ± 475
neigh	57 ± 2	36 ± 1	9 ± 1	21 ± 9	26 ± 4	6 ± 1	16 ± 7
enhan	8 ± 7	52 ± 14	26 ± 19	44 ± 8	153 ± 36	49 ± 11	109 ± 27

solution escapes from the basin of attraction, and the local optima generated by PLS are non-dominated by the current NDA (= $\#success/\#restarts$). The *improvement probability* is the probability that, *if* the restart escapes from the basin of attraction of its parent, the generated local optima from that restart are non-dominated by the current NDA (= $\#success/\#escapes$). Because *pIPLS* can get stuck after few restarts, we have chosen $N = 10$ for an initial NDA with diverse solutions - 5-15 solutions - and $M = 20$. On average, the number of swaps for 10 multi restarts of PLS are about $7 * 10^5$ with a variation of about 10^5 swaps. The measurements are started after the initial phase of N swaps that are multi restarted PLSs for all the algorithms.

In Figure 2(a), the escape probability of *pIPLS* is about 1 which means that m-exchange path guided is proposing solutions that do not belong to the parents' basin of attraction. On the other hand, the escape probaility of *IPLS* varies very much with m. For small m, the probability of escaping the local optima is close to 0 and for m large it is close to 1. On the other hand, Figure 2(b) reveals that, when escaping, the improvement probability of *IPLS* is larger than of *rIPLS* and *pIPLS*. Furthermore, the improvement probability decreases when m increases for *IPLS*, whereas this probability increases when m increases for *rIPLS*. The success probability for *rIPLS* is the highest, see Figure 2(c), suggesting the *rIPLS* is enhancing its NDA more often than the other two algorithms. By definition, for a given m, the success probability from Figure 2(c) are equivalent with the escape probability from Figure 2(a) multiplied with the improvement probability from Figure 2(b). The bad performance of pIPLS, as seen in Figure 2 (d), is due to the fact that it gets stuck very quickly (after about $3 * 10^5$ swaps) in local optima where no improvements are possible. Clearly, one needs to mix uniformly random mutation for exploration and path-guided mutation for exploitation of the search space. In the following paragraph, we compute the performance of the *IPLS* and *rIPLS* algorithms and investigate their success and improvement probabilities.

Performance assessment. The *hypervolume* indicator is a unary performance measure designed to compare the Pareto fronts given by multi-objective combinatorial optimization algorithms [10]. The larger the hypervolume, the larger is the volume between a reference point, in this case the worst point in the bi-variate normalized search space, and the Pareto front. The larger the hypervolume the better the algorithm performs. Another method to compare the performance of two (and more) algorithms are the *attainment functions* (EAF). The unary *attainment function* gives the probability of attaining

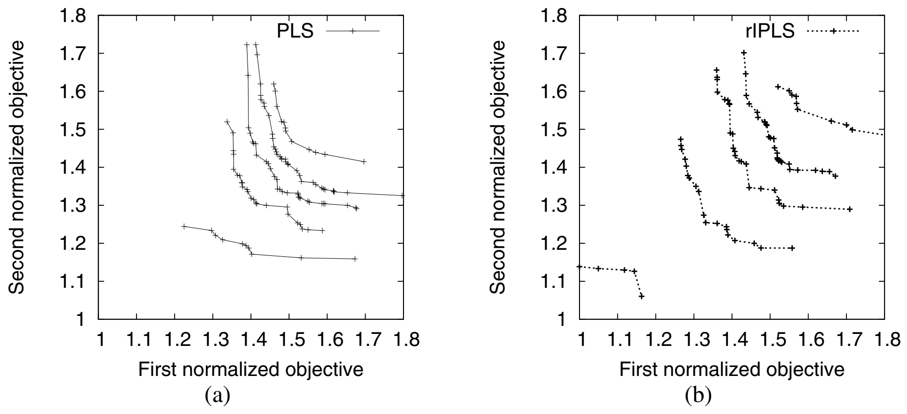


Fig. 3. Attainment surfaces at 1%, 25%,50%, 75% and 100% (lines from bottom left to top right) of the normalized outputs of a) *PLS* and b) *rIPLS* with $m = 10$. The objective values found at 1% EAFs correspond to the best NDA found over 50 runs and those found at 50% EAFs are the median outcome.

each point (independently) in the objective space. Certain contour surfaces through certain probabilities can then be drawn. For comparison purposes, a normalization function assigns to the best point(s) in a dimension the value 1 and to the worst point(s) the value 2. All the other points are scaled to a value between 1 and 2 in both dimensions, and the reference point is $\{2, 2\}$.

For this second experiment, the performance of three algorithms - that are *PLS*, *IPLS* and *rIPLS* - are compared. *PLS* is restarted for $M = 110$ runs - that is about $77 * 10^5$ swaps. *IPLS* and *rIPLS* are run also for $77 * 10^5$ swaps, where $N = 10$.

Table 1 records the hypervolume of *PLS* and the maximum, minimum and the average of *IPLS* and *rIPLS*, whereas Figure 4(a) shows the hypervolume from small ($m > 2$) to large exchange rates ($m = 17$) for *IPLS* and *rIPLS*. Mann-Whitney nonparametric two-sided test for unary hypervolume indicators with significance level $p < 0.05$ compares the outputs of: i) m instances of *rIPLS* with *PLS*, ii) m instances of *rIPLS* with *PLS*, and iii) m instances of *rIPLS* with corresponding instances of *IPLS*. The hypervolume of *PLS* is significantly smaller than the hypervolume of *rIPLS* for all the 15 tested m values. 14 from 15 instances of *IPLS* have the hypervolume statistically higher than *PLS*. 10 from 15 instances of *rIPLS* have the hypervolume higher than *IPLS* with the same m-exchange value. Both *PLS* and *IPLS*'s hypervolumes are significantly smaller than the average hypervolume of *rIPLS*. In Figure 3, the EAFs of *PLS* have larger, and thus worse, values that EAF's of *rIPLS* showing that *rIPLS* is finding faster good (smaller) solutions than *PLS*.

The performance of the three algorithms can be partly explained with aspects of their dynamical behavior. PLSs are always restarted 100 times, whereas *rIPLS* restarts and improves the NDAs more often than the two other algorithms. In Table 1 and Figure 4(c), *IPLS*s are restarted, on average, about 2-3 times more often than *PLS*, and *rIPLS*s even 4-6 times more often. In Table 1 and Figure 4(d), the number of neighborhoods explored within PLS is the lowest for *rIPLS* and the highest for *PLS*. For m

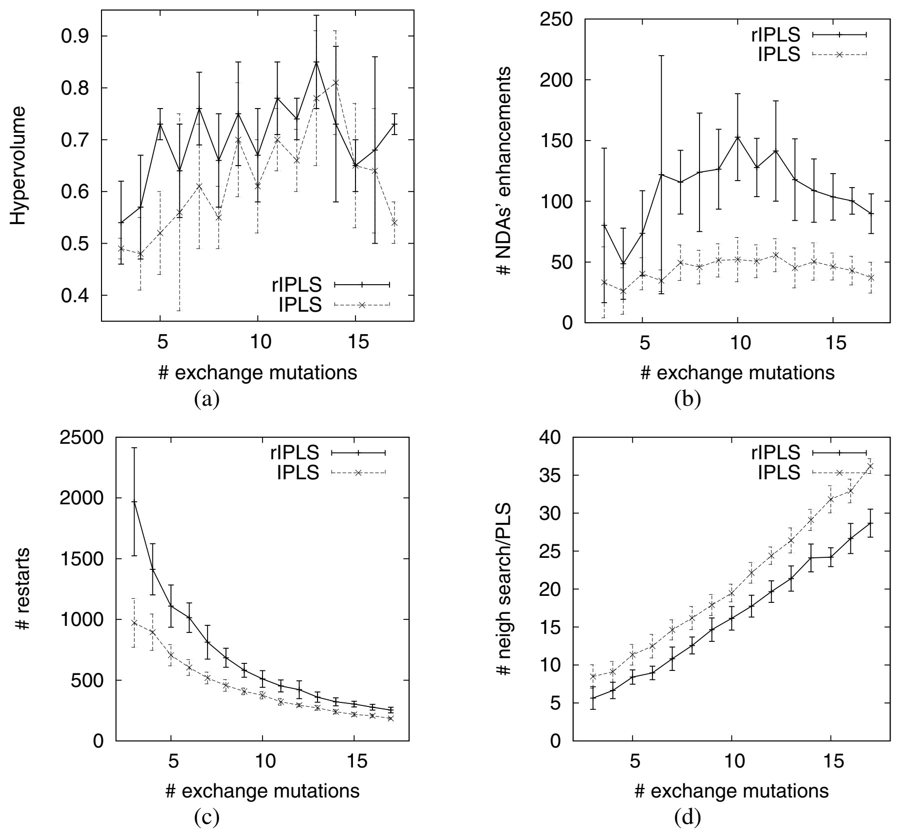


Fig. 4. a) The unary hypervolume indicator, b) the average number of enchangements in current NDA, c) the average number of restarts per run and d) the average number of neighbourhood exploration per PLS for 15 exchange mutations

large, the number of restarts multiplied with the number of neighborhood searches is about 570 for *IPLS*, *rIPLS*, and *PLS*. The product increases for small values of m. Also in Figure 4 (b) and Table 1, the number of times the NDA is improving is the largest for *rIPLS* and both *IPLSs* improve their Pareto front more often than *PLS*. For a given m, the success probability from Figure 2 (c) multiplied with the number of restarts from Figure 4 (c) is equal with the number of NDA enhancements from Figure 4 (b).

We conclude the *rIPLS* is the most performant algorithm tested because of the high escape and success rates, and a larger number of restarts, leading to a larger number of Pareto front enhancements. That means that the IPLS and rIPLS algorithms perform their best when the used exchange mutation is below half of the number of facilities $n/2$. This is reflected in the larger hypervolume indicator.

5 Conclusions

In this paper, we have introduced a path-guided mutation that mutates a solution in the direction of another solution in the population. We have applied it in a multi-objective setting, where the mutated and the guiding solutions are members of the non-dominated Pareto set. Path-guided mutation exploits the structure of the landscape since it protects the commonalities between the mutated and guiding solutions. At the same time it explicitly specifies the mutation step size. These properties make it well suited to be applied as perturbation operator for iterated Pareto local search algorithms. To balance the exploitation and exploration of the IPLS, a mixture of path-guided mutation and uniform random mutation is used. We have tested the algorithm on instances of the bivariate quadratic assignment problem. We proposed to associate the performance of an algorithm with some simple measurements like the escape probability from a local optimum and the speed of NDA enhancements. As expected, the *IPLS* with uniform random mutation, and the *rIPLS* with mixed path-guided and uniform random mutation, both outperform the multi-restart *PLS*. The *rIPLS* also outperforms the *IPLS* because it has a higher escape probability and a higher success probability than *IPLS*. As future work, we want to develop an algorithm that adapts its mutation to escape local optima based on the speed of NDA enhancements.

References

1. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
2. Paquete, L., Stutzle, T.: A study of stochastic local search algorithms for the biobjective QAP with correlated flow matrices. European J. of Oper. Res. 169(3), 943–959 (2006)
3. Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.: ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In: Advances in Multi-objective Nature Inspired Computing, pp. 386–400. Springer, Heidelberg (2006)
4. Li, H., Landa-Silva, D.: An Elitist GRASP Metaheuristic for the Multi-objective Quadratic Assignment Problem. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 481–494. Springer, Heidelberg (2009)
5. Ehrgott, M., Gandibleux, X.: Hybrid metaheuristics for multi-objective combinatorial optimization. In: Hybrid Metaheuristics, pp. 221–259 (2008)
6. Merz, P.: Advanced fitness landscape analysis and the performance of memetic algorithms. Evolutionary Computation 12(3), 303–325 (2004)
7. Stützle, T.: Iterated local search for the quadratic assignment problem. European J. of Oper. Res. 174(3), 1519–1539 (2006)
8. Jaszkiewicz, A., Zielniewicz, P.: Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. European J. of Oper. Res. 193(3), 885–890 (2009)
9. Knowles, J., Corne, D.: Instance generators and test suites for the multiobjective quadratic assignment problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 295–310. Springer, Heidelberg (2003)

10. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Trans. on Evol. Comp. 7(2), 117–132 (2003)
11. Paquete, L., Chiarandini, M., Stützle, T.: A study of local optima in the multiobjective traveling salesman problem. Technical Report AIDA-02-07, Fachgebiet Intellektik, Fachbereich Informatik, Technische Universität Darmstadt (2002)
12. Paquete, L., Stützle, T.: Stochastic local search algorithms for multiobjective combinatorial optimization: A review. In: Handbook of approximation algorithms and metaheurististics. Chapman & Hall/CRC, Boca Raton (2007)
13. Schiavinotto, T., Stützle, T.: A review of metrics on permutations for search landscape analysis. Comput. Oper. Res. 34(10), 3143–3153 (2007)

Scheduling English Football Fixtures over the Holiday Period Using Hyper-heuristics

Jonathon Gibbs, Graham Kendall, and Ender Özcan

School of Computer Science, University of Nottingham,
Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{jxg16u,gxk,exo}@cs.nott.ac.uk

Abstract. One of the annual issues that has to be addressed in English football is producing a fixture schedule for the holiday periods that reduces the travel distance for the fans and players. This problem can be seen as a minimisation problem which must abide to the constraints set by the Football Association. In this study, the performance of *selection hyper-heuristics* is investigated as a solution methodology. Hyper-heuristics aim to automate the process of selecting and combining simpler heuristics to solve computational search problems. A selection hyper-heuristic stores a single candidate solution in memory and iteratively applies selected low level heuristics to improve it. The results show that the learning hyper-heuristics outperform some previously proposed approaches and solutions published by the Football Association.

Keywords: Hyper-heuristic, Metaheuristic, Local Search, Machine Learning, Sports Scheduling.

1 Introduction

The idea behind hyper-heuristics dates back to the 1960s, although the term was introduced by Dezinger et al. [1]. A hyper-heuristic is a high level problem solving methodology that performs a search over the search space generated by a set of low level heuristics [2]. One of the hyper-heuristic frameworks is concerned with automating the process of selecting and combining several simple heuristics to solve a computational search problem. A *selection hyper-heuristic* is based on this framework which operates on a set of perturbative low level heuristics performing a single point search, storing only one solution, and applying different heuristic strategies to determine which heuristic to apply, and move acceptance strategies, to determine whether the move should be accepted or not [3,4]. Bilgin et al. [5] provide a performance comparison of a variety of selection hyper-heuristics which combine different heuristic selection and move acceptance criteria. More on hyper-heuristics can be found in [6,7,8].

Scheduling, particularly sports scheduling, is a largely studied area [9]. In American sports, it is possible to schedule road trips, where a team travels to different locations, stadiums, without returning home. This is not necessary in

England as the distance between any two teams is relatively small. Therefore minimising the travelling distance over the entire season is not possible. The need for scheduling with distance minimisation in English football however does arise over the holiday period (Boxing Day and New Years Day) as fans do not wish to travel long distances during this time of the year. The travelling salesman problem is the problem of connecting each city together and returning back to the starting city [10]. This problem differs from the travelling salesman problem as it is only necessary for each city to visit only one other city.

Two approaches to the problem have been previously made by Kendall [11,12]. The first approach [11] is a two phase process. Depth first search is used to create fixtures for Boxing Day and New Years Day and then a local search algorithm aims to satisfy the remaining constraints to generate a feasible schedule, with the minimal distance possible. Although Kendall provides an improved fixture schedule from those published by the Football Association it could take up to 30 hours for a feasible solution to be created. Kendall [12] then adopts a different approach to improve the existing run time of the previous solution, a CPLEX and Simulated Annealing approach is adopted, reducing the runtime to approximately 4 minutes. In this paper, a set of hyper-heuristics combining different heuristic selection methods and acceptance criteria are applied to the fixture scheduling problem where they are evaluated and compared based on their ability to produce good quality solutions.

2 Preliminaries

Scheduling fixtures is a real world constraint optimisation problem. Due to the large size of the search space, it is impractical to use an exhaustive method, since the computation time becomes excessive [12]. Therefore, an alternative intelligent search method is needed. In this study, we investigate hyper-heuristics for solving this problem.

Hyper-heuristics can be considered as a set of general search methods that can be applied to computationally hard problems [6]. They are search and optimisation methodologies to *select* or *generate* heuristics. This study focuses on a *selection hyper-heuristic* framework as illustrated in Figure 1. In this framework, a *perturbative* low level heuristic H is selected by one of the *heuristic selection* strategies, such as, simple (uniform) random selection, which is to applied to $S_{current}$ to create a new solution S_{new}. A perturbative heuristic accepts a complete solution, perturbs it, if necessary and returns a new solution. Whether the new solution S_{new} is accepted or rejected is determined by a *move acceptance criteria*, such as, simulated annealing. This process is repeated until some termination criteria is reached. A selection hyper-heuristic will be identified as <heuristic selection method> − <move acceptance criterion> from this point onward. Determining which low level heuristic to apply is done by the heuristic selection methods.

A selection hyper-heuristic approach has been chosen here as, at each state of the problem (the distance), an operation, low level heuristic, can be chosen

```
1 generate an initial candidate solution $S_{current}$
2 while(termination criteria not satisfied){
3   *select a heuristic* (or subset of heuristics) $H$ from $\{LLH_1, ..., LLH_n\}$
4   generate a new solution $S_{new}$ by applying $H$ to $S_{current}$
5   *decide whether to accept or reject* $S_{new}$
6   if($S_{new}$ is accepted) then
7     $S_{current} = S_{new}$
8 }
9 return $S_{current}$;
```

Fig. 1. A selection hyper-heuristic framework

which performs well. It allows the combination of hill climbers, to obtain a local optimum, and also mutational heuristics, to explore the search space. [13] and [3] discuss different selection hyper-heuristic frameworks for utilising hill climbers and mutational heuristics efficiently. In this study, a generic framework is used.

Choice function (CF) is a heuristic selection method. It uses a simple learning capability based on a scoring mechanism that evaluates the low level heuristics' most recent performance and the time that has passed since the last invocation. As time progresses, it chooses more relevant low-level heuristics to apply to the solution, increasing the likelihood of finding an optimal or good quality solution. One term of the choice function allows each low level heuristic another opportunity to be called (even if it has a poor recent performance) to see if an improved solution can be generated, thus allowing the opportunity to escape local minima and not punish, or discriminate against, the low-level heuristics for previously poor solutions. The heuristic with the maximum score is selected for invocation at each step. Most of the simple hyper-heuristic components are investigated in [6]. The authors describe hyper-heuristics that combine different heuristic selection methods, including *simple random* (SR) and choice function with two move acceptance methods; *accept all moves* (AAM) and *only improving moves accepted*. The choice function−accept all moves hyper-heuristic is reported to outperform the other methods in solving a scheduling problem. Choice function as a hyper-heuristic component has also been used in [14] and shown to yield good results. Another simple acceptance method is *accept improving and equal moves* (AIEM).

Reinforcement learning (RL) heuristic selection is classified as an online learning hyper-heuristic where learning takes place while the algorithm is solving an instance of the problem [15,16]. A utility value is assigned to each low level heuristic at each iteration. If the selected heuristic improves the current solution then its score is increased by some ratio. Equally if the heuristic decreases the evaluation (assuming a minimisation problem) its score is reduced by some, different, ratio. Low level heuristics are then chosen based upon the highest score for the current state of the problem.

There are numerous reinforcement learning approaches. This paper uses a QV-Learning approach which is an extension of Q-Learning and Sarsa where the state values are taken into consideration. QV-Learning, [17], uses Equation 1 to evaluate each heuristic at each state.

$$Q(s, a) = R(d, a) + \gamma Q(s, a) \tag{1}$$

Equation 1, defines two matrices, consisting of state, where state is the current solution, and actions which relate to the low level heuristics. Q is a matrix that holds an integer value for each heuristic, a, that specifies its quality at each stage of execution. R is another matrix holding rewards or punishments based on the quality of the solution generated by a. Where s is the current state of the solution, the current solution distance, and d is the reduction or increase made to the solution. γ is determined experimentally and typically $0 \leq \gamma \leq 1$. Numerous studies have been made using reinforcement learning with feasible solutions. Burke et al. [16] provides a number of advantages when using the online learning approach, where the learning takes place as the algorithm is solving the problem. Reinforcement learning has been used in a variety of different scheduling problems [18,19,20,21], each of which report to have found optimal solutions.

Bai and Kendall [22] used *simulated annealing* (SA) [23] as a hyper-heuristic move acceptance criteria which performed well for solving a shelf allocation problem. Bilgin et al. [5] reported the success of simulated annealing with a linear cooling schedule as a move acceptance criteria for hyper-heuristics when investigating examination timetabling benchmark instances. Simulated annealing accepts all improving moves, and the worsening moves are accepted with a probability given in Equation 2.

$$e^{-\frac{\Delta f}{\Delta F(1-t/M)}}, \tag{2}$$

where Δf is the fitness change, ΔF is the (estimate of) maximum fitness change, t is the current step and M is the maximum number of steps.

Great deluge (GD) algorithm is an optimisation heuristic proposed by Dueck [24]. This method uses a threshold that decreases in time at a given rate (e.g., linearly) representing an expected solution quality. Improving moves are accepted, while worsening moves may also be accepted if it is better than the threshold. This acceptance criteria is used as a hyper-heuristic component with simple random in [25] for solving a mobile telecommunication network problem. The same hyper-heuristic is reported to perform well over a set of benchmark functions in [5]. More on selection hyper-heuristic components can be found in [6,7,3,16].

3 Hyper-Heuristics for Sports Scheduling

The English Football League is made up of four leagues known, at the time of writing, as The Barclays Premier League, Coca-Cola Championship, Coca-Cola League One and Coca-Cola League Two. Each league consists of 24 teams except for the Barclays Premier League which has 20 teams. Therefore there will be a total of 46 fixtures on both Boxing Day and New Years Day (that is 92 teams each having to play). Each league can be treated as an individual search space with the exception of teams that are classified as paired teams, these are teams that are geographically close together and have limits on how many paired teams may play in the same location on the same day. Our goal is to generate a set of fixtures which are more efficient in terms of travelling distance (i.e. lower travelling distances) than the fixtures that are released by the Football Association each June/July whilst ensuring that all constraints are respected. This section describes the problem and defines the constraints that have been put in place by the Football Association, the constraints presented here are based on [11], which can be referred to for a more complete discussion.

C1. The first constraint, home and away, requires that if a team plays at home on Boxing Day, then it must play away on New Year's Day. Equally, if a team plays at home on New Year's Day, it must play away on Boxing Day.

C2. This constraint, playing twice, requires the same teams not to play each other on both New Year's Day and Boxing Day.

C3. The third constraint, known as paired teams, requires paired teams not to play each other over the holiday period. Paired teams are teams that are, typically, geographically close to each other. This constraint has been put in place by the Football Association due to the policing requirements. We treat this as a hard constraint, however the Football Association do occasionally violate it.

C4. A pair clash occurs when two or more paired teams play at home on the same day. This constraint, pair clashes, restricts the total number of paired teams playing at home, which cannot exceed the limits specified in the Football Associations fixtures during the holidays.

C5. This constraint, London and Manchester based, restricts the number of London-based clubs that can play at home which must not exceed the limits used by the Football Association during the holidays. Similarly, there is a limit on the London based Premier League teams and Greater Manchester clubs that can play at home on the same day.

The main objective is to minimise the total travelling distance that each team is required to undertake during the holiday period; see Equation 3.

$$\min\{\sum_{x=0}^{n}\sum_{y=0}^{n} D_{x,y}X_{x,y}\} \tag{3}$$

where $D_{x,y}$ is the distance, in miles, between team x and team y and $X_{x,y}$ is 1 if team x is playing team y or 0 otherwise.

The selection hyper-heuristic framework shown in Figure 1 is used during the experiments. The performance of twelve hyper-heuristics that combine the following heuristic selection and move acceptance criteria combinations are investigated: {simple random, choice function and reinforcement learning} versus {simulated annealing, great deluge, all moves accepted, accept improving and equal moves}. Six low level heuristics are implemented which do not allow the violation of any of the five hard constraints, explained in the problem definition, throughout execution with the exception of when the initial Boxing Day or New Years Day fixtures are generated. Each time a new solution is generated a check is made using another heuristic to ensure that the swaps made do not violate constraints, if it does the move is not made. In Figure 1, low level heuristics are selected using heuristic selection at line 3 before applying the heuristic at line 4. The low level heuristics are described as follows.

LLH_1: A hill climbing heuristic that aims to improve the distance for Boxing Day. LLH_1 selects a random league and two random teams and performs a swap, if the solution yields a worse fitness value, the original solution is returned. The swap is only made for home teams.

LLH_2: A mutational heuristic that allows the solution to move away from local optima. A random league and two random teams are selected and swapped. Providing there is no violation of constraints the move is accepted regardless of its fitness value.

LLH_3: A mutational heuristic which randomly selects a league and a fixture. The home and away teams of the chosen fixture are swapped having only a minor effect on the distance but allowing a new range of home or away swaps to occur. The main purpose of LLH_3 is to conform to C4, specified in the problem definition.

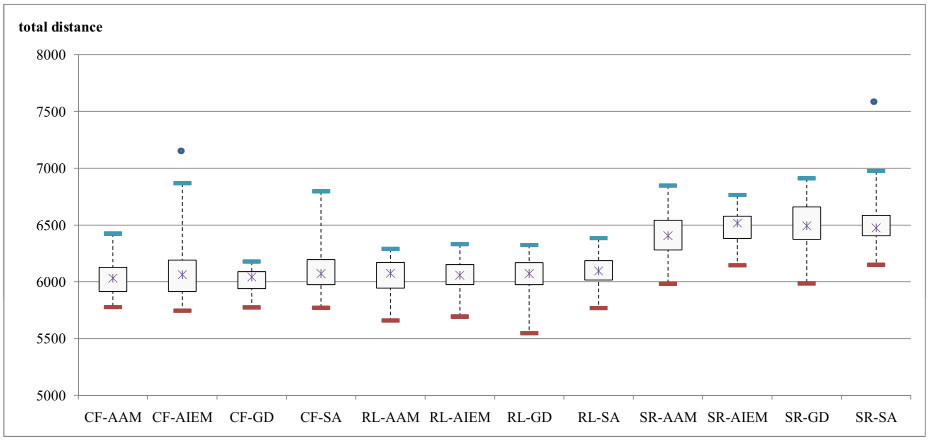
LLH_4: A hill climbing heuristic that selects a random league and two random teams and performs a swap. It uses delta evaluation, evaluating only the changed items in the solution, to evaluate the swaps and if either of them have a distance greater than 120miles (we use 120miles as this represents about 2 hours travelling time) the original solution is returned.

LLH_5: A next gradient hill climbing heuristic. A random league is selected and each home team is swapped with the one below, when ordered as a list of fixtures. Each time a swap is made, if the fitness value is improved the move is accepted. This happens for the entire league each time the heuristic is used.

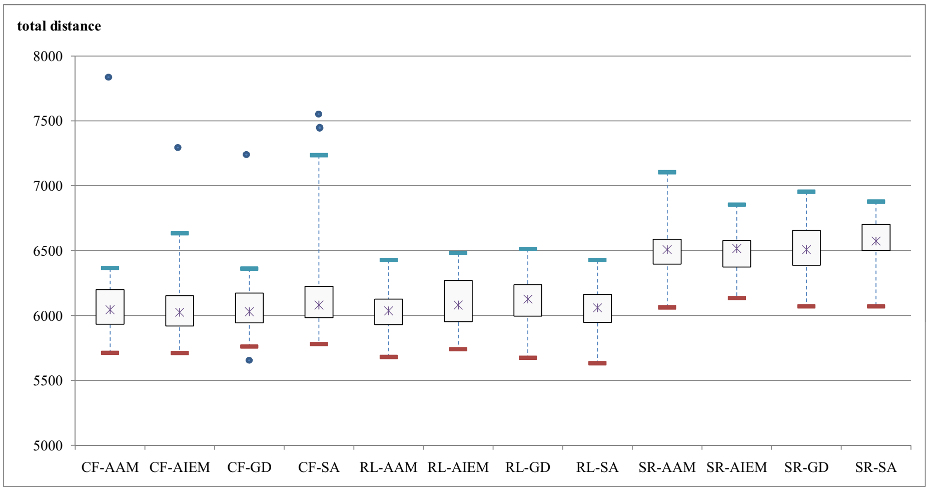
LLH_6: A hill climbing heuristic that relates to LLH_1. A random league and two random teams are selected and swapped. If the move results in a worse move the original solution is returned. The only difference between LLH_1 and LLH_6 is that LLH_6 also swaps away teams. This has been added as a separate heuristic as the performance of the two heuristics does differ.

4 Experimental Results

Two datasets are used during the experiments, the 2009-2010 season fixture set along with the 2005-2006 season from the top four divisions in England; Premier League, Coca Cola Championship and Division One and Two. The 2005-2006 season was selected to be able to compare our results to the previously proposed approaches. Season 2009-2010 was used so that the results generated here could be compared to the latest season. Distances were collected based on the



(a) 2005-2006



(b) 2009-2020

Fig. 2. Box plot of total distances found in all runs by each hyper-heuristic (displaying maximum, upper quartile, median, lower quartile and mininimum) for (a) 2005-2006 and (b) 2009-2010 problems.

postcodes of the football stadiums from greenflag (http://www.greenflag.co.uk - last accessed 4th April 2010).

An Intel Core 2 Duo, 2GHz laptop with 2.00GB memory was used to conduct the experiments. All of the selection and acceptance method combinations were executed fifty times, each of which generated a random initial fixture list; there were no pre-defined fixture lists. They were allowed to run up to the maximum number of 100,000 iterations or until 6,000 non-improving moves were made.

Table 1. The comparison of the total distances for both days against the Football Association (FA), local search approach [11], CPLEX and simulated annealing approach [12], and the best performing hyper-heuristics.

Instance	FA	Kendall [11]	Kendall [12]	Hyper-heuristics
2005-2006	10631	6917	6020	5547 (RL−GD)
2009-2010	8621	-	-	5633 (RL−SA)

Figure 2 summarises the results obtained by using different hyper-heuristics for the problem. The best results are obtained using reinforcement learning−great deluge and reinforcement learning−simulated annealing for 2005-2006 and 2009-2010 fixtures, respectively. Hyper-heuristics produce improvements over the published fixtures shown in Table 1. Indeed both approaches by Kendall in [11,12] improve the distances generated by the Football Association. However, we do not need to explicitly compare against these results due to the differences in collecting the distance data. A rough comparison does show that the hyper-heuristic approach is superior. When the average performance of hyper-heuristics are compared, choice function−great deluge and reinforcement learning−accept all moves are better than the others for 2005-2006 and 2009-2010 fixtures, respectively. Wilcoxon test shows that both of these hyper-heuristics perform significantly better than the simple random heuristic selection based hyper-heuristics within a confidence interval of 95%. Almost the rest of the learning hyper-heuristics deliver a similar performance.

Simple random requires a much shorter computational time (4 seconds on average) than choice function (12 seconds on average) and reinforcement learning (52 seconds on average) for solving a given problem instance. However, the learning approaches, choice function and reinforcement learning, do produce improved results. It can be observed that, on average, as the computation time increases, typically with the time spent for learning, the results improve. The results illustrate that there is a trade off between time and quality of solutions. Simple random as a heuristic selection runs quicker than any of the other heuristics yet produces the worst results. Choice function improves the results of simple random, by approximately 5%, yet increases the run time by approximately three times as much, on average. Finally, reinforcement learning as a heuristic selection method provides the highest ranked results for each of the criteria. It dramatically improves the solutions generated by simple random but the computation time is much higher.

5 Conclusion

Each combination of heuristic selection and move acceptance within a selection hyper-heuristic framework improved the results and provided a good quality solution to the sports scheduling problem in reasonable time whilst conforming to the constraints specified. While applying the 2005-2006 dataset, the reinforcement learning−great deluge hyper-heuristic made savings to the Boxing Day and New Years Day fixtures of 44.19% and 50.28%, respectively, whilst even comparing results from the current season, dataset 2009-2010, it was possible to improve the Boxing Day fixtures by 41.42% and New Years Day fixtures by 28.29% using the reinforcement learning−simulated annealing hyper-heuristic.

A trade off must be made between computation time and quality of solutions. For each of the datasets, reinforcement learning found the best solutions but had a runtime of at least two times than that of choice function and over ten times the amount compared to simple random. Reinforcement learning produced the most consistent set of solutions on average where each move acceptance criterion generated similar results and, with the exception of great deluge, all executed in similar times. Therefore we conclude, that reinforcement learning is the best heuristic selection component to be used within hyper-heuristics for solving this sports scheduling problem.

References

1. Denzinger, J., Fuchs, M., Fuchs, M.: High performance atp systems by combining several ai methods. In: Proceedings of the 4th Asia-Pacific Conference on SEAL, IJCAI, pp. 102–107 (1997)
2. Özcan, E., Bykov, Y., Birben, M., Burke, E.K.: Timetabling using late acceptance hyper-heuristics. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 997–1004. IEEE Press, Los Alamitos (2009)
3. Özcan, E., Bilgin, B., Korkmaz, E.: A comprehensive analysis of hyper-heuristics. In: Intelligent Data Analysis, pp. 3–23 (2008)
4. Özcan, E., Mısır, M., Ochoa, G., Burke, E.K.: A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 1(1), 39–59 (2010)
5. Bilgin, B., Özcan, E., Korkmaz, E.E.: An experimental study on hyper-heuristics and exam timetabling. In: Proceedings of the 6th Practice and Theory of Automated Timetabling (PATAT 2006). LNCS, vol. 3867, pp. 394–412. Springer, Heidelberg (2006)
6. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 176–190. Springer, Heidelberg (2001)
7. Burke, E.K., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 457–474. Kluwer, Dordrecht (2003)
8. Ross, P.: Hyper-heuristics. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, pp. 529–556. Springer, Heidelberg (2005)
9. Kendall, G., Knust, S., Ribeiro, C., Urrutia, S.: Scheduling in sports: An annotated bibliography. Computers & Operations Research 37, 1–19 (2010)
10. Applegate, D.L., Bixby, R.E., Chvatal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2007)

11. Kendall, G.: Scheduling english football fixtures over holiday periods. Journal of the Operational Research Society 59(6), 743–755 (2008)
12. Kendall, G.: Hybridising cplex with simulated annealing to minimise travel distances for english football fixtures (2009) (in review)
13. Özcan, E., Bilgin, B., Korkmaz, E.E.: Hill climbers and mutational heuristics in hyperheuristics. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 202–211. Springer, Heidelberg (2006)
14. Kendall, G., Cowling, P., Soubeiga, E.: Choice function and random hyperheuristics. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning, SEAL, pp. 667–671 (2002)
15. Nareyek, A.: Choosing search heuristics by non-stationary reinforcement learning. In: Resende, M.G.C., de Sousa, J.P. (eds.) Metaheuristics: Computer Decision-Making, pp. 523–544. Kluwer, Dordrecht (2003)
16. Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.: A classification of hyper-heuristic approaches. In: Handbook of Metaheuristics. Springer, Heidelberg (to appear, 2010)
17. Wiering, M.: Qv(lambda)-learning: A new on-policy reinforcement learning algorithm. In: Proceedings of the 7th European Workshop on Reinforcement Learning (2005)
18. Aydin, M., Öztemel, E.: Dynamic job-shop scheduling using reinforcement learning agents. In: Robotics and Autonomous Systems, vol. 33, pp. 39–59. Elsevier, Amsterdam (2000)
19. Luiz, A., Ribeiro, C., Costa, A., Bianchi, R.: Heuristic reinforcement learning applied to robocup simulation agents. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 220–227. Springer, Heidelberg (2008)
20. Wang, Y., Usher, J.: Application of reinforcement learning for agent-based production scheduling. In: Engineering Applications of Artificial Intelligence, vol. 18, pp. 73–82 (2005)
21. Zhang, W., Dietterich, T.: A reinforcement learning approach to job-shop scheduling. In: Proceedings of the 14th international joint conference on Artificial intelligence, vol. 1, pp. 1114–1120 (1995)
22. Bai, R., Kendall, G.: An investigation of automated planograms using a simulated annealing based hyper-heuristics. In: Ibaraki, T., Nonobe, K., Yagiura, M. (eds.) Metaheuristics: Progress as Real Problem Solver. Operations Research/Computer Science Interface Serices, vol. 32, pp. 87–108. Springer, Heidelberg (2005)
23. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
24. Dueck, G.: New optimization heuristics: The great deluge algorithm and the record-to record travel. Journal of Computational Physics 104, 86–92 (1993)
25. Kendall, G., Mohamad, M.: Channel assignment optimisation using a hyper-heuristic. In: Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems (CIS 2004), Singapore, December 1-3, pp. 790–795 (2004)

Graph Clustering Based Model Building

David Iclănzan[1,2] and Dumitru Dumitrescu[2]

[1] Department of Electrical Engineering,
Sapientia Hungarian University of Transylvania,
Şoseaua Sighişoarei 1C, 547367, Corunca, Romania
david.iclanzan@gmail.com

[2] Department of Computer Science, Babe s-Bolyai University,
Kogălniceanu no. 1, 400084, Cluj-Napoca, Romania
ddumitr@cs.ubbcluj.ro

Abstract. Probabilistic models of high-order statistics, capable of expressing complex variable interactions, have been successfully applied by estimation of distribution algorithms (EDAs) to render hard problems tractable. Unfortunately, the dependence structure induction stage in these methods imposes a high computational cost that often dominates the overall complexity of the whole search process.

In this paper, a new unsupervised model induction strategy built upon a maximum flow graph clustering technique is presented. The new approach offers a model evaluation free, fast, scalable, easily parallelizable method, capable of complex dependence structure induction. The method can be used to infer different classes of probabilistic models.

1 Introduction

Estimation of Distribution Algorithms (EDAs) extend the classical framework of Evolutionary Algorithms (EAs) with a novel approach consisting in learning and exploiting information from selected individuals. Global statistical information is extracted from promising solutions and used to infer a probabilistic model. New solutions are then sampled from the probability distribution model in order to generate the next population.

The search for an appropriate model in EDAs capable of modeling higher order dependencies, requires many model evaluations with regard to the population. Given the implied population sizes as the dimension of the problems increases, the computational cost of model building may quickly exceed economical practicality. Recent benchmarking and profiling results showed that easily more than 90% of EDAs running time may be spent in the model building phase [1].

Recent efforts have aimed making higher order EDAs computationally less expensive. Enhancements and modifications of the original methods considered parallelization [2,3] and hybridization with local search methods [4], the usage of iterative [5] and sporadic model building [6] or incorporation of initial knowledge [7]. More direct approaches aim to reduce the complexity of model building by restricting the search over a reduced set of variables in each epoch [1].

Another line of research concentrate on the usage of global statistics extracted from the data to reduce or bypass the number of model accuracy evaluations. The improved Estimation of Dependency Networks Algorithm [8] uses a multivariate dependency network approximation by considering only bivariate statistics. In another work [9], the $O(n^3)$ model building of the Extended Compact Genetic Algorithm (eCGA) is successfully replaced by a variable correlation guided search of linear complexity. Some methods completely avoid the goodness-of-fit evaluations of the models with regard to the data, by clustering a pairwise variable interaction matrix. The Dependency Structure Matrix Genetic Algorithm (DSMGA) [10] and its extension to hierarchical problems DSMGA++ [11] both use dependency structure matrix clustering techniques for linkage learning. These methods still employ a costly search process to find a clustering setting that minimizes a metric based on the Minimum Description Length (MDL) principle. In [12] the authors use the affinity clustering of the sampled mutual information matrix to obtain a marginal product model factorization which is not able to represent overlapping linkages but may suffice for many applications.

In this paper we further explore the confluence between clustering algorithms and EDAs, where graph clustering algorithms are applied to pairwise interaction statistic matrices to reveal dependency structures. We term this class of methods as Graph Clustering assisted EDAs (GCEDAs). We are especially interested in finding efficient clustering algorithms allowing the induction of various probabilistic model classes.

Here, we focus on the class of flow-based graph clustering algorithms as they are know to be relatively fast and simple while some variants still being able to handle overlapping clusters. From the variants built upon this idea we have chosen to use the Markov Clustering Algorithm (MCL) [13], as it has a simple and elegant formulation based on just two operators, proved effectiveness in clustering real-world biological data [14], good documentation and available source code under GNU General Public License[1].

The main technical contribution of this paper lays in showing that given the pairwise interaction map of the variables, a simple *unsupervised* graph clustering algorithm is able to assist qualitative linkage learning by allowing the inference of different probabilistic models like Bayesian Networks, overlapping linkage models and marginal product models. Hard optimization problems, characterized by non-separable, high-order of interactions among the variables, with deceptive subproblems are solved.

The following section provides some preliminaries and a discussion about how an EDA can use the result of graph clustering for probabilistic model building. Section 3 presents an EDA, which implements the ideas and particular clustering techniques discussed in Section 2. Section 4 contains the description of our experimental setup, the results and a discussion of our findings related to the MCL assisted EDA are given in section 5. Finally, Section 6 discusses results and implications of this work and outlines some future work.

[1] http://www.micans.org/mcl/

2 Preliminaries

Let $G = (V, E)$ denote the input graph for the graph clustering algorithm, with V and E denoting the node set and edge set respectively. Let A be the $|V||V|$ adjacency matrix, with $A(i, j)$ denoting the weight of the edge between the vertex v_i and the vertex v_j . In our setting, this weight represent the strength of the pairwise interaction between variables, as extracted from the data available for model building. In this paper, the pairwise dependency is quantified by sampled mutual information.

2.1 Graph Clustering Paradigm, Stochastic Matrices and Flows

Maximum flow clustering algorithms rely on the following core idea: by simulating a special flow within a graph, which promotes flow where the current is strong, and reduces flow where the current is weak will reveal the cluster structure within the graph, as the flow across borders between different groups diminish with time, while it increases within the group.

Simulation of flow through a graph is easily done by transforming the adjacency matrix into a column-stochastic square matrix, where each column sums to 1. This matrix, which we denote by M, can be interpreted as the matrix of the transition probabilities of a random walk (or a Markov chain) defined on the graph, where $M(j, i)$ represents the probability (stochastic flow) of a transition from vertex v_i to v_j.

The flow matrix M is obtained by normalizing the columns of the adjacency matrix to sum up to 1. Flow expansion can be simulated by computing powers of the flow (Markov) matrix M.

2.2 Markov Clustering Algorithm

The MCL algorithm [13] is a fast and scalable unsupervised graph clustering algorithm, based on simulation of stochastic flow in graphs. It offers several advantages, like a simple, elegant mathematical formulation, robustness to topological noise [14], support for easy paralellization and adaptation via a simple parameter enables the obtaining of clusters of different granularities.

MCL iteratively simulates random walks within a graph by applying two operators called expansion and inflation, until convergence occurs. At the end of each inflation step a pruning step is also performed, in order to reduce the computational complexity by keeping M sparse.

Intuitively, the MCL process may be regarded as alternative expansion and contraction of the flow in the graph. The expansion step is responsible in spreading the flow out of a vertex to potentially new vertices and with the strengthening of the flow to those vertices which are reachable by multiple paths. This has the effect of enhancing within-cluster flows as there are more paths between the nodes belonging to the same cluster. The inflation operator is responsible for both strengthening intra-cluster flow and weakening inter-cluster flow of current and by this, introducing a non-linearity in the distribution of the flows. At the beginning the flow distribution is relatively smooth and uniform, but with each iteration it becomes more and more peaked. In the end, all the nodes within a tightly-linked group of nodes will start to flow towards one node within the group, forming star sub-graphs associated with the MCL limits.

The idealized Markov Cluster process, consisting just from the expansion and inflation operators is known to converge quadratically in the neighborhood of so called doubly idempotent matrices [13]. In practice, the numbers of epochs until convergence is reported to be nearly always far below 100.

2.3 Interpretation of MCL Clustering as Dependency Models

MCL iterants M_t are generally diagonally positive semi-definite matrices. Using the property that minors of a diagonally positive semi-definite matrix are non-negative, in [15] it is shown that M_t-s have a structural property which associates a directed acyclic graph (DAG) with each of them. These DAGs generalize the star graphs associated with the MCL limits.

We present several approaches on how the information from MCL iterants can be conveyed in dependency models able to represent and exploit linkages.

In the *first* approach, the DAGs represented by M_t-s are directly used for defining the structure for a Bayesian network, with the edges representing the conditional dependencies between variables. Then, the parameters, which consist of the conditional probabilities of each variable given the variables that this variable depends on, are extracted from the data in the same way as in BOA [16] or EBNA [17]. The obtained Bayesian network will encode the joint probability distribution of the variables and can be used to sample the next generation. This approach presents two small impediments. First, one has to decide from which M_t to construct and use the Bayesian network, thus a few model evaluations against the data still have to be computed. The second issue relates to the rare occasions where a MCL iterant contains cycles. Before performing the parameter extraction, these cycles must be detected and eliminated.

In a *second* approach, DAGs are interpreted as clusters by taking as cores all end nodes (sinks) from the DAG, and by attaching to each core all the nodes that reach it with a flow amount greater than a threshold. This procedure may result in clusters containing overlap. The extracted linkages can be used to perform building block wise crossover like in DSMGA [10] or DSMGA++ [11] or they can be used to build overlapping linkage model based probability distributions.

The *third* approach is the cheapest one, as it deals only with the last iterant, when the stochastic flow matrix M is completely converged. Here, the nodes have found one "attractor" node to which all of their flow is directed, corresponding to only one non-zero entry per column in M. Nodes sharing the same "attractor" node are grouped in clusters. This approach is suitable for modeling non-overlapping building blocks, by building marginal product models as in eCGA. [18].

Excepting the first approach, the dependency structure inferring is completely autonomous, as its does not need to check the model fit with regard to the data.

3 MCL Assisted EDA

Wishing to present an EDA with unsupervised model building, capable of modeling complicated variable interactions, we employ the second interpretation of the MCL iterants to obtain a overlapping linkage model based probabilistic model. We name this algorithm Markov Clustering EDA (MCEDA). The details of the algorithm are presented in the followings.

In this paper, the degree of pairwise dependency between variables is calculated using sampled mutual information between two variables and record into an adjacency matrix A, which will be the input of the graph clustering algorithm.

The transformation of A in a stochastic Markov matrix is handled by the MCL algorithm by normalization of the columns to sum up to 1.

3.1 The Overlapping Linkage Model (OLM)

In MCEDA, the multivariate variable interactions are modeled with the use of overlapping linkage models (OLMs), which closely resembles the marginal product model adopted by the eCGA [18]. The difference is that OLM models subsets of variables jointly as clusters, allowing overlaps, in contrast with partitions, which always divides the variables in collectively exhaustive and mutually exclusive blocks. The clusters can naturally represent building blocks, providing a direct linkage map of the variables, thus we will use this terms interchangeably in the context of OLMs. Clusters together with the marginal distributions over them form the OLMs.

3.2 Dependency Structure Building and Sampling

The clusters that form the basis of the OLMs are extracted from the iterants M_t of the MCL algorithm.

For each node a potential building block is formed, by grouping together all nodes that reach it with a flow amount greater than a threshold F_{min}. Basic clusters of size 1 are only allowed, if the described single position is not contained in any other cluster. After all iterants have been processed, the procedure returns the unique entries of the potential building block list. This sorting must be performed, as the same cluster may be detected several times from different iterants, or even from the same M_t in the rare cases when it contains cycles.

The building block extraction is depicted in Function `ExtractBBs`.

Please note that a practical implementation does not have to store all the iterants for a final batch processing. It is described in this way to keep the presentation clean and simple. A clever building block extraction works in interplay with the MCL, processing each iterant right after it was computed, retaining the unique building blocks in the same fashion.

After the building blocks are determined, their probability distribution is estimated by simply counting the frequencies in the data.

As the performance of the method did not seem to be very sensible on the setting of F_{min}, we use a fixed value of $10e-4$.

Function `ExtractBBs(`*Mlist*`)` returns BBlist

```
1 $BBlist \leftarrow \emptyset$;
2 //each iterant from the MCL algorithm is processed
3 foreach $M_t$ from $Mlist$ do
4     //for all nodes find significant incoming flows
5     for $i \leftarrow 1$ to $size(M_t)$ do
6         $pBB \leftarrow find(M_t(i,:) > F_{min}(i))$;
7         if $length(pBB) > 1$ then
8             $BBlist \leftarrow BBlist \bigcup pBB$;
9 $BBlist \leftarrow unique(BBlist)$;
```

Algorithm 2: The Markov Clustering EDA

```
1 $pop \leftarrow RandomInit()$;
2 repeat
3     $ps \leftarrow Selection(pop)$; //select promising solutions
4     $\{ps \leftarrow ReduceEntropy(ps)\}$; //optionally reduce entropy by LS
5     $A \leftarrow MutualInformation(ps)$; //extract global statistics
6     $Mlist \leftarrow MCL(A)$; //apply graph clustering
7     $BBlist \leftarrow ExtractBBs(Mlist)$; //extract dependency structure
8     $freq \leftarrow FrequencyCount(BBlist, ps)$; //compute marginal probabilities
9     $olm \leftarrow BuildMPM(BBlist, freq)$; //combine results into a OLM
10    $pop \leftarrow Sample(olm)$; //generate a new population using the model
11 until convergence criteria is met ;
```

The building blocks, together with the frequencies obtained from the data form the OLM model. This probabilistic model is sampled by enumerating the building blocks in a random order, and choosing a configuration according to the registered probabilities.

3.3 The Markov Clustering EDA

Starting from a random population, the MCEDA applies the process of evaluation, selection, MCL assisted OLM model-building and sampling until a halting criterion is met, which is usually a combination of several criteria like computational and time resources spent, amount of progress between epochs, the energy of best solution found so far etc.

As the method relies only on pairwise information statistics, the signal of this measure must be relatively strong. A good entropy reduction of the samples used to generate the statistics can be achieved by performing a local search, and/or using a big population size with a high selection pressure.

Algorithm 2 summarizes the structure and workings of the method.

4 Experiments

In this paper the class of additively decomposable functions (ADFs) with deceptive trap subproblems is considered, a test bed which is widely used in the literature as benchmarking problems[16,19,20].

4.1 Test Functions

The *concatenated trap-5* [16] is an ADF based on unitation (number of ones from a binary string) measures, exhibiting a single global optimum in the string formed exclusively from ones.

The input string is partitioned into disjoint shuffled groups of five bits each. A 5-bit trap function is applied to each of the groups and the fitness of the individual is the sum of the contributions of each 5-bit group.

Each subproblem of five variables has two competing schemata which are maximally distant: (0, 0, 0, 0, 0) and (1, 1, 1, 1, 1). The fitness gradient leads towards the string formed by zeros, thus low order statistics, taking into account less than 5 variable statistics, may lead away from the optimal value, deceiving the search.

Non-separability can be introduced by applying a fixed length, circular overlapping scheme between the trap-5 functions [19]. For example, for a problem with 3 subproblems and overlap length $l = 2$, the fitness is given by $trap5(y1y2y3y4y5) + trap5(y4y5y6y7y8) + trap5(y7y8y9y1y2)$, where yi is a random permutation of the variables xi, meant to break the tight linkage. Every building block shares $2l$ variables, l with each of its two neighbor.

4.2 Numerical Results

Experiments are performed with the simple concatenated trap-5 function without overlap (denoted by ctf5o0), with overlap 1 (ctf5o1) and overlap 2 (ctf5o2).

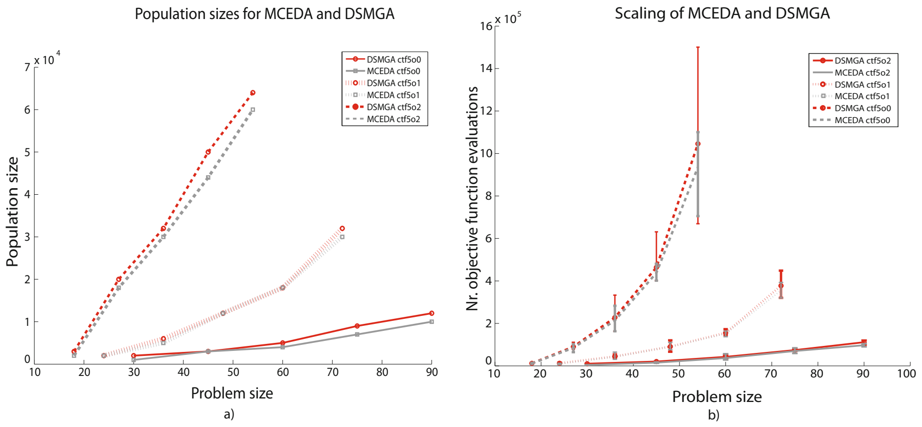


Fig. 1. The scaling of MCEDA and DSMGA on the test problems

In order to test the scalability, for each ADF the number of subproblems k is scaled from 6 to 18 by increments of 3, resulting in various problem sizes up to 90 variables.

The MCEDA performance is compared with the DSMGA having the following parameterization: tournament selection of size 8, crossover probability 1 and no mutation. Population sizes for both algorithms were determined by the bisection method, requiring the methods to converge to the global optima in 10 out of 10 independent runs. The obtained population sizes are depicted in Figure 1 a).

In these experiments, the MCEDA did not use local-search for entropy reduction. The selection operator chooses the winners based on truncation selection, where the best 10% of the population is promoted.

Figure 1 b) presents the scaling of the methods for the different problem types and sizes. The results show a similar scaling of the two methods. MCEDA uses slightly fewer objective function evaluations and works with smaller population sizes than the DSMGA. The performance difference is most likely explained by the following two factors:

– DSMGA uses a crisp value when dealing with variable interactions. The mutual information matrix is transformed in a binary matrix according to a threshold, where two variables are considered fully interacting or completely independent. In contrast, the MCEDA works directly with the normalized mutual information values, which can describe more nuanced, weighted levels or interactions, facilitating the earlier discovery of better models.
– The MCEDA has a better diversity maintenance mechanism as a higher number of building-blocks are extracted due to the usage of early iterants of the MCL process. Furthermore, the method samples according to the exactly observed frequencies in the data. DSMGA may confront the hitch-hiking phenomena [21] where some low fitness alleles are promoted together with high-quality building-blocks in above average individuals and the right crossover must be performed to eliminate them.

The MCL graph clustering is much faster than the MDL based clustering used by DSMGA, having a worst case complexity $O(nk^2)$ [13], where k is the pruning factor (at most how many non-zero entries will be in a row of the stochastic matrix - a very small number in practice, $k << n$). Clustering of 5000 nodes by the MCL takes only a few seconds, compared with minutes, in the case of DSMGA model-building.

5 Conclusions and Future Work

The paper proposes a new model inferring method for EDAs, where unsupervised graph clustering algorithms are applied to global statistics extracted from the population data, in order to reveal dependency structures that facilitates the induction of various probabilistic model types. As there is no explicit search for models, with computationally expensive fit-to-data evaluations, this approach has the potential to alienate the model building cost bottleneck in EDAs, enabling them to scale up to truly large problem sizes. Graph clustering algorithms are usually highly paralellizable, some are able to work in a decentralized, distributed fashion and they are known to be able to cluster graphs containing millions of nodes.

Test results show that the method is able to efficiently solve well known benchmark problems, exhibiting deceptiveness and non-separable building blocks. The strength of the algorithm is also its weakness: as it relies solely on pairwise interaction information, this information needs to be of good quality.

The great advantage of the presented method is that it allows the induction of various probabilistic model classes.

Future work will advance the study and exploration of the GCEDA framework by interpreting clusterings as directed acyclic graphs, in order to build Bayesian networks. Other research will focus in the development of highly parallel model building and large scale optimization.

Acknowledgments

This work is supported by the Grant IDEI 508 "New Computational Paradigms for Dynamic Complex Problems" funded by the Ministry of Education, Research and Innovation, Romania and by the Sapientia Institute for Research Programs (KPI).

References

1. Duque, T.S., Goldberg, D.E., Sastry, K.: Enhancing the efficiency of the ECGA. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 165–174. Springer, Heidelberg (2008)
2. Sastry, K., Goldberg, D.E., Llora, X.: Towards billion-bit optimization via a parallel estimation of distribution algorithm. In: GECCO 2007: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 577–584. ACM, New York (2007)
3. Ocenásek, J., Schwarz, J.: The parallel bayesian optimization algorithm. In: Proceedings of the European Symposium on Computational Inteligence, pp. 61–67. Springer, Heidelberg (2000)
4. Pelikan, M., Hartmann, A.K., Sastry, K.: Hierarchical BOA, cluster exact approximation, and ising spin glasses. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 122–131. Springer, Heidelberg (2006)
5. Pelikan, M., Sastry, K., Goldberg, D.E.: iBOA: the incremental bayesian optimization algorithm. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 455–462. ACM, New York (2008)
6. Pelikan, M., Sastry, K., Goldberg, D.: Sporadic model building for efficiency enhancement of the hierarchical BOA. Genetic Programming and Evolvable Machines 9(1), 53–84 (2008)
7. Baluja, S.: Incorporating a priori knowledge in probabilistic-model based optimization. Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, pp. 205–219 (2006)

8. Gámez, J.A., Mateo, J.L., Puerta, J.M.: Improved EDNA(estimation of dependency networks algorithm) using combining function with bivariate probability distributions. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation GECCO 2008, pp. 407–414. ACM, New York (2008)
9. Iclanzan, D., Dumitrescu, D., Hirsbrunner, B.: Correlation guided model building. In: GECCO 2009: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, July 8-12, pp. 421–428. ACM, New York (2009)
10. Yu, T.L., Goldberg, D.E., Yassine, A., Chen, Y.P.: Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 1620–1621. Springer, Heidelberg (2003)
11. Yu, T.L., Goldberg, D.E.: Conquering hierarchical difficulty by explicit chunking: substructural chromosome compression. In: GECCO 2006, pp. 1385–1392. ACM Press, NY (2006)
12. Santana, R., Larrañaga, P., Lozano, J.A.: Estimation of distribution algorithms with affinity propagation methods. Technical Report EHU-KZAA-IK-1/08, Department of Computer Science and Artificial Intelligence, University of the Basque Country (January 2008)
13. van Dongen, S.: Graph Clustering by Flow Simulation. PhD thesis, U. of Utrecht (2000)
14. Brohée, S., van Helden, J.: Evaluation of clustering algorithms for protein-protein interaction networks. BMC Bioinformatics 7, 488 (2006)
15. van Dongen, S.S.: A stochastic uncoupling process for graphs. Technical Report INS-R0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam (2000)
16. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Wu, A., et al. (eds.) GECCO 1999, Orlando, FL, July 13-17, vol. I, pp. 525–532. Morgan Kaufmann Publishers, San Fransisco (1999)
17. Etxeberria, R., Larranaga, P.: Global optimization using Bayesian networks. In: Proceedings of the Second Symposium on Artificial Intelligence (CIMAF 1999), pp. 151–173 (1999)
18. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report no. 99010, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign (February 4, 1999)
19. Yu, T.L., Sastry, K., Goldberg, D.E.: Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1217–1224. ACM, New York (2005)
20. Correa, E., Shapiro, J.: Model complexity vs. performance in the bayesian optimization algorithm. In: Parallel Problem Solving from Nature-PPSN IX, pp. 998–1007 (2006)
21. Mitchell, M., Holland, J.H.: When will a genetic algorithm outperform hill climbing? In: Forrest, S. (ed.) Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo, CA, USA, p. 647. Morgan Kaufmann, San Francisco (1993)

How to Choose Solutions for Local Search in Multiobjective Combinatorial Memetic Algorithms

Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Yoshihiko Wakamatsu, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
{hisaoi@,hitotsu@ci.,wakamatsu@ci.,nojima@}cs.osakafu-u.ac.jp

Abstract. This paper demonstrates that the performance of multiobjective memetic algorithms (MOMAs) for combinatorial optimization strongly depends on the choice of solutions to which local search is applied. We first examine the effect of the tournament size to choose good solutions for local search on the performance of MOMAs. Next we examine the effectiveness of an idea of applying local search only to non-dominated solutions in the offspring population. We show that this idea has almost the same effect as the use of a large tournament size because both of them lead to high selection pressures. Then we examine different configurations of genetic operators and local search in MOMAs. For example, we examine the use of genetic operators after local search. In this case, improved solutions by local search are used as parents for recombination while local search is applied to the current population after generation update.

Keywords: Multiobjective genetic local search (MOGLS), evolutionary multiobjective optimization (EMO), hybrid algorithms, memetic algorithms, multiobjective combinatorial optimization.

1 Introduction

Since the mid-1990s [3], [4], local search has often been combined with evolutionary multiobjective optimization (EMO) algorithms to improve their search ability in the literature [11]. Hybrid EMO algorithms with local search were first proposed under the name of multiobjective genetic local search (MOGLS [3], [4], [7], [8]). Such a hybrid algorithm is also referred to as a multiobjective memetic algorithm (MOMA [6], [9]-[11]). In early studies [3], [4], [7]-[10], MOMAs were mainly applied to multiobjective combinatorial optimization problems. Recently local search has been also combined with EMO algorithms for multiobjective continuous optimization [13].

It is well-known that hybrid evolutionary algorithms with local search have high search ability for single-objective combinatorial optimization problems. They are often called genetic local search (GLS) or memetic algorithms (MAs [14]). A number of issues for designing high-performance MAs have been discussed for single-objective optimization [12], [16], [17] and multiobjective optimization [5]. An important issue is the balance between local search and genetic search especially in

MOMAs [6]. When this balance is not appropriately specified, the performance of EMO algorithms is often severely degraded by the hybridization with local search.

Another important issue in the design of high-performance MOMAs is the choice of solutions to which local search is applied. This issue has not been discussed in detail in MOMAs for multiobjective combinatorial optimization in the literature. This is because the performance of EMO algorithms is usually improved by simply applying local search to good offspring as long as the balance between local search and genetic search is appropriate. In this paper, we examine a number of strategies for choosing local search solutions in MOMAs. For this purpose, we use a simple MOMA called S-MOGLS [2], which is a hybrid algorithm of NSGA-II [1] with local search. Its generation update mechanism is illustrated in Fig. 1. First genetic search (i.e., selection, crossover and mutation) is applied to the current population in the same manner as NSGA-II. Next local search is applied to the offspring population. Then the next population is constructed by choosing good solutions from the current, offspring and improved populations in the same manner as NSGA-II. Of course, similar MOMAs can be designed using other EMO algorithms instead of NSGA-II.

In this paper, we examine the following strategies to choose local search solutions:

(i) Selection of local search solutions from the offspring population as in Fig. 1.
(ii) Selection from non-dominated solutions in the offspring population.
(iii) Selection from a merged population of the current and offspring populations.
(vi) Selection from the current population. In this case, local search is used before genetic search as shown in Fig. 2.

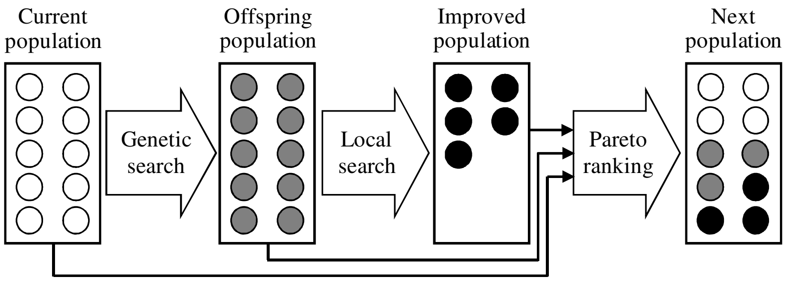


Fig. 1. Our standard MOMA with the CP-GS-LS structure (S-MOGLS [2])

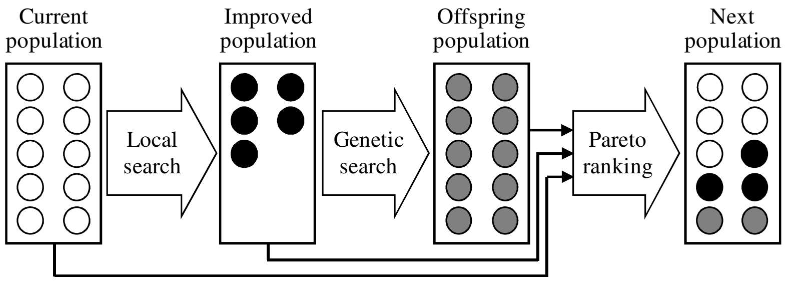


Fig. 2. Our MOMA with the CP-LS-GS structure

2 Our Multiobjective Memetic Algorithm

Let us consider the following *k*-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ ...,\ f_k(\mathbf{x}))\ . \tag{1}$$

We explain our MOMA and its variants using this multiobjective problem.

Our MOMA in Fig. 1 is a simple hybrid algorithm of NSGA-II with local search. We denote the structure of MOMAs in Fig. 1 as "CP-GS-LS" since local search (LS) is used after genetic search (GS) is applied to the current population (CP). The outline of our MOMA with the CP-GS-LS structure can be written as follows:

[MOMA with the CP-GS-LS structure]

```
Step 1: $P$ = Initialize($P$)
Step 2: While the stopping condition is not satisfied, do
Step 3:    $P'$ = Genetic Search($P$)
Step 4:    $P''$ = Local Search($P'$)
Step 5:    $P$ = Generation Update($P \cup P' \cup P''$)
Step 6: End while
Step 7: Return Non-dominated($P$)
```

First an initial population P with N_{pop} solutions is randomly generated in Step 1 where N_{pop} is the population size. Then Steps 3-5 are iterated until a prespecified stopping condition is satisfied. Step 3 is exactly the same as the genetic search of NSGA-II. An offspring population P' is generated. Step 5 is conceptually the same as the generation update mechanism of NSGA-II. The best N_{pop} solutions are selected as the next population P from the merged population $P \cup P' \cup P''$ in Step 5 using Pareto ranking and crowding distance.

In Step 4, we use the following weighted sum fitness function for local search:

$$f(\mathbf{x}) = \lambda_1 f_1(\mathbf{x}) + \lambda_2 f_2(\mathbf{x}) + \cdots + \lambda_k f_k(\mathbf{x})\,, \tag{2}$$

where $\boldsymbol{\lambda}=(\lambda_1, \lambda_2, ..., \lambda_k)$ is a weight vector. Of course we can use other functions. We use a set of uniformly distributed weight vectors satisfying the following conditions:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k = d \quad \text{and} \quad \lambda_i \in \{0, 1, ..., d\} \text{ for } i = 1,2,...,k\ . \tag{3}$$

The same weight vector generation mechanism was used in [15] and [18]. We specify d in (3) as $d = 100$ to generate 101 weight vectors for two-objective problems.

In Step 4, first a weight vector is randomly drawn from the weight vector set. Then a local search solution is selected from the offspring population P' using tournament selection with replacement. Various values of tournament size are examined in our computational experiments. Each solution in P' is evaluated by the weighted sum fitness function in (2) with the current weight vector. Local search is applied to the chosen solution. The weighted sum fitness function in (2) with the current weight vector is used to compare the current solution and its neighbors in local search.

In local search, a neighbor is randomly generated from the current solution. When a better neighbor is found, the current solution is replaced with it. That is, we use the first move strategy where local search accepts the first improved neighbor rather than the best move strategy. As a termination condition of local search, we use the total number of examined neighbors (say, $N_{\rm LS}$ neighbors) in a series of local search from the local search solution (i.e., starting solution) chosen from the offspring population.

The number of solutions to which local search is applied in each generation can be specified using the local search application probability $P_{\rm LS}$ as $P_{\rm LS}N_{\rm pop}$. Since $N_{\rm LS}$ neighbors are examined in a series of local search from each local search solution, the total number of examined solutions by local search in each generation can be calculated as $P_{\rm LS}N_{\rm pop}N_{\rm LS}$ while $N_{\rm pop}$ solutions are examined by genetic search.

3 Variants of Our Multiobjective Memetic Algorithm

As shown in Fig. 1, local search is usually applied to the offspring population in MOMAs. This is, however, not necessarily the best structure of MOMAs. In general, the offspring population may include many poor solutions due to the random nature of crossover and mutation. In Fig. 3, we show an example of the current population and its offspring population in a single run of NSGA-II on the two-objective 500-item 0/1 knapsack problem [19]. Conditions of this experiment will be shown in Section 4.

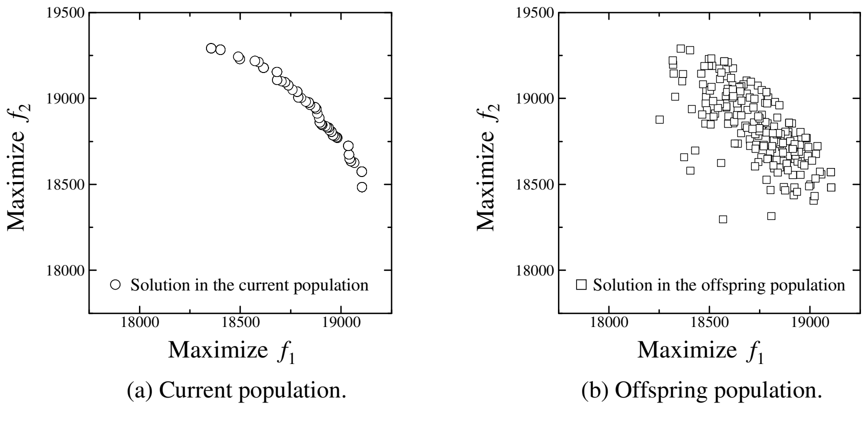


Fig. 3. Current and offspring populations at the 200th generation in a single run of NSGA-II on the two-objective 500-item 0/1 knapsack problem (see Section 4 for parameter values)

The application of local search to poor solutions is often the waste of time. In our MOMA, we can choose a good solution from the offspring population using tournament selection with a large tournament size. We can also use a strategy to apply local search only to non-dominated solutions in the offspring population. This idea is written as follows in our MOMA in the previous session.

```
Step 4:   P'' = Local Search(Non-dominated(P'))
```

As shown in Fig. 3 (a), the current population does not include many poor solutions. This is because the deterministic generation update mechanism of NSGA-II always chooses the best N_{pop} solutions for the next population. It may be a good idea to apply local search to the current population. One possible implementation of this idea is to choose local search solutions from the current and offspring populations. We denote this version of our MOMA as "(CP-GS)-LS" in order to explicitly show that LS is applied to the current and offspring populations. Except for the selection of local search solutions, the (CP-GS)-LS structure is the same as the CP-GS-LS structure in Fig. 1. Thus only Step 4 of our MOMA algorithm with the CP-GS-LS structure in Fig. 1 is modified for describing the (CP-GS)-LS structure as follows:

[MOMA with the (CP-GS)-LS structure]

```
Step 4:   $P'' = $ Local Search($P \cup P'$)
```

It is also possible to use local search before genetic search in each generation as shown in Fig. 2 in order to apply local search to solutions in the current population. We denote this version as "CP-LS-GS". In the CP-LS-GS structure, parents for recombination are chosen from the improved population. The difference between the CP-LS-GS structure in Fig. 2 and the CP-GS-LS structure in Fig. 1 is only the order of local search (LS) and genetic search (GS). Thus only Step 3 and Step 4 of our MOMA algorithm with the CP-GS-LS structure in Fig. 1 are modified as follows:

[MOMA with the CP-LS-GS structure]

```
Step 3:   $P' = $ Local Search($P$)
Step 4:   $P'' = $ Genetic Search($P'$)
```

One potential difficulty in the CP-LS-GS structure in Fig. 2 is the possibility that the improved population P' is empty (i.e., no solutions are improved by local search in Step 3). Only in this special case, we choose parents for recombination in genetic search from the current population P. This potential difficulty of the CP-LS-GS structure can be easily removed by choosing parents for recombination from the current and improved populations, which leads to the (CP-LS)-GS structure as follows:

[MOMA with the (CP-LS)-GS structure]

```
Step 3:   $P' = $ Local Search($P$)
Step 4:   $P'' = $ Genetic Search($P \cup P'$)
```

4 Computational Experiments

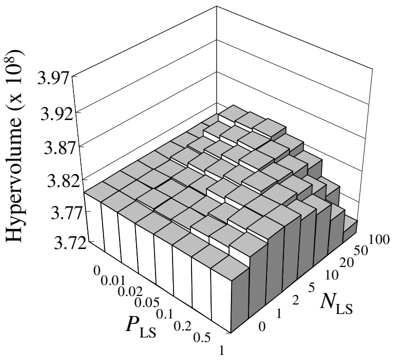
In this section, we examine the effect of the choice of local search solutions on the performance of our MOMA through computational experiments.

Knapsack Problem: We first show experimental results on the two-objective 500-item knapsack problem [19]. Our experiments were performed under the following setting (we used the same greedy repair as in [19] to handle infeasible solutions):

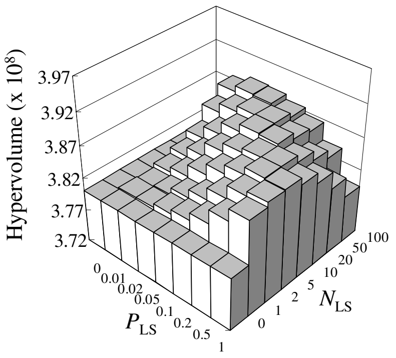
Population size: 200,
Total number of examined solutions (Termination conditions): 400,000,
Tournament size for parent selection in genetic search: 2,
Crossover probability in genetic search: 0.8 (Uniform crossover),
Mutation probability in genetic search: 0.002 (Bit-flip mutation),

Tournament size for local search solution selection: 1, 2, 5, 10, 20, 50,
Neighbor generation in LS: Bit-flip operation with the probability 0.008,
Local search probability: $P_{LS} = 0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0$,
LS length (LS termination condition): $N_{LS} = 0, 1, 2, 5, 10, 20, 50, 100$.

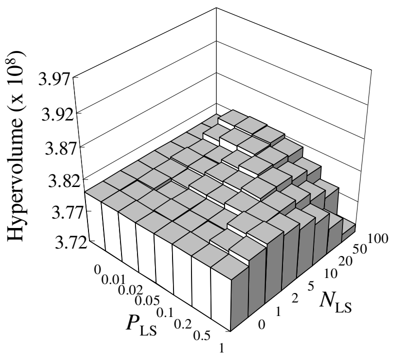
In Fig. 4, we show average hypervolume values over 100 runs by our standard CP-GS-LS MOMA and its non-dominated variant (i.e., selection of only non-dominated offspring for local search). We used the origin (0, 0) of the objective space as the reference point for hypervolume calculation. Since our MOMA is exactly the same as NSGA-II when local search is not used (i.e., when $P_{LS} = 0$ or $N_{LS} = 0$), the left-bottom and left-top rows with the same height bars (about 3.8×10^8 hypervolume) can be viewed as the results of NSGA-II in each plot. We obtained better results from the two variants of our CP-GS-LS MOMA than NSGA-II in a wide range of P_{LS} and N_{LS}. Their performance was, however, severely degraded when both P_{LS} and N_{LS} were too large (i.e., when the genetic search and local search balance was not good).



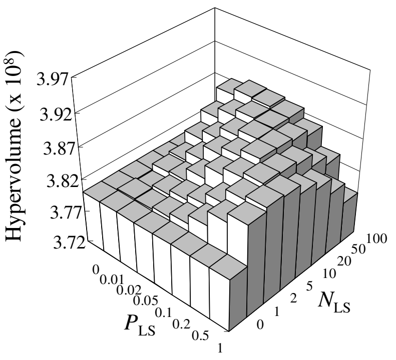
(a) Standard CP-GS-LS MOMA (Size 10).



(b) Standard CP-GS-LS MOMA (Size 50).



(c) Non-dominated CP-GS-LS (Size 2).

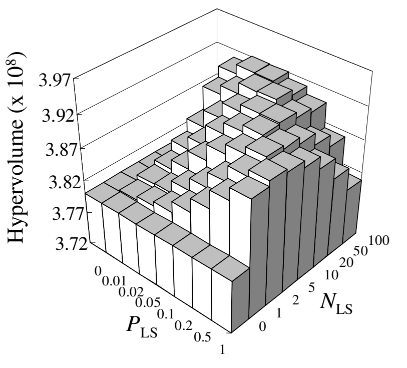


(d) Non-dominated CP-GS-LS (Size 10).

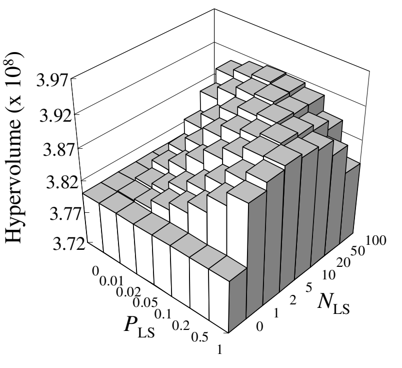
Fig. 4. Two variants of CP-GS-LS (Size: tournament size for local search solution selection)

In Fig. 4, similar results were obtained by the two variants of the CP-GS-LS structure. It should be noted, however, that larger values were used as tournament size in the upper plots than the lower plots. This means that the use of non-dominated solutions for local search in the lower plots has a similar effect to the use of large tournament size for local search solution selection on the performance of our MOMA.

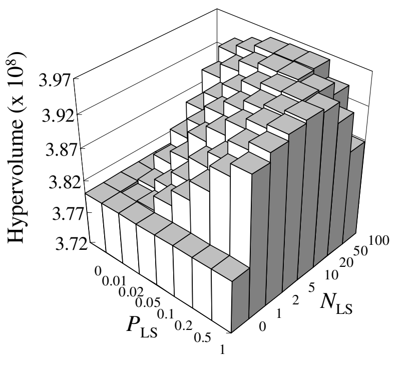
Four variants of our MOMA are compared with each other in Fig. 5 under the same tournament size of 50 for local search solution selection in all the four plots. The best results were obtained from the CP-LS-GS MOMA in Fig. 5 (c).



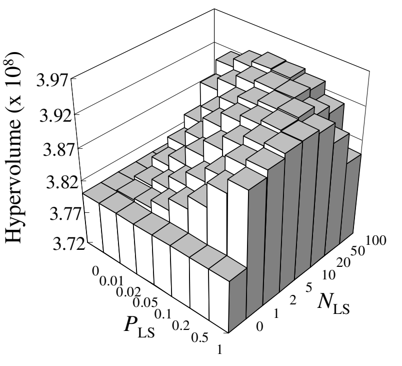
(a) Non-dominated CP-GS-LS (Size 50).



(b) (CP-GS)-LS MOMA (Size 50).



(c) CP-LS-GS MOMA (Size 50).



(d) (CP-LS)-GS MOMA (Size 50).

Fig. 5. Four variants of our MOMA (Tournament size for local search solution selection is 50)

In order to visually demonstrate the statistical significance of the difference in the performance between the CP-GS-LS and CP-LS-GS structures, we show the histogram of 100 hypervolume values obtained from 100 runs of each variant in Fig. 6. In Fig. 6, we used the experimental results with the best combination of P_{LS} and N_{LS} with respect to the average hypervolume in each plot of Fig. 5. For comparison, we also show the results of NSGA-II. We can observe in Fig. 6 that the CP-LS-GS variant clearly outperformed the standard CP-GS-LS MOMA. We can also see that the hybridization with local search clearly improved the performance of NSGA-II.

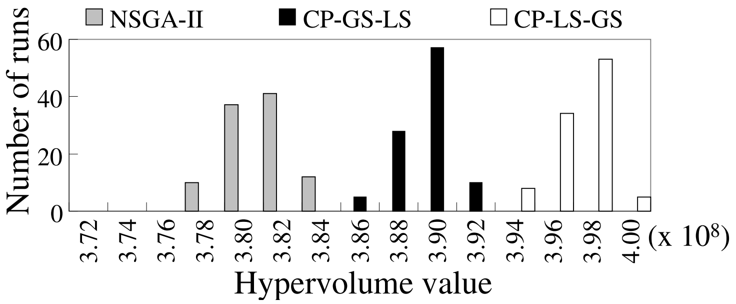


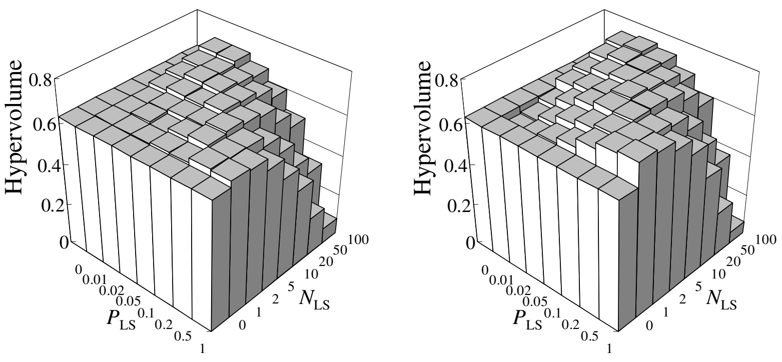
Fig. 6. Histogram of 100 hypervolume values obtained from 100 runs of each algorithm

Flowshop Scheduling: We also applied our MOMA variants to two-objective 20-machine flowshop scheduling problems with 20 and 80 jobs [6]. We used the following setting (The other parameters were the same as in the previous experiments):

Total number of examined solutions (Termination conditions): 100,000,
Crossover probability in genetic search: 0.9 (Two-point crossover [6]),
Mutation probability in genetic search: 0.6 (Insertion mutation [6]),
Neighbor generation in local search: A single use of an insertion operator.

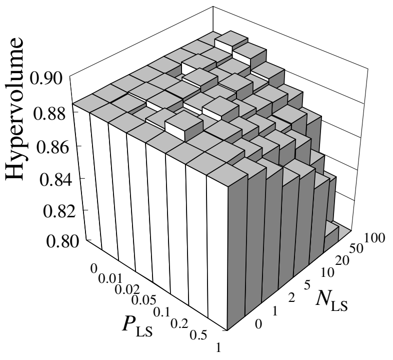
Before hypervolume calculation, we normalized the objective space so that overall non-dominated solutions were in the unit square [0, 1]×[0, 1]. Hypervolume was calculated in the normalized objective space using the reference point (1.1, 1.1). Due to the page limitation, we show a part of experimental results on the 80-job problem in Fig. 7. The best results were obtained from the CP-LS-GS structure with the largest tournament size in Fig. 7 for the 80-job problem as in Fig. 5 on the knapsack problem.

In Fig. 8, we show experimental results on the 20-job problem. The size of the search space of the 20-job problem is 20!, which is much smaller than 80! of the 80-job problem. Thus good results were not obtained from high selection pressure for local search solution selection. The best results were obtained from CP-LS-GS with the tournament size 1 (i.e., random selection) for local search solution selections.

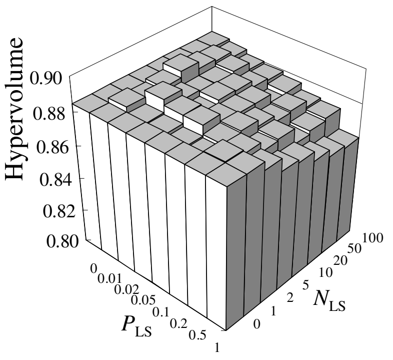


(a) Standard CP-GS-LS MOMA (Size 50). (b) CP-LS-GS MOMA (Size 50).

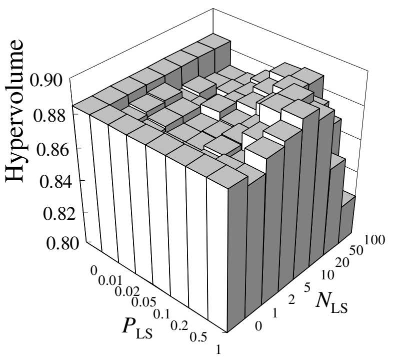
Fig. 7. Results on the two-objective 20-machine 80-job flowshop scheduling problem



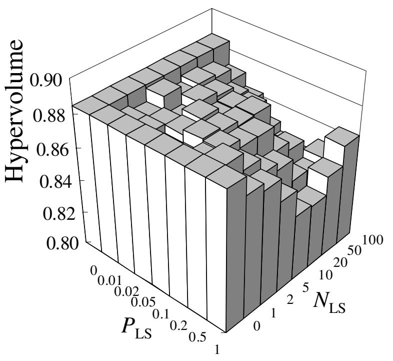
(a) Standard CP-GS-LS MOMA (Size 1).



(b) Standard CP-GS-LS MOMA (Size 50).



(c) CP-LS-GS MOMA (Size 1).



(d) CP-LS-GS MOMA (Size 50).

Fig. 8. Results on the two-objective 20-machine 20-job flowshop scheduling problem

5 Conclusions

We demonstrated that the choice of local search solutions had a large effect on the performance of our MOMA, which is a hybrid algorithm of NSGA-II with local search. Its performance was improved by choosing good solutions for local search through tournament selection with large tournament size. Among four variants of our MOMA, the best results were obtained from a non-standard structure of MOMA: CP-LS-GS. These observations were obtained from our computational experiments on a two-objective 500-item knapsack problem and a two-objective 80-job flowshop scheduling problem. Whereas we did not report due to the page limitation, similar results were also obtained from computational experiments on a three-objective 80-job flowshop problem and 500-item knapsack problems with four and six objectives. The best results, however, were obtained from random selection of local search solutions for a small-size flowshop problem with 20 jobs. Even in this case, the CP-LS-GS structure was the best among the four variants. These observations suggest high potential of the CP-LS-GS structure which has not been examined in many studies in the literature.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6, 182–197 (2002)
2. Ishibuchi, H., Hitotsuyanagi, Y., Tsukamoto, N., Nojima, Y.: Use of Heuristic Local Search for Single-Objective Optimization in Multiobjective Memetic Algorithms. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 743–752. Springer, Heidelberg (2008)
3. Ishibuchi, H., Murata, T.: Multi-Objective Genetic Local Search Algorithm. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 119–124 (1996)
4. Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28, 392–403 (1998)
5. Ishibuchi, H., Narukawa, K.: Some Issues on the Implementation of Local Search in Evolutionary Multiobjective Optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1246–1258. Springer, Heidelberg (2004)
6. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. IEEE Trans. on Evolutionary Computation 7, 204–223 (2003)
7. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research 137, 50–71 (2002)
8. Jaszkiewicz, A.: On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem - A Comparative Experiment. IEEE Trans. on Evolutionary Computation 6, 402–412 (2002)
9. Knowles, J.D., Corne, D.W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. In: Proc. of 2000 IEEE Congress on Evolutionary Computation, pp. 325–332 (2000)
10. Knowles, J.D., Corne, D.W.: A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization. In: Proc. of 2000 Genetic and Evolutionary Computation Conference Workshop Program: WOMA I, pp. 103–108 (2000)
11. Knowles, J.D., Corne, D.W.: Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospective. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 313–352. Springer, Berlin (2005)
12. Krasnogor, N., Smith, J.: A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues. IEEE Trans. on Evolutionary Computation 9, 474–488 (2005)
13. Lara, A., Sanchez, G., Coello, C.A.C., Schutze, O.: HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms. IEEE Trans. on Evolutionary Computation 14, 112–132 (2010)
14. Moscato, P.: Memetic Algorithms: A Short Introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 219–234. McGraw-Hill, London (1999)
15. Murata, T., Ishibuchi, H., Gen, M.: Specification of Genetic Search Directions in Cellular Multi-Objective Genetic Algorithm. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 82–95. Springer, Heidelberg (2001)
16. Ong, Y.S., Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. on Evolutionary Computation 8, 99–110 (2004)
17. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans. on Systems, Man, and Cybernetics: Part B - Cybernetics 36, 141–152 (2006)
18. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation 11, 712–731 (2007)
19. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3, 257–271 (1999)

Secure and Task Abortion Aware GA-Based Hybrid Metaheuristics for Grid Scheduling

Joanna Kołodziej[1], Fatos Xhafa[2], and Marcin Bogdański[1]

[1] Department of Mathematics and Computer Science
University of Bielsko-Biała, ul. Willowa 2, Bielsko-Biała, Poland
jkolodziej@ath.bielsko.pl, mbogdanski@gmail.com

[2] Department of Computer Science and Information Systems
Birkbeck, University of London, Bloomsbury, London WC1E 7HX, UK
fatos@dcs.bbk.ac.uk

Abstract. In traditional distributed computing the users and owners of the computational resources usually belong to the same administrative domain. Therefore security and reliability of the resources are not concerned in such a setting. These issues need to be addressed in scheduling in the Computational Grid systems, where the users and distributed resource clusters work in different autonomous domains. In this paper we present a non-cooperative symmetric game to address the requirements for the security and reliability. The game model takes into account the realistic feature that Grid users usually act independently. The users' cost of playing the game is interpreted as a total cost of the secure job execution, which can be aborted due the machines unreliability and Grid dynamics. The Grid users game is transformed into a bi-level optimization problem, which is solved by four hybrid genetic-based heuristics. We have experimentally evaluated the approach using a Grid simulator under the heterogeneity, the large-scale and dynamics conditions. The relative performance of four hybrid schedulers is measured through the makespan and flowtime metrics. The obtained results suggest that it is worth for the Grid users to pay some additional cost of the verification of the security conditions and possible task abortion in order to achieve an efficient allocation of tasks to the trustful and reliable resources.

Keywords: Scheduling; Security; Computational Grid; Genetic Algorithm; Game Theory; Grid Simulation.

1 Introduction

Computational Grids (CGs) primarily concerned with the development of high-performance applications, which can be executed simultaneously on multiple computers or supercomputers connected by wide-area networks. Unlike traditional distributed computing systems, in which the users and owners of the computational resources usually belong to the same administrative domain, in CGs the security and reliability of the resources are crucial issues. Thus, one of the objectives of the research in this domain is to achieve an efficient assignment of tasks to the trustful machines. However, grid schedulers have recently started to address these issues as the important scheduling criterions. Unfortunately security and resource reliability are addressed separately in most of current approaches, while, because of the complex nature of Grid systems, it is necessary to integrate both those features into the Grid schedulers.

Meeting those additional scheduling requirements we defined in this paper a hierarchical model of Grid, which is aware of trust and task abortion. We adapt a general concept of the Meta-broker, who is in our approach responsible for checking the security condition and the resource availability. Next, we translated it in a non-cooperative symmetric game for for addressing the requirements for the security and resource reliability. This game model takes into account the realistic feature that Grid users usually act independently. We also assume that none of them can have a privileged to resources. The users' cost of playing the game is interpreted as a total cost of the secure job execution, which can be aborted due the machines unreliability and Grid dynamics.

The Grid users' game is then transformed into a bi-level optimization problem, solved through four genetic-based hybrid metaheuristics, which combine GAs and modified Minimum Completion Time method. We have experimentally evaluated the approach using a Grid simulator under the heterogeneity, the large-scale and dynamics conditions. The relative performance of four hybrid schedulers is measured through the makespan and flowtime metrics.

The remainder of this paper is organized as follows. In Sect. 2 we define a secure Grid meta-broker model and recall some preliminary concepts of independent task batch scheduling. The users' game model and is specified in Sect. 2.2. In Sect. 3 four hybrid GA-based schedulers for solving the users' game are defined. An experimental evaluation of proposed hybrid metaheuristics is presented in Sect. 4. The papers is concluded in Sect. 5.

2 Game-Theoretical Model

Computational Grids, hierarchical by their nature, are usually modelled as multi-level large-scale systems for an effective management of tasks and resources. In this work we modify the a simple meta-broker (MB) model (see e.g. [3]) by integrating the resource management with the secure scheduling.

The MB in our system plays the double role of *Trust Manager* and *Resource Manager*. As a trust manager he is responsible for the verification of a *security assurance condition* for each task-machine pair. The verification procedure is as a comparison of the coordinates of a *security demand* vector $SD = [sd_1, \ldots, sd_n]$ and a *trust level* vector $TL = [tl_1, \ldots, tl_m]$ (see [6]) specified for tasks and machines respectively (m-number of machines, n-number of tasks)[1]. The security assurance condition for a given task-machine pair is satisfied if $sd_j \leq tl_{x_j}$. In the other case the failure of machine x_j can be observed. Let us denote by P_f

[1] The values of sd_j and tl_{x_j} are real fractions in the range [0,1] with 0 representing the lowest and 1 the highest security requirements and the most risky and fully trusted machine.

the *Machine Failure Probability* matrix, the elements of which, are interpreted as the probabilities of machines failures during the particular tasks executions due the high security restrictions. These probabilities denoted by $P_f[j][x_s]$ are modelled by an exponential distribution given by the following formulae:

$$P_f[j][x_j] = \begin{cases} 0 & , sd_j \leq tl_{x_j} \\ 1 - e^{-\alpha(sd_j - tl_{x_j})} & , sd_j > tl_{x_j} \end{cases} \tag{1}$$

where α is interpreted as a failure coefficient and is a global parameter of the model.

The other duty of MB in our model is controlling the resource allocation and communication between Grid users and service resource owners. In some cases machines in the grid system could be unavailable due to dynamics or special policies of the resource owners. Analyzing the updated resource providers reports MB generates the reliability probabilities $P_{x_j}, j = 1, \ldots, n$ for each machine x_j. The execution of the given task j can be then aborted with the probability defined as follows:

$$P_{ab}(j) = (1 - P_{x_j}). \tag{2}$$

This task failure predictor was introduced by Rood and Lewis in [5].

2.1 Scheduling Problem Definition

In this work we consider the Independent Job Scheduling problem, in which tasks are processed in the batch mode [9]. The total number of tasks n in the batch can be calculated as the sum of tasks submitted by all users, i.e.: $n = \sum_{l=1}^{N} k_l$, where N is the number of Grid users and k_l is the number of tasks submitted by the user l.

A *schedule* of the batch of tasks at the Grid site is defined as a vector $x = [x_{(1)}, \ldots, x_{(k_1+\ldots+k_l)}, \ldots, x_n]^T$, in which $x_j \in [1, m]$ indicates the number of the machine, to which task j is assigned $(j = 1, \ldots, (k_1 + \ldots + k_l), \ldots, n)$.

The problem formulation in this approach is based on the Expected Time to Compute matrix model [1], in which an instance is defined by: (a)- the computational loads of the tasks (usually in millions of instructions); (b)- the computing capacities of machines (usually in millions of instructions per second, MIPS); (c)- the estimation of the prior load of each available machine and (d)- the *ETC* matrix, the elements of which, define the estimations of the time needed for the completion of the tasks on machines in the system.

2.2 The Secure and Task Abortion Aware Game

We consider the scenario where Grid users perform independently of each other and resource usage privileges are the same for all of them. Each user tries to assign his tasks to the machines to minimize the allocation cost under security and resource reliability criteria. The users cannot cooperate, but they may know the others' previous decisions. The users behavior can be then modelled by a symmetric non-cooperative game defined as a tuple $G_N = (N; \{\{J_l\}; \{Q_l\}\}_{l=1,\ldots,N})$, where:

- N is the number of Grid users;
- $\{J_1, \ldots, J_N\}; l = 1, \ldots, N$ are the sets of users strategies;
- $\{Q_1, \ldots, Q_N\}; Q_l : J_1 \times \ldots \times J_N \to \mathbb{R}; \forall_{l=1,\ldots,N}$ is the set of users cost functions.

Usually the users' scheduling costs are limited to the costs of tasks execution (expressed as a makespan and/or flowtime) or to the resource utilization expressed in the terms of the utility function (see [3]). In our approach an additional cost can come from the possible machine failure as the result of some technical networks problems or special policies of the resource owners. We also consider a resource utilization costs in the terms of the idle times of the machines to which the users tasks are assigned. This is the method of utilization cost calculation from the users point of view, not the resource owners (as it is presented in [3]).

The players cost functions $Q_l, l \in \{1, \ldots, N\}$ in the users' game are composed of the following three factors:

$$Q_l = Q_l^{(s)} + Q_l^{(ab)} + Q_l^{(u)}, \tag{3}$$

where: $Q_l^{(s)}$ indicates the cost of security-assured allocation of the user tasks, $Q_l^{(ab)}$ is the cost of possible abortion of the user's task due the resource unavailability and $Q_l^{(u)}$ denotes a resource utilization cost.

The values of the function $Q_l^{(s)}$ depend on the scheduling strategy and the result of the verification of security condition. We consider in this work two scheduling strategies:

- **risky mode** - in which all risky and failing conditions are ignored by the users. In this case $Q_l^{(s)} = 0$, $l = 1, \ldots, N$.
- **secure mode** - in which $Q_l^{(s)}$ function is defined as follows:

$$Q_l^{(s)} = \sum_{j=(k_1+\ldots+k_{l-1}+1)}^{(k_1+\ldots+k_l)} \frac{P_f[j][x_j] \cdot ETC[j][x_j]}{(ETC)_{m(l)} \cdot k_l}, \tag{4}$$

where $ETC[j][x_j]$ and $P_f[j][x_j]$ are the elements of the ETC and TFP matrices and $(ETC)_{m(l)}$ is the (expected) maximal computation time of the tasks of the user l in a given schedule. This cost is calculated as an average 'wasted' time as the result of failures of the machines during the user's tasks execution due the security restrictions.

The cost of user's tasks abortion due the unavailability of the resources, denoted as $Q_l^{(e)}$, is defined using the following formulae:

$$Q_l^{(ab)} = \frac{\sum_{j=(k_1+\ldots+k_l+1)}^{(k_1+\ldots+k_l)} P_{ab}(j) \cdot ETC[j][x_j]}{(ETC)_{m(l)} \cdot k_l}, \tag{5}$$

where $P_{ab}(j)$ is a task abortion probability defined by 2. It can be interpreted as an average 'wasted' time as the result of the task abortion.

The resource utilization cost in our approach is calculated for each Grid user as an average idle time of machines on which his tasks are executed. We define function $Q_l^{(u)}$ by the following formulae:

$$Q_l^{(u)} = \sum_{x_j \in machines(l)} \left(1 - \frac{Completion_{(l)}[x_j]}{makespan}\right) \cdot \frac{\sum_{j \in Tasks_{(l)}[x_j]} ETC[j][x_j]}{Completion_{(l)}[x_j]} \tag{6}$$

where $Completion_{(l)}[x_j]$ is the completion time of a given machine, $machines(l)$ denotes a set of machines, to which all tasks of the user l are assigned and $Tasks_{(l)}[x_j]$ is the set of the tasks of the user l assigned to the machine x_j. The completion time is calculated as the sum of ready time of this machine and expected computational times of all tasks assigned to it.

Users' game cost function. The objective of playing the game for each user l is to minimize his cost function Q_l. The equilibrium state for the game, where each player holds correct expectations concerning the other players behavior, is the result of the minimization of a *game cost function* $Q : J_1 \times \cdots \times J_N \to \mathbb{R}$ defined by the following formulae:

$$Q(x^1, ..., x^N) = \sum_{l=1}^{N} \big([Q_l(x^1, ..., x^N) - minQ_l]\big), \tag{7}$$

where $minQ_l = \min_{x^l \in J_l} \{Q_l(x^1, ..., x^N)\}, (l = 1, \ldots N)$ denote the minimal values of the players cost functions Q_l calculated independently by the Grid users[2]. To find the equilibrium states is the main objective of the Grid users' game.

To compute the values of the game cost function Q defined by the Eq. (7) we need to minimize first the cost functions of all players. The problem of solving the Grid users game is then defined as a hierarchic procedure composed of two cooperated modules: (1) **Global Module** - where the game cost function is minimized, and (2) **Players Module** - in which the users cost functions Q_l are minimized.

In order to explain the communication mechanism between two modules, let us denote by $x_{(0)} = [x^1_{(0)}, \ldots, x^N_{(0)}]$ an initial schedule for the optimization procedure. Vector $x_{(0)}$ is replicated and sent to the Players Module - one copy per user - and each user independently optimizes his game cost function by changing the assignments of his tasks. Then the optimal values of the players cost functions calculated for the schedule $x_{(0)}$, i.e. $minQ_{l;(0)} = \min_{x^l \in J_l} \{Q_l(x_{(0)})\}; l = 1, \ldots, N$, are sent back to the Global Module[3], where the values of the objective function for the whole game Q (Eq. (7) is calculated for the schedule $x_{(0)}$.

[2] In the case of continuous players' cost functions the solution of the game, given by Eq. (7), is called the *Nash equilibrium* [2].

[3] We denote by x^l the vector of the decision variables of the player l.

3 Genetic-Based Metaheuristics for Solving the Game

We defined four GA-based hybrid metaheuristics for solving the Grid users game and denoted them by *SGA-GA*, *RGA-GA*, *SGA-PMCT* and *RGA-PMCT*. Each hybrid is composed of two GA-based schedulers - *Risky Genetic Algorithm (RGA)* and *Secure Genetic Algorithm (SGA)*- in the Global Module- and two local level optimizers - *Player's Genetic Algorithm (PGA)* and *Player's Minimum Completion Time (PMCT)* - in the Players Module. The main difference between *RGA* and *SGA* lies in the method of calculation of the players costs Q_l, which is different in risky and secure modes.

As the basic mechanism of genetic scheduler in the Global Module we used the GA implementation for independent batch scheduling [7] (see Alg. 1 for its template).

Algorithm 1. Genetic Algorithm template

```
1: Generate the initial population $P^0$ of size $\mu$;
2: Send the $ready\_times$ vectors of the machines corresponding to the individuals of the population
   $P^0$ to the **Players Module**;
3: Receive the $minQ_l$ values from the subordinate unit
4: Evaluate $P^0$;
5: while not termination-condition do
6:     Select the parental pool $T^t$ of size $\lambda$; $T^t := Select(P^t)$;
7:     Perform crossover procedure on pairs of individuals in $T^t$ with probability $p_c$; $P_c^t :=$
       $Cross(T^t)$;
8:     Perform mutation procedure on individuals in $P_c^t$ with probability $p_m$; $P_m^t := Mutate(P_c^t)$;
9:     Send the $ready\_times$ vectors of the machines corresponding to the individuals of the popu-
       lation $P_m^t$ to the **Players Module**;
10:    Receive the $minQ_l$ values from the subordinate unit
11:    Evaluate $P_m^t$ ;
12:    Create a new population $P^{t+1}$ of size $\mu$ from individuals in $P^t$ and/or $P_m^t$ ;
13:    $t := t + 1$;
14: end while
15: return Best found individual as solution;
```

A schedule (an individual in the population) is represented by the vector $x = [x_1, \ldots, x_n]^T$ of machines to which the particular tasks are assigned. In order to implement the crossover and mutation procedures specified for the combinatorial optimization we transformed the vectors x into the permutation-based representation, which is the permutation vector of tasks to machines (see [7] for details).

A combination of the genetic operators applied in Global Module algorithms was selected based on the results of the tuning process performed in [7]. We used linear ranking selection, cycle crossover (CX), re-balancing mutation and elitist generational replacement as the main evolutionary mechanism in Alg. 1. The initial population is generated randomly and the game cost function Q is defined as the fitness.

The algorithms implemented in the Players Module are executed sequentially in order to minimize the users cost functions. We applied two modifications of the well-known Grid schedulers, namely *Player's Minimum Completion Time*

- *(PMCT)* and *Player's Genetic Algorithm-(PGA)*. The first method is a simple modification of *Minimum Completion Time - MCT* heuristic, which is applied independently for each user's tasks. The template of the main mechanism of *PMCT* procedure is defined in Alg. 2.

Algorithm 2. *PMCT* algorithm template

```
1: Receive the population of schedules and $ready\_times$ of the machines from the Main Unit;
2: for all Schedule in the population do
3:     Calculate the completion times of the machines in a given schedule;
4:     for all Individual user do
5:         for all User's Task do
6:             Find the machine that gives minimum completion time;
7:             Assign task to its best machine;
8:             Update the machine completion time;
9:         end for
10:        Calculate the $minQ_l$ value for a given schedule;
11:    end for
12:    Send the $minQ_l$ values to the Main Unit;
13: end for
```

The second *PGA*is an extension of the classical GA-based scheduler [7](with the same combination of the genetic operators as in Alg. 1) applied independently for each user with his cost function Q_l as a fitness. The genetic operators are executed on sub-schedules of the length k_l labeled just by the tasks submitted by user l.

4 Experimental Analysis

In this section we present the results of the experimental evaluation of four hybrid metaheuristics For this we integrated the schedulers with the discrete event-based Grid simulator *HyperSim-G* [8]. The experiments were conducted on two benchmarks composed by a set of static and dynamic instances generated using the simulator [8]. In the static case, the number of tasks and the number of machines is constant during the simulation, while in the dynamic case, both parameters values may vary over time. In both static and dynamic cases four Grid size scenarios are considered: small (32 hosts/512 tasks), medium (64 hosts/1024 tasks), large (128 hosts/2048 tasks), and very large (256 hosts/4096 tasks).

In Tables 1 and 2 we define the settings for the simulator in static and dynamic case and the GA setting at global and local levels for all experiments.

There are 16 Grid users and each of them maintains an equal fraction of the task pool. The coefficients of SD, TL vectors and the machines reliability probabilities P_{x_j} are defined as the uniformly generated fractions in the ranges [0.6 ; 0.9], [0.3 ; 1] and [0.85 ; 1] respectively. The value of failure coefficient α needed for the generation of the TFP matrix (see Eq. 1) is 3.

To evaluate the scheduling performance we used the two main metrics, namely *makespan* and *flowtime*,calculated as follows: $makespan = \max_{j \in Tasks} F_j$ and $flowtime = \sum_{j \in Task} F_j$, where F_j is the time of finishing the task j.

Table 1. Setting for the grid simulator for static and dynamic cases

	Static setting			
	Small	**Medium**	**Large**	**Very Large**
Nb. of hosts	32	64	128	256
Resource cap. (in MIPS)	$N(1000, 175)$			
Total nb. of tasks	512	1024	2048	4096
Workload of tasks	$N(250000000, 43750000)$			
	Dynamic setting			
Init. hosts	32	64	128	256
Max. hosts	37	70	135	264
Min. hosts	27	58	121	248
Resource cap. (in MIPS)	$N(1000, 175)$			
Add host	$N(625000, 93750)$	$N(562500, 84375)$	$N(500000, 75000)$	$N(437500, 65625)$
Delete host	$N(625000, 93750)$			
Total tasks	512	1024	2048	4096
Init. tasks	384	768	1536	3072
Workload	$N(250000000, 43750000)$			
Interarrival	$E(7812.5)$	$E(3906.25)$	$E(1953.125)$	$E(976.5625)$

Table 2. GA setting in the Global and Players Modules for static and dynamic cases

Parameter	**Global Module**	**Players Module**
evolution steps	$5 * n$	$\lceil 0.5 * n \rceil$
population size (*pop_size*)	60	20
intermediate pop.	48	14
cross probab.	0.9	0.9
mutation probab.	0.15	
max_time_to_spend	200 secs (*static*) / 400 secs (*dynamic*)	

Both $flowtime$ and $makespan$ metrics are minimized in the scheduling process over the set of all possible schedules for a given Grid configuration.

Each experiment was repeated 30 times under the same configuration and we report the averaged results.

Experimental results. We present in Fig. 1 the values of flowtime and makespan achieved by four hybrid GA-based schedulers in static and dynamic cases.

It can be observed that in both static and dynamic cases the two *PMCT* hybrids outperform the *RGA-GA* and *SGA-GA* algorithms. For makespan values the differences in the results achieved by *PMCT* and *GA* hybrids are significant, while in the case of flowtime all values are at the same level, except those obtained for very large Grid size.

The best results in all instances are achieved by *SGA-PMCT* algorithm. However, in the case of static scheduling scenario the efficiencies of *RGA-PMCT* and *SGA-PMCT* are very similar, while in the dynamic case, especially for makespan values, the differences in both schedulers performances are significant.

The results of makespan and flowtime values achieved by the schedulers with the same method implemented in the Payers Module suggest that the security criterion is important in all cases. It can be also observed that as the instance size is doubled, the flowtime values increase considerably for all applied schedulers, while the makespan is almost at the same level.

In order to perform a simple statistical analysis of the obtained results we measured the dispersion of the results obtained in each run of the simulator

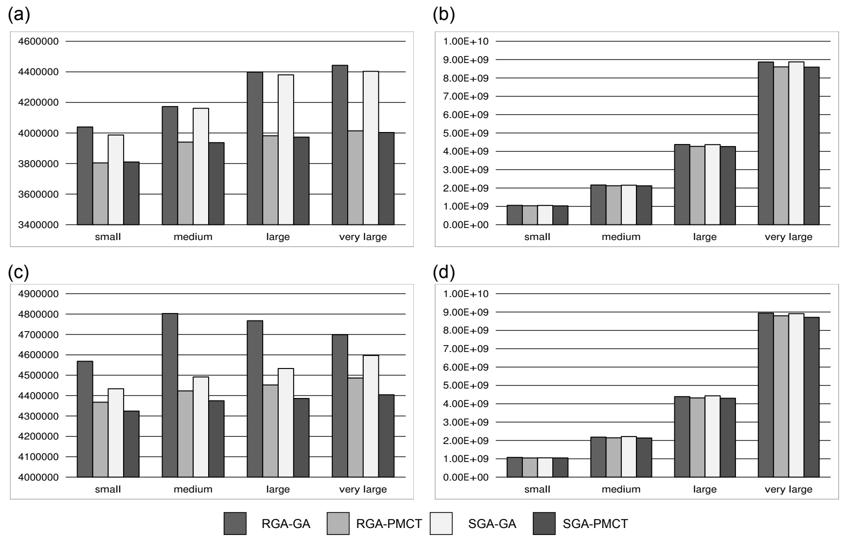


Fig. 1. Experimental results achieved by four hybrid schedulers: in static case - (a) average makespan, (b) average flowtime ; in dynamic case - (c) average makespan, (d) average flowtime

around their average values. We used for that the *coefficient of variation (CV)* [4] which expresses the variation of the data as a percentage of its mean value, i.e.: $CV(x) = s.d./mean(x) \cdot 100\%$. For stable heuristic methods the values of CV should not be greater than 5%. The values of CV calculated for the makespan and flowtime results achieved by four metaheuristics in all considered Grid and scheduling scenarios are presented in Table 3.

It can be noted that just in two cases for makespan values CV is greater than 5%: *RGA-PMCT* for very large grid and *RGA-GA* for large Grid. It is also interesting to observe that the CV for all metaheuristics achieves the biggest values in the case of small Grid scenario. In the other cases CV is very small, which confirms the stability of applied algorithms.

Table 3. CV values for makespan and flowtime measures in large static and dynamic instances

Strategy	**Small**	**Medium**	**Large**	**Very Large**	**Small**	**Medium**	**Large**	**Very Large**
	Makespan				**Flowtime**			
	Large-scale static instances							
RGA - GA	3.02%	2.17%	1.86%	1.65%	2.37%	1.78%	1.25%	0.86%
RGA-PMCT	2.07%	1.46%	3.09%	**7.01** %	2.07%	1.07%	1.02%	1.60%
SGA - GA	3.01%	2.08%	2.08 %	1.55%	2.02%	1.53%	1.25%	0.91%
SGA-PMCT	2.03%	2.16%	4.72%	6.13%	2.11%	1.41%	1.93%	1.82%
	Dynamic instances							
RGA - GA	4.1%	2.7%	**6.85**%	2.7%	3.06%	1.35%	1.6%	0.06%
RGA-PMCT	4.5%	4.95%	3.13%	3.92%	4.5%	4.95%	3.13%	3.92%
SGA - GA	4.7%	3.6%	3.22%	4.78%	3.01%	1.85%	1.05%	0.99%
SGA-PMCT	4.4%	3.9%	3.30%	3.59%	3.00%	1.41%	1.14%	1.00%

5 Conclusions

In this paper we have presented an approach for independent task addressing the requirements for the security and reliability in Grid scheduling. Our approach combines game-theoretic model and Genetic Algorithms based metaheuristics. The former is used to model the scheduling problem as a non-cooperative non-zero sum game of the grid users, while the later are used to minimize the game cost function at global and users' levels. We have evaluated the proposed model under the heterogeneity, the large-scale and dynamics conditions using a Grid simulator. The relative performance of four hybrid schedulers is measured by the makespan and flowtime. The obtained results suggest that it is worth for the Grid users to pay some additional scheduling cost of verification of the security conditions and possible task abortion in order to achieve an efficient allocation of tasks to the trustful and reliable resources.

References

1. Ali, S., Siegel, H.J., Maheswaran, M., Hensgen, D.: Task execution time modelling for heterogeneous computing systems. In: Proceedings of Heterogeneous Computing Workshop (HCW 2000), pp. 185–199 (2000)
2. Edlefsen, L.E., Millham, C.B.: On a formulation of Discrete N-person Non-cooperative Games. Metrika 18(1), 31–34 (1972)
3. Garg, S.K., Buyya, R., Segel, H.J.: Scheduling Parallel Aplications on Utility Grids: Time and Cost Trade-off Management. In: Mans, B. (ed.) Proc. of the 32nd ACSC, Wellington, Australia. CRPIT, vol. 91 (2009)
4. Mann, P.S.: Introductory Statistics, 7th edn. Wiley, Chichester (2010)
5. Rood, B., Lewis, M.J.: Resource Availability Prediction for Improved Grid Scheduling. In: Proc. of 4th IEEE Int. Conf. on eScience (eScience 2008), pp. 711–718 (2008)
6. Song, S., Hwang, K., Kwok, Y.-K.: Risk-resilient Heuristics and Genetic Algorithms for Security- Assured Grid Job Scheduling. IEEE Transactions on Computers 55(6), 703–719 (2006)
7. Xhafa, F., Carretero, J., Abraham, A.: Genetic Algorithm Based Schedulers for Grid Computing Systems. International Journal of Innovative Computing, Information and Control 3(5), 1053–1071 (2007)
8. Xhafa, F., Carretero, J.: Experimental Study of GA-Based Schedulers in Dynamic Distributed Computing Environments. In: Alba, et al. (eds.) Optimization Techniques for Solving Complex Problems, ch. 24. Wiley, Chichester (2009)
9. Xhafa, F., Abraham, A.: Computational models and heuristic methods for Grid scheduling problems. Future Generation Computer Systems 26, 608–621 (2010)

A Memetic Algorithm for the Pickup and Delivery Problem with Time Windows Using Selective Route Exchange Crossover

Yuichi Nagata and Shigenobu Kobayashi

Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology
4259 Nagatsuta Midori-ku Yokohama, Kanagawa 226-8502, Japan
nagata@fe.dis.titech.ac.jp, kobayasi@dis.titech.ac.jp

Abstract. The pickup and delivery problem with time windows (PDP TW) is a variant of the vehicle routing problem. In this paper, we present an memetic algorithm (MA) for the PDPTW. Particular attention is paid to the design of the crossover because it is usually very hard to design an effective crossover operator for tightly constrained problems such as the PDPTW. Experimental results on Li and Lim's benchmarks demonstrate that our MA is competitive with existing approaches and improves 146 best-known solutions out of 298 instances.

1 Introduction

The vehicle routing problem with time window (VRPTW) is one of the most studied NP-hard combinatorial optimization problems. This problem consists in designing a least cost set of routes for a fleet of vehicles to satisfy a set of transportation requests. Each request consists of delivering goods from a depot to a customer within a specified time frame. The pickup and delivery problem with time windows (PDPTW) is an extension of the VRPTW where each request consists of delivering goods from a customer to another one (pickup and delivery constraint). An objective is to minimize a combination of the number of vehicles and the total travel distance.

There have been proposed a lot of heuristic algorithms for the VRPTW and promising heuristic solution approaches are based on evolution strategies [4], large neighborhood search (LNS) [8], iterated local search [2], and memetic algorithm (MA) [6]. In particular, the MA by Nagata *et al.* [6] was proposed very recently and has been one of the most effective algorithms. In fact, several MAs were proposed for the VRPTW, but competitive results had not been obtained. We believe that the main reason is the difficulty in designing an effective crossover operator for the VRPTW because this problem has a fairly constrained search space. In general, it is very hard to combine parent solutions such that offspring solutions inherit meaningful building blocks even if a small violation of constraints is allowed. In the MA [6], so-called edge assembly crossover (EAX) was successfully designed along with a repair procedure.

The pickup and delivery constraint makes the search space of the PDPTW more constrained than that of the VRPTW. State-of-the-art heuristic algorithms for this problem are based on LNS [1][9]. As suggested in [10][1], the use of LNS seems to be a good choice to optimize tightly constrained combinatorial optimization problems such as the PDPTW because of its ability to effectively address side constraints. On the other hand, the existence of the pickup and delivery constraint makes the design of an effective crossover operator more difficult. For example, offspring solutions generated by the EAX will seriously violate the pickup and delivery constraint. By applying an appropriate repair procedure, the constraint violation may be eliminated, but the resulting solutions will no longer inherit meaningful building blocks from the parent. So we believe that it is worthwhile to develop a MA for the PDPTW in order to investigate how we can apply MAs to tightly constrained problems in an effective manner.

In this paper we develop a MA for the PDPTW where particular attention is paid to the design of the crossover. The suggested crossover generates offspring solutions by combining routes from two parents. The key idea is to determine combinations of the routes through a local search procedure so as to approximately minimize the amount of constraint violation in the resulting intermediate offspring solutions, which are converted into feasible solutions by a repair procedure. So we call the suggested crossover *selective route exchange crossover* (SREX). Other part of the MA is basically based on the MA for the VRPTW [6] with several adaptations to the PDPTW.

The remainder of this paper is arranged as follows. The problem definition and notations are described in Section 2. Section 3 presents the framework of the MA followed by the description of the SREX and remaining components. Computational results are presented in Section 4. Section 5 gives conclusions.

2 Problem Definition and Notations

The PDPTW is defined on a complete directed graph $G = (V, E)$ with a set of vertices $V = \{0, 1, \ldots, N\}$ and a set of edges E. Node 0 represents the depot and the set of nodes $\{1, \ldots, N\}$ represents the customers. Let $H = \{1, \ldots, N/2\}$ be a set of requests. For each request h $(\in H)$, let p_h and d_h be the corresponding pickup and delivery customers, respectively, and q_h the amount of goods, i.e. goods specified by request h are picked up at p_h and dropped at d_h in a route. Each node v $(\in V)$ is assigned a time interval. Each edge is assigned a travel distance and travel time. All vehicles have the same capacity.

Given a route for a vehicle that departs from and returns to the depot, the route is called feasible if the following constraints are satisfied. The total amount of goods in the vehicle must not exceed the vehicle capacity at any location (capacity constraint). Each customer must be visited within its time interval (time window constraint). A pickup customer exists in the route if and only if the corresponding delivery customer exists in the route, and the pickup customer must be visited before visiting the delivery customer (pickup and delivery constraint).

A feasible solution is defined as a set of feasible routes such that all customers are visited exactly once. In standard benchmarks, the objective consists of finding a feasible solution that minimizes the number of routes m (primary objective) and, in case of ties, minimizes the total travel distance F (secondary objective).

3 Problem Solving Methodology

Given the hierarchical objective, state-of-the-art heuristic algorithms [1][9] for the PDPTW use the two-stage approach where the number of routes is minimized in the first stage and the travel distance is then minimized in the second stage. The two-stage approach allows us to independently develop algorithms for the route minimization and for the distance minimization. Our MA is also based on the two stage approach. This section first presents the outline of the MA followed by the descriptions of the SREX and remaining components.

3.1 Memetic Algorithm

The procedure of the MA is shown in Algorithm 1. In the first stage, the number of routes is first minimized by executing the route minimization heuristic by Nagata and Kobayashi [7] (line 1). Let m be the minimized number of the routes. The population consisting of N_{pop} solutions, each consisting of m routes, is then created by repeating the route minimization heuristic (line 2).

In the second stage, the main part of the MA minimizes the travel distance F with the number of routes kept constant. For each generation (lines 4–14), each population member is selected once both as parent σ_A^p and as parent σ_B^p in random order (lines 4 and 6). For each pair of parents, σ_A^p and σ_B^p, crossover SREX generates offspring solutions where n_{ch} refers to the number of generated offspring solutions (line 8). A local search procedure is then applied to the offspring solutions to reduce their travel distance (line 10). If the best offspring solution has a smaller travel distance than σ_A^p, it replaces the population member selected as σ_A^p (lines 7, 11 and 13). Iterations of the generation are repeated until the termination condition is met (line 15).

Here, the replacement strategy (lines 7, 11, and 13) may seems to be slightly strange because this strategy replaces only one parent σ_A^p rather than both parents. However, we confirmed that this selection model is superior to conventional ones in maintaining the population diversity because it prevents two parent solutions from being replaced by two similar offspring solutions. In addition, we will design the crossover operator so that offspring solutions tend to be more similar to σ_A^p than σ_B^p to better make use of this selection model.

3.2 Selective Routes Exchange Crossover

The basic idea of the SREX is to combine routes from two parents, σ_A^p and σ_B^p, to generate offspring solutions. In other words, offspring solutions are generated from σ_A^p by replacing some of the routes with different routes selected from σ_B^p.

For two parents, σ_A^p and σ_B^p, we define notations (see Fig. 1 for illustration). Let R_A(and R_B) $= \{1, \ldots, m\}$ be a set of the routes in σ_A^p (and σ_B^p). Let $S_A(\subseteq$

Algorithm 1 : Procedure MA()

```
1: $m$ := DETERMINE_M();
2: $\{\sigma_1, \ldots, \sigma_{N_{pop}}\}$ := GENERATE_INITIAL_POPULATION($m$);
3: repeat
4:    Let $r(\cdot)$ be a random permutation of $1, \ldots, N_{pop}$;
5:    for $i := 1$ to $N_{pop}$ do
6:       $\sigma^p_A := \sigma_{r(i)}$; $\sigma^p_B := \sigma_{r(i+1)}$; (Note: $r(N_{pop}+1) = r(1)$)
7:       $\sigma^c_{best} := \sigma^p_A$;
8:       $\{\sigma^c_1, \ldots, \sigma^c_{n_{ch}}\}$ := CROSSOVER($\sigma^p_A, \sigma^p_B$);
9:       for $j := 1$ to $n_{ch}$ do
10:         $\sigma^c_j$ := LOCAL_SEARCH($\sigma^c_j$);
11:         if $F(\sigma^c_j) < F(\sigma^c_{best})$ then $\sigma^c_{best} := \sigma^c_j$;
12:      end for
13:      $\sigma_{r(i)} := \sigma^c_{best}$;
14:   end for
15: until improvement of the best individual stagnates for the last $g_{stag}$ generations
16: return the best individual in the population;
```

R_A) be a set of the replaced routes on σ^p_A and $S_B(\subseteq R_B)$ a set of the inserted routes selected from σ^p_B. Let V^A_i(and V^B_j) be a set of the customer nodes in route $i(\in R_A)$ (and route $j(\in R_B)$). Let $V_{A\backslash B}$ be a set of the customer nodes that exist in routes S_A but not exist in routes S_B. Let $V_{B\backslash A}$ be a set of the customer nodes that exist in routes S_B but not exist in routes S_A.

Assuming that S_A and S_B are determined, the SREX generates two (type I and type II) offspring solutions by the following steps (see Fig. 1 for illustration).

Procedure SREX-Sub(σ^p_A, σ^p_B, S_A, S_B)
Step 0 Copy σ^p_A into offspring solutions: $\sigma^c_I := \sigma^p_A$ and $\sigma^c_{II} := \sigma^p_A$.
Step 1 (Type I) Remove routes S_A from σ^c_I and then eject customer nodes $V_{B\backslash A}$ from σ^c_I.
Step 1 (Type II) Remove routes S_A from σ^c_{II}.
Step 2 (Type I) Insert routes S_B into σ^c_I.
Step 2 (Type II) Insert routes S_B from which customer nodes $V_{B\backslash A}$ are ejected into σ^c_{II}.
Step 3 Insert the unserved requests (customers) into σ^c (σ^c_I or σ^c_{II} depending on the type). Here, one should note that if a customer p_h is unserved then d_h is inevitably unserved, and vice versa.
  (3-1) Randomly select a request h from the unserved requests.
  (3-2) Insert p_h and d_h into σ^c such that the increase in the travel distance is minimized. Here the two customers can be inserted only if the resulting route is feasible.
  (3-3) If all requests are served or the two customers can not be inserted, then terminate Step 3, otherwise go to (3-1).
Step 4 Return σ^c_I and σ^c_{II} (only feasible solutions are returned).

After Step 2, two intermediate solutions are obtained, where one or more requests are possibly unserved and a set of the customer nodes corresponding to the unserved requests is $V_{A\backslash B}$. Therefore, the number of the unserved requests is given by $|V_{A\backslash B}|/2$ ($|V|$ means the number of the elements in a set V). The

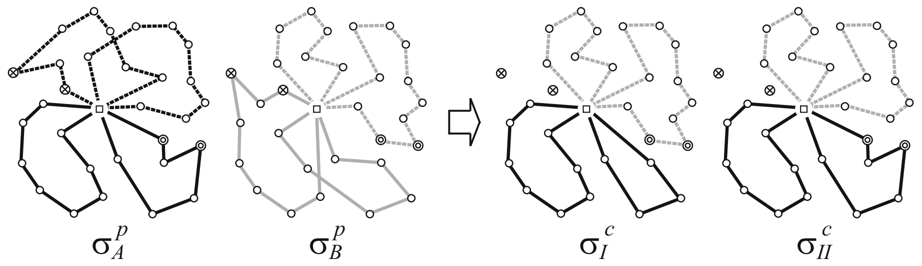


Fig. 1. Illustration of the SREX. σ_A^p and σ_B^p are parents. Routes S_A and S_B are represented as dotted lines, customer nodes $V_{A\backslash B}$ are represented by circles with x-mark, and the customer nodes in $V_{B\backslash A}$ are represented by double circles. σ_I^c and σ_{II}^c are intermediate offspring solutions obtained after Step 2.

SREX has the following properties; (i) Intermediate solutions do not violate the capacity, time window, and pickup and delivery constraints, and (ii) various offspring solutions can be generated depending on the selection of S_A and S_B. However, one problem is that the SREX does not necessarily generate a feasible solution (serving all requests) after Step 3. In fact, we observed that the greater the number of the unserved requests, the lower the probability of generating an feasible offspring solution. So we should select S_A and S_B such that $|V_{A\backslash B}|$ is approximately minimized. In addition, we impose the condition that $|S_A| = |S_B|$ in order to keep the number of routes constant.

To determine S_A and S_B, we use a simple local search procedure where a solution is a pair of S_A and S_B, denoted as (S_A, S_B), and $|V_{A\backslash B}|$ is minimized. The neighborhood is defined as a set of the solutions that are obtained from the current solution (S_A, S_B) by adding one (non-selected) route to each of S_A and S_B or removing one (selected) route from each of them in all possible ways. During the search, we do not accept the following solutions: (i) S_A and S_B are empty or all routes, and (ii) routes S_A are identical to routes S_B. Note that such solutions give $|V_{A\backslash B}| = 0$ but create intermediate solutions identical to either σ_A^p or σ_B^p. The search is started by randomly selecting a pair (S_A, S_B) on condition that $|S_A| = |S_B|$. At each iteration, the current solution is moved to the best solution in the neighborhood. The search is then terminated when no improvement is found in the neighborhood. At each iteration, the evaluation of $|V_{A\backslash B}|$ for all solutions in the neighborhood is too time-consuming without an efficient computation method, which is described in the Appendix.

In our MA, we generate several offspring solutions by selecting several pairs of S_A and S_B. The overall procedure of the SREX is described as follows.

Procedure SREX-Overall(σ_A^p, σ_B^p)

Step 1 Execute the local search procedure N_{total} times, generating a set of N_{total} pairs (S_A, S_B).

Step 2 Eliminate duplication in the obtained set. In addition, we eliminate a pair (S_A, S_B) if the number of the arcs that exist in routes S_B but not exist in routes S_A is greater than the half of the number of arcs that exist in σ_B^p but not exist in

σ_A^p. This criterion is introduced in order to preferably generate offspring solutions which are more similar to σ_A^p rather than σ_B^p as described in Section 3.1.

Step 3 Select top N_{cross} pairs (S_A, S_B) from the obtained set in ascending order of $|V_{A\setminus B}|$. If the obtained set after Step 2 consists of less than N_{cross} pairs, all pairs are selected.

Step 4 For each pair (S_A, S_B) selected in Step 3, procedure SREX-Sub(σ_A^p, σ_B^p, S_A, S_B) generates offspring solutions.

Step 5 Return all offspring solutions obtained through Steps 1–4.

3.3 Local Search

The local search algorithm (Algorithm 1, line 10) is a simple hill climbing algorithm. At each iteration, the current solution is moved to an better solution in the pair relocation neighborhood of the current solution with the first improvement strategy. The pair relocation neighborhood is defined as a set of feasible solutions that are obtained from the current solution by ejecting two customers corresponding to the same request and re-inserting them in all possible way. Iterations are repeated until no better solution is found in the neighborhood.

The local search procedure is the most time consuming part of our MA and we apply the search limitation strategy [5] to reduce the computation time. More precisely, we rule out the moves that do not affect "new" routes where a route is regarded as "new" if the same route does not exist in σ_A^p.

3.4 Generation of the Initial Population

We employ the route minimization (RM) heuristic for the PDPTW [7] to create the initial population of the MA. The RM heuristic starts with an initial solution consisting of $N/2$ routes (each request is served by a different route), and the number of routes is reduced one by one until a termination condition is met.

At the beginning of the MA, the number of routes m is minimized and determined by executing the RM heuristic once for T_1 seconds (Algorithm 1 line 1). After m is determined, the initial population is generated by executing the RM heuristic until N_{pop} solutions, each consisting of m routes, are generated (Algorithm 1, line 2). Each run is executed with the time limit of T_2 seconds (more than N_{pop} runs may be required to generate N_{pop} solutions). We call this way of generating the initial population *complete population strategy* (CPS).

However, CPS is sometimes very time consuming when the minimization of the number of routes is difficult for an instance. So we design another way of generating the initial population. First, the RM heuristic is repeated until N_{pop} solutions, each consisting of m routes, are obtained or until the total execution time reaches T_3 seconds. If the generated solutions consists of less than N_{pop} solutions, these solutions are equally duplicated to generate N_{pop} solutions. The duplicated solutions are then randomly perturbed by iteratively executing random moves selected from the pair relocation neighborhood and pair exchange neighborhood [7] for a given number of times (10000 in our experiment). The perturbation procedure is used also in the RM heuristic and we refer the reader to [7] for more details. We call this way of generating the initial population *incomplete population strategy* (IPS), which will reduce the computation time at the cost of the population diversity in the initial population.

4 Experimental Results

4.1 Experimental Settings and Benchmarks

The proposed MA was implemented in C++ and was executed on an AMD Opteron 2.6GHz computer. The parameters for the main part of the MA were determined by preliminary experiments and set as follows: $N_{pop} = 50$, $g_{stag} = 100$, $N_{total} = 40$, and $N_{cross} = 5$. The paremegters for the generation of the initial population were set as follows: $T_1 = 600$, $T_2 = 300$, $T_3 = N$. These parameter values were determined so as to roughly equalize the computation times for the main part of the MA and for the generation of the initial population with IPS. As described in Section 3.4, we use CPS and IPS for generating the initial population. The MAs were applied five times to each instance.

The MAs were tested on the well-known Li and Lim's benchmarks [3]. The data set and best-known solutions are available at `http://www.sintef.no/projectweb/top`. The benchmarks consist of five data sets of 200, 400, 600, 800 and 1000-customer. Each data set consist of 60 instances and they are divided into two groups (Class 1 and Class2), each consisting of 30 instances (two instances are not available in Class2 1000-customer instance set). Vehicles have smaller capacities and customers have shorter time intervals in Class 1 instances, meaning that more vehicles are required to serve all requests in Class 1 instances.

In the following experiments, we evaluate the solution quality with respect to the travel distance because the main part of the MA is developed for minimizing the travel distance. However, the RM heuristic is very powerful at minimizing the number of routes and the value of m determined by the RM heuristic at the beginning of the MA sometimes improves upon the best-known solution. This makes it difficult to compare our results with previous ones with respect to the travel distance because the optimal travel distance varies depending on the number of routes. So we set m to the number of routes of the best-known solution (we denote it as m^*) in case that m is less than m^*. For each instance, the value of m is determined by executing the RM heuristic once for T_1 seconds and the same value is used for all experiments.

4.2 Analysis of Performance

We compare our results with the best-known solutions to analyze the behavior and performance of the MAs. Table 1 shows several average data for each instance size and class. Note that if m is different from (greater than) m^* on an instance, we ignore such an instance in order to better analyse our results in terms of the ability to minimize the travel distance (see Section 4.1). We present in the table (column #I) the number of instances from which results were obtained.

Table 1. Results of the MAs (instances in which $m \neq m^*$ are ignored). Column "#I" gives the number of instances in which $m = m^*$. Column "W" (and "L") gives the number of instances in which the best travel distance is better (and worse) than that of the best-known solution. The rest of the columns give average data over each instance group. Column "m" gives the number of routes. Columns "best" and "average" give the excess (%) of the best and average travel distances from the best-known travel distance, respectively. Columns "T-S1" and "T-S2" give the CPU time (second) spent for the generation of the initial population and for the main part of the MA, respectively.

			MA with CPS						MA with IPS					
Instance	#I	m	best	ave	W	L	T-S1	T-S2	best	ave	W	L	T-S1	T-S2
200-Class1	30	15.60	-0.054	0.116	9	7	296	19	-0.045	0.272	7	8	111	20
200-Class2	30	4.60	-0.352	0.049	7	4	583	12	-0.219	0.499	7	4	120	13
400-Class1	29	29.79	-0.531	-0.161	19	2	2281	94	-0.229	0.757	16	6	330	93
400-Class2	30	8.77	-0.034	0.723	14	8	2201	135	0.127	1.052	13	8	343	133
600-Class1	28	43.14	-0.920	-0.571	21	3	4783	277	-0.332	0.497	17	9	496	268
600-Class2	26	11.62	-0.350	1.053	12	11	3345	528	0.624	2.701	11	12	630	499
800-Class1	28	55.18	-1.736	-1.315	20	5	6651	656	-1.021	-0.167	15	12	732	645
800-Class2	24	15.21	0.525	1.954	16	8	81791	1458	1.885	3.765	15	9	822	1371
1000-Class1	25	64.56	-1.614	-0.942	21	3	7786	1243	-0.516	0.327	17	7	926	1182
1000-Class2	21	20.67	2.274	4.280	9	10	20018	1588	3.679	6.356	7	12	1155	1668

First, let us focus on the results of the MA with CPS. The best results (best) on the Class 1 instances are better than the best-known solutions for all instance sizes, improving the best-known solutions on 88 instances (but not reaching them on 20 instances). The average results (ave) are also better than the best-known solutions for all instance sizes except for the 200-customer instance set. The best and average results on the Class 2 instances are inferior to those on the Class 1 instances for all instance sizes except for 200, in particular for the 800- and 1000-customer instance sets. This tendency would arise from the fact that the SREX cannot adequately reduce $|V_{A\setminus B}|$ due to the lack of possible pairs of S_A and S_B when the number of routes is too small (see Section 3.2). But the results on the Class 2 instances are still competitive with the best-known solution, improving the best-known solutions on 58 instances (but not reaching them on 41 instances). The computation time for generating the initial population (T-S1) is considerably greater than that for the main procedure of the MA (T-S2) when the initial population is generated with CPS. Next, let us focus on the results of the MA with IPS. We can see that the values of T-S1 are considerably reduced at the cost of the solution quality. However, comparable results are still obtained in particular for the Class 1 instances.

4.3 Comparisons with Other Algorithms

We compare our results with those of state-of-the-art heuristic algorithms listed below: BH (Bent and Hentenryck [1]) and RP (Ropke and Pisinger [9]). BH and RP heuristics were executed ten and five times, respectively, to each instance.

Table 2 compares the results of the four algorithms. Here, we compare the results with respect to the cumulative travel distance (CTD) calculated from the best solutions obtained by the given number of runs. We also report in the

Table 2. Comparisons with state-of-the art algorithms. Columns "CNV" and "CTD" gives the cumulative number of vehicles and the cumulative travel distance, respectively, calculated from the best solutions obtained by the five or 10 runs. Column "Time" gives the average CPU time (second) spent for a run on an instance. At the bottom of the table, specifications on the computers used in the experiments are shown.

	BH (10 runs)			RP (5 runs)			MA with CPS (5 runs)			MA with IPS (5 runs)		
N	CNV	CTD	Time	CNV	CTD	Time	CNV	CTD	Time	CNV	CTD	Time
200	614	180358	3900	606	180931	264	606	179996	456	606	180164	132
400	1188	423636	6000	1158	422201	881	1158	418455	2409	1158	419588	453
600	1718	879940	6000	1679	863442	2221	1670	851174	4866	1670	859028	938
800	2245	1480767	8100	2208	1432078	3918	2189	1416710	13670	2189	1435337	1732
1000	2759	2225190	8100	2652	2137034	5370	2659	2120760	16505	2659	2145564	3011
CPU	Athlon 1.2GHz			Penti. IV 1.5GHz			Opteron 2.6GHz					

table the cumulative number of vehicles (CNV) for reference. Here, one should note that the number of routes can be usually reduced at the cost of the travel distance when it is close to m^*.

Let us first compare the MA with CPS with the RP heuristic (the BH heuristic is dominated by the RP heuristic). The CTDs of the MA with CPS are better than those of the RP heuristic for all instance sizes. However, the computational costs of the MA with CPS are much greater than those of the RP heuristic (we estimate that an Opteron 2.6GHz processor is about three times faster than a Pentium IV 1.5GHz processor). On the other hand, the computational costs of the MA with IPS would not be greater than twice those of the RP heuristic for all instance sizes. The CTDs of the MA with IPS are better than those of the RP heuristic for the 200-, 400-, and 600-customer instance sets but worse than those of the RP heuristic for the 800- and 1000-customer instance sets. As shown in Table 1, the results on the Class 2 800- and 1000-customer instance sets are not good, deteriorating the CTDs of these instance sizes.

5 Conclusions

We have demonstrated that the simple MA framework using the SREX shows very good performance on the standard PDPTW benchmarks. One interesting feature of the SREX is that the local search-based procedure is incorporated into the crossover procedure in order to approximately minimize the amount of the constraint violation in the intermediate offspring solutions. This feature makes it possible to effectively generate offspring solutions on tightly constrained problems such as the PDPTW. The SREX can be applied to most of the other VRP variants without change. In addition, the SREX can be easily adapted to other combinatorial optimization problems such as the graph coloring problem.

References

1. Bent, R., Hentenryck, P.: A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. Computers & Operations Research 33(4), 875–893 (2006)

2. Hashimoto, H., Yagiura, M., Ibaraki, T.: An iterated local search algorithm for the time-dependent vehicle routing problem with time windows. Discrete Optimization 5(2), 434–456 (2008)
3. Li, H., Lim, A.: A metaheuristic for the pickup and delivery problem with timewindows. In: Proc. of the 13th International Conference on Tools with Artificial Intelligence, pp. 160–167 (2001)
4. Mester, D., Bräysy, O.: Active guided evolution strategies for large-scale vehicle routing problems with time windows. Computers & Operations Research 32(6), 1593–1614 (2005)
5. Nagata, Y., Bräysy, O.: Efficient local search limitation strategies for vehicle routing problems. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 48–60. Springer, Heidelberg (2008)
6. Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Computers & Operations Research 37(4), 724–737 (2010)
7. Nagata, Y., Kobayashi, S.: Guided Ejection Search for the Pickup and Delivery Problem with Time Windows. In: Cowling, P., Merz, P. (eds.) EvoCOP 2010. LNCS, vol. 6022, pp. 202–213. Springer, Heidelberg (2010)
8. Prescott-Gagnon, E., Desaulniers, G., Rousseau, L.: A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. Networks 54(4), 190–204 (2009)
9. Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40(4), 455–472 (2006)
10. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Maher, M.J., Puget, J.-F. (eds.) CP 1998. LNCS, vol. 1520, pp. 417–431. Springer, Heidelberg (1998)

Appendix

This section describes how the evaluation function $|V_{A\backslash B}|$ can be computed efficiently during the local search. First, the following notations are defined: $n_i^A = |V_i^A|$, $n_j^B = |V_j^B|$, and $w_{ij} = |V_i^A \cap V_j^B|$ $(i, j = 1, \dots, m)$. These values are computed in advance for a given pair of parents. In addition, we define the following two notations: $W_i^B = \sum_{j \in S_B} w_{ij}$ and $W_j^A = \sum_{i \in S_A} w_{ij}$ $(i, j = 1, \dots, m)$. These values are computed at the beginning of each local search.

Let G be $|V_{A\backslash B}|$ of the current solution (S_A, S_B). From the definition, G is calculated as $G = \sum_{i \in S_A}(n_i^A - \sum_{j \in S_B} w_{ij})$. Let G' be $|V_{A\backslash B}|$ of a solution $(S_A \cup \{i'\}, S_B \cup \{j'\})$, i.e. a solution in the neighborhood obtained by adding routes i' and j' to S_A and S_B, respectively. G' can be expressed in the following formula: $G' = \sum_{i \in S_A \cup \{i'\}}(n_i^A - \sum_{j \in S_B \cup \{j'\}} w_{ij}) = \sum_{i \in S_A}(n_i^A - \sum_{j \in S_B} w_{ij}) - \sum_{i \in S_A} w_{ij'} + n_{i'}^A - \sum_{j \in S_B} w_{i'j} - w_{i'j'} = G + n_{i'}^A - w_{i'j'} - W_{i'}^B - W_{j'}^A$.

Let G'' be $|V_{A\backslash B}|$ of a solution $(S_A \backslash \{i'\}, S_B \backslash \{j'\})$. In the same way, G'' can be expressed in the following formula: $G'' = G - n_{i'}^A - w_{i'j'} + W_{i'}^B + W_{j'}^A$.

According to these formulas, each solution in the neighborhood can be evaluated in constant time. Each time the current solution (S_A, S_B) is moved, W_i^B and W_j^A $(i, j = 1, \dots, m)$ must be updated according to their definitions.

Ant Based Hyper Heuristics with Space Reduction: A Case Study of the p-Median Problem

Zhilei Ren[1], He Jiang[2,3], Jifeng Xuan[1], and Zhongxuan Luo[1]

[1] School of Mathematical Sciences, Dalian University of Technology
{ren,xuan}@mail.dlut.edu.cn, zxluo@dlut.edu.cn
[2] School of Software, Dalian University of Technology
jianghe@dlut.edu.cn
[3] Department of Computer Science, University of Vermont

Abstract. Recent years have witnessed great success of ant based hyper heuristics applying to real world applications. Ant based hyper heuristics intend to explore the heuristic space by traversing the fully connected graph induced by low level heuristics (LLHs). However, existing ant based models treat LLH in an equivalent way, which may lead to imbalance between the intensification and the diversification of the search procedure. Following the definition of meta heuristics, we propose an Ant based Hyper heuristic with SpAce Reduction (AHSAR) to adapt the search over the heuristic space. AHSAR reduces the heuristic space by replacing the fully connected graph with a bipartite graph, which is induced by the Cartesian product of two LLH subsets. With the space reduction, AHSAR enforces consecutive execution of intensification and diversification LLHs. We apply AHSAR to the p-median problem, and experimental results demonstrate that our algorithm outperforms meta heuristics from which LLHs are extracted.

Keywords: Hyper Heuristics, p-Median, Ant Colony Optimization, Meta Heuristics, Heuristic Space Reduction.

1 Introduction

By definition, a hyper heuristic is the process of using heuristics to choose heuristics to solve the problem in hand [1]. Since its emergence, hyper heuristics have been applied to many problems, such as the timetabling problem [2] and the bin packing problem [3]. The main motivation of hyper heuristics is to conduct search over the heuristic space, rather than directly over the solution space. Among various hyper heuristics, ant based algorithms have attracted much attention in that the pheromone trail provides an intuitive but general representation of the heuristic space. Ant based hyper heuristics explore the heuristic space by traversing the fully connected graph induced by LLHs. However, existing ant based hyper heuristics treat LLHs in an equivalent way, such that LLHs of similar functionalities may be executed consecutively, which may lead to imbalance between the intensification and the diversification of the search procedure.

In this paper, we propose an Ant based Hyper heuristic with SpAce Reduction (AHSAR). Unlike existing ant based hyper heuristics, AHSAR reduces the heuristic space into a subspace by restricting LLHs to be selected from the Cartesian product of two subsets of LLHs. The idea is inspired by the definition of meta heuristics, which interprets the process of meta heuristics as the combination of intensification and diversification mechanisms. With the space reduction, AHSAR is able to adapt the intensification and the diversification of the search procedure effectively. As a case study, AHSAR is applied to a classic NP-hard problem, the p-median problem. Experimental results show that our new algorithm outperforms meta heuristics from which LLHs are extracted.

The paper is organized as follows. In Section 2 we introduce related work of both ant based hyper heuristics and meta heuristics. In Section 3, AHSAR is developed to express iterative intensification and diversification searching methodologies. In Section 4, we apply the new algorithm to the p-median problem, and present the experimental results. Finally, conclusion is given in Section 5.

2 Related Work

2.1 Ant Based Hyper Heuristics

In this subsection, we briefly summarize existing work related to ant based hyper heuristics. In 2005, Burke et al. [4] propose an ant based hyper heuristic to solve the project presentation problem. Alberto et al. [5] apply an ant based hyper heuristic with multiple pheromone matrices for the 2D bin packing problem in the same year. A recent ant based hyper heuristic is proposed by Chen et al. [6] to solve the travelling tournament problem in 2007. Various applications have shown the generality of ant based hyper heuristics. In existing ant based hyper heuristics [4,6], a fully connected graph is firstly constructed, where each vertex represents an LLH, and arcs between vertices indicate invokation sequence relationship between heuristics. Each ant is represented as a sequence of LLHs, which is associated with a solution. At each iteration, artificial ants are constructed by traversing the graph. During the construction phase, the selection of LLHs is guided by the pheromone trail, with each entry representing probability of transition between LLHs. After each LLH sequence is constructed and applied over the corresponding solution, the pheromone information is then updated according to the quality of the solutions obtained.

2.2 Meta Heuristic

Over the last few decades, great efforts have been focused on various meta heuristics, including Genetic Algorithm (GA) [7], ACO [8], Tabu Search (TS) [9,10], Variable Neighborhood Search (VNS) [11], Greedy Randomized Adaptive Search (GRASP) [12], etc. Despite appearing to be far different from each other, these algorithms share a few common aspects [13]. By concept, a meta heuristic algorithm is defined as "an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space" [14], or the combination of intensification and diversification. In the definition, exploitation (also called intensification) indicates mechanisms that conduct intensive search in order to improve the quality of solutions; while exploration (also called diversification) refers to various mechanisms that lead the search to diverse regions of the search space. Some ways of achieving intensification and diversification are listed in Table 1.

Table 1. Some ways to achieve intensification and diversification in meta heuristics

Meta Heuristics	Intensification	Diversification
GA	Survival selection	Crossover and mutation
ACO	Local search	Initialization with pheromone
	Pheromone accumulation	Pheromone evaporation
VNS	Local search	Shake
GRASP	Local search	Greedy randomized initialization

As presented in Table 1, meta heuristic algorithms intend to balance the intensification and the diversification mechanisms by combining those intensification and diversification LLH operators. During the intensification process, heuristics such as local search operators are usually applied so as to improve the quality of solutions; while during the diversification process, various perturbing heuristics such as crossover, mutation and shake are employed to restart the search procedure in new regions of the search space.

3 Ant Based Hyper Heuristic with Space Reduction

In this section, we discuss strategies of exploring the heuristic space, and propose our new algorithm, AHSAR. For ant based models, how to traverse the graph derived by LLHs so as to explore the heuristic space plays an essential role. Burke et al. [4] discuss two traversing criteria: the Any Moves (AM) hyper heuristics that accept any LLH sequences, and the Only Improving (OI) hyper heuristics that only accept LLH sequences that improve solution quality. They claim that AM outperforms OI, in that OI strategy provides no diversification mechanism [4]. However, AM strategy treats all LLHs in an equivalent way, thus heuristics with similar functionalities may be invoked consecutively. This may diminish the effect of some LLHs. For example, if a random restart heuristic is invoked immediately after a greedy heuristic, the execution of this greedy heuristic is generally a waste of time. Based on this observation, we propose our new ant model AHSAR, which directly follows the definition of meta heuristics. To enforce the consecutive execution of intensification and diversification LLHs, we replace the fully connected graph induced by the whole LLH set with a bipartite graph derived by two subsets of LLHs.

Without loss of generality, given a minimization problem, let S be the solution space, with objective function $f : S \mapsto \mathbb{R}$, and a heuristic is defined as a function

$h : S \mapsto S$. Let H be the set of LLHs, and $I = \{i | i \in H, \forall s \in S, f(i(s)) \leq f(s)\}$ and $D = \{d | d \in H, \exists s \in S, f(d(s)) > f(s)\}$ be the subset of H that provide intensification and diversification mechanisms, respectively. For heuristics with more than one input solution, such as the crossover of GA, we only need to replace the objective function f with some other evaluation function. Instead of traversing the fully connected graph in search for LLHs, at each iteration of AHSAR, each ant selects a tuple $\langle i, j \rangle$ from the Cartesian product $I \times D$, and then the chosen heuristics are applied. In order to guide the LLH selection, a pheromone matrix τ is incorporated, with each entry τ_{ij} measuring the desirability of selecting $\langle i, j \rangle$. The higher the pheromone value is, the higher the probability of choosing the corresponding tuple will be. In contrast to existing ant based hyper heuristics, the scale of pheromone matrix decreases significantly from $|H \times H|$ to $|I \times D|$.

In addition to the pheromone matrix τ, we also incorporate the heuristic information to balance the probability of choosing each LLH. The idea is intuitive, which is used in other ant based hyper heuristics [4] as well: the computational complexity of LLHs may be quite different from each other, so that we should introduce some mechanism to penalize those time consuming LLHs. For an extreme example, if we consider the exhaustive search as an operator, it can certainly obtain the best solution quality. However, the running time may increase exponentially as the scale of the problem instance grows. Although LLHs with high complexity may achieve high quality solutions, we should also take running time into account. In AHSAR, η_{ij} is defined as:

$$\eta_{ij} = \frac{T_{norm}}{T_i + T_j} \tag{1}$$

where T_i and T_j indicate the running time of LLH i and j, respectively, and T_{norm} is a normalizer in order to balance those problem instances of different scale. In practice, it can be set with the running time of an LLH, which can be determined during the initialization of the algorithm. The definition of η_{ij} implies that we prefer those LLHs with low complexity, with respect to the efficiency.

With τ and η, we can define the probability of choosing the tuple $\langle i, j \rangle$ at each iteration:

$$P_{ij} = \frac{\eta_{ij} \cdot \tau_{ij}}{\sum\limits_{i \in I} \sum\limits_{j \in D} \eta_{ij} \cdot \tau_{ij}} \tag{2}$$

At each iteration, after the tuple $\langle i, j \rangle$ is chosen and applied on the solution associated with each ant k, the pheromone τ is updated using the following rule:

$$\tau_{ij} = \begin{cases} \rho \cdot \tau_{ij} + \dfrac{C^{best}}{C^k} & \text{ant } k \text{ chooses } \langle i, j \rangle, \\ \rho \cdot \tau_{ij} & \text{otherwise.} \end{cases} \tag{3}$$

where C^k and C^{best} represent the objective function of the solution corresponding with ant k and the current best solution, respectively, and ρ indicates the evaporation rate.

```
Algorithm AHSAR
Input: maximum iteration $N$,
       number of ant $K$,
       number of elite ant $k^*$,
       evaporation rate $\rho$
Output: solution $s$
Begin
(1) Initialize $\tau$ and $\eta$ with sufficiently small random values
(2) for $iteration = 1$ to $N$ do
  (3) for $k = 1$ to $K$ do
    (3.1) Choose $\langle i, j \rangle$ with probability defined in equation (2)
    (3.2) Apply LLH tuple $\langle i, j \rangle$ to the solution
          associated with ant $k$
    (3.3) Update $\eta$ with equation (1)
    (3.4) if a better solution has been obtained
    (3.4.1) Record the current best solution with $s$ and update $C^{best}$
  (4) Select $k^*$ solutions with highest quality
  (5) for $k = 1$ to $k^*$ do
    (5.1) Update pheromone with equation (3)
(6) return the best solution obtained $s$
End
```

Fig. 1. Pseudo Code of AHSAR

Fig. 1 presents the pseudo code of AHSAR. First of all, initialization heuristics are treated as special cases of diversification heuristics in order to make the algorithm compact and easier to implement. Thus at each iteration, after the tuple $\langle i, j \rangle$ is selected according to equation (2), the diversification operator j is applied before i. Second, crossover is a pairwise heuristic that requires two parent solutions, thus if the crossover operator is chosen, an extra solution is randomly selected from the population. Third, in step (4), k^* out of K ants are selected, and only these ants are allowed to release pheromone. This mechanism is similar to the survival selection of GA and the elite ant strategy in ACO. We introduce this strategy to penalize those ineffective heuristics, in that the pheromone evaporation mechanism will decrease the probability of choosing them. Finally, an interesting byproduct of the algorithm in this study is that AHSAR actually provides a unified framework under which many existing meta heuristics can be considered as special cases. Although the definition of the two subsets of LLH may not be quite precise (for example, most intensification LLHs are local search operators), the proposed model can express a wide range of meta heuristics. For example, by fixing the value of the pheromone entry corresponding to $\langle \text{Local search, Greedy randomized initialization} \rangle$ to be 1, and that of all the other entries to be 0, AHSAR will degenerate into GRASP. Specifically, if we are considering individual based algorithms such as VNS, we just set parameter $k = 1$, which makes the ant model similar to Fast Ant (FANT) [15].

4 Experimental Results: A Case Study of the p-Median Problem

In this section, we apply our AHSAR to the p-median problem. The reasons we choose the p-median problem are as follows. Firstly, it is a classic NP-hard problem from location theory with wide applications ranging from industry to data mining. Since location theory is a new domain for hyper heuristics, the proposed algorithm demonstrates the generality and the extensibility of hyper heuristics. Secondly, for the p-median problem, there exists many meta heuristics from which LLHs can be extracted, such as VNS [16], ACO [17], and GA [18].

Given a set L of m facilities, a set U of n users, and an $n \times m$ matrix C with the cost traveled c_{ij} for satisfying the demand of the user located at i from the facility located at j, for all $j \in L$ and $i \in U$. The objective of the p-median problem is to minimize the sum of these costs:

$$\min \sum_{i \in U} \min_{j \in J} c_{ij} \tag{4}$$

All LLHs are extracted from those existing meta heuristics mentioned above. Intensification heuristics are listed as follows.

- Interchange: Interchange is first proposed in [19], and widely used in meta heuristics, such as VNS [16] and GRASP [20]. The heuristic iteratively swaps facilities aiming to reduce objective function, until no move can be applied.
- LK(2): LK(k) is extracted from ACO algorithm [17], in which k is a depth parameter. Traversing an LK(k) neighborhood involves k swaps, which is k times that of interchange. For LK(2), parameter k is set to be 2.
- LK($m/2$): Same as LK(k), with $k = m/2$, where m indicates the number of facilities.
- LK(m): Same as LK(k), with $k = m$.

Besides intensification heuristics, diversification heuristics are also listed.

- Crossover and mutation: These two heuristics are extracted from GA [18]. Note that crossover requires two input solutions.
- Initialization with pheromone (AntInit): At each iteration of ACO [17], each ant is constructed with probability according to the pheromone trail. In this study, there are two pheromone matrices, one for the solution space, while the other for the heuristic space.
- Shake: Shake is proposed in VNS [16], and can be viewed as a special case of mutation, whose input is provided by the current best solution.
- Random: Random is actually an initialization operator, which can provide diversification functionality as well.
- Greedy: Greedy is also an initialization operator. Greedy starts with an empty solution, followed by solving p 1-median problems.
- Random plus greedy (RPG): As mentioned above, greedy is a deterministic heuristic, thus in [20], randomness is combined with greedy operator.

Table 2. Results of Algorithms on ORLIB instances

id	ACO	time	GA	time	MStart	time	RANDH	time	AHFAM	time	AHSAR	time
1	5819	2.75	5819	0.81	5819	0.25	5819	2.79	5819	0.77	5819	0.72
2	4093	1.68	4102	0.86	4093	0.26	4093	1.66	4093	0.57	4093	0.52
3	4250	1.70	4250	0.83	4250	0.25	4250	1.70	4250	0.53	4250	0.46
4	3034	1.08	3038	0.83	3034	0.44	3034	1.08	3034	0.40	3034	0.45
5	1355	0.92	1362	0.87	1355	0.36	1355	0.91	1355	0.36	1355	0.37
6	7824	17.84	7824	0.81	7824	0.81	7824	17.83	7824	5.21	7824	4.26
7	5631	8.13	5631	0.83	5631	0.69	5631	8.15	5631	2.07	5631	2.09
8	4445	5.75	4445	0.82	4445	0.84	4445	5.87	4445	1.36	4445	1.25
9	2734	3.33	2734	0.86	2734	1.23	2734	3.20	2734	1.06	2734	1.07
10	1255	3.12	1272	0.86	1255	1.06	1255	3.16	1255	1.06	1255	1.10
11	7696	40.33	7696	0.82	7696	1.48	7696	39.85	7696	9.79	7696	5.98
12	6634	24.21	6634	0.84	6634	1.33	6634	24.20	6634	6.71	6634	5.80
13	4374	7.43	4381	0.85	4374	2.21	4374	7.45	4374	2.20	4374	2.30
14	2968	7.15	2975	0.89	2968	2.37	2968	6.85	2968	2.00	2968	1.87
15	1729	7.28	1736	0.88	1729	2.16	1729	7.22	1729	2.14	1729	2.27
16	8162	121.40	8162	0.83	8162	2.85	8162	121.66	8162	30.71	8162	16.78
17	6999	60.34	6999	0.84	6999	2.04	6999	61.10	6999	15.48	6999	7.58
18	4809	12.70	4815	0.87	4809	4.51	4809	12.78	4809	3.61	4809	3.76
19	2845	10.34	2853	0.94	2846	3.52	2845	10.30	2845	3.17	2845	3.09
20	1789	11.93	1795	0.94	1789	3.53	1789	11.87	1789	3.51	1789	3.22
21	9138	178.31	9138	0.82	9138	4.04	9138	178.78	9138	50.67	9138	28.92
22	8579	137.75	8579	0.94	8579	3.50	8579	139.86	8579	32.11	8579	16.47
23	4619	17.37	4627	0.95	4619	8.13	4619	17.15	4619	5.58	4619	5.78
24	2961	16.04	2977	0.90	2961	5.60	2961	16.36	2961	4.94	2961	4.46
25	1828	21.51	1847	0.97	1828	5.87	1828	20.77	1828	6.24	1828	6.01
26	9917	393.83	9924	1.10	9917	7.63	9917	395.34	9917	87.15	9917	46.96
27	8307	270.92	8307	0.94	8307	5.64	8307	277.16	8307	65.12	8307	28.12
28	4498	24.65	4520	1.06	4498	13.09	4498	24.66	4498	7.73	4498	7.80
29	3033	26.05	3042	1.18	3033	8.57	3033	25.08	3033	7.47	3033	7.29
30	1989	36.01	2004	1.21	1994	9.29	1989	34.70	1989	9.85	1989	7.55
31	10086	666.73	10086	1.00	10086	12.17	10086	663.73	10086	154.79	10086	108.91
32	9297	420.62	9297	0.91	9297	8.57	9297	420.05	9297	103.52	9297	34.86
33	4700	34.36	4723	1.04	4700	20.84	4700	33.76	4700	11.30	4700	11.82
34	3013	42.16	3032	1.30	3014	12.58	3013	40.11	3013	10.78	3013	10.69
35	10400	1124.54	10400	1.29	10400	17.44	10400	1119.59	10400	256.02	10400	127.66
36	9934	815.42	9951	1.30	9934	12.28	9934	804.58	9934	181.60	9934	98.18
37	5057	49.25	5071	1.24	5057	31.29	5057	48.79	5057	15.97	5057	18.26
38	11060	2065.51	11105	0.87	11060	27.03	11060	2106.39	11060	396.10	11060	206.86
39	9423	1027.39	9423	0.87	9423	15.81	9423	1027.65	9423	236.53	9423	106.17
40	5128	63.13	5134	1.09	5129	42.63	5128	64.00	5128	20.00	5128	23.21

All experiments of this paper are performed on a Pentium IV 3.2 GHz PC with 4GB memory, running GNU/Linux with kernel 2.6.32. All the codes are implemented in C++, compiled using gcc 4.4.3 with flag -O2. Running time is measured in seconds, calculated using clock() function. For pseudo random number we use rand() from standard library. The benchmark instance set consists of 40 instances from ORLIB [21], and 10 instances from TSPLIB [22] (We consider instance f1400, with various p indicated by *id* in Table 3). To evaluate the effectiveness and the efficiency of our algorithm, we implement several algorithms for comparison, including ACO [17], GA [18], a multistart local search (denoted as MStart), and a basic hyper heuristic algorithm that randomly chooses LLHs at each iteration (denoted as RANDH). For ant based hyper heuristics, besides AHSAR, we also implement the version that traverses over the fully connected graph with AM strategy (denoted as AHFAM), as discussed in [4].

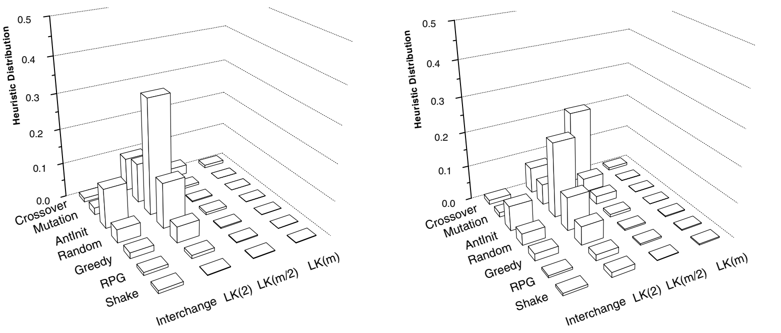
We first compare performance between existing meta heuristics and hyper heuristics proposed in this paper. For all algorithms, parameters are set with same values (population of solutions $K = 10$, maximum iteration $N = 100$, number of elite solutions $k^* = 5$ for ant based hyper heuristics and GA, and evaporation rate $\rho = 0.1$) in order to compare both effectiveness and efficency of each algorithm. For the results presented in Table 2 and Table 3, solutions with best qualities are underscored for each instance. We can observe that solution quality of AHSAR outperforms meta heuristic algorithms from which LLH

Table 3. Results of Algorithms on TSPLIB instances

id	ACO	time	GA	time	MStart	time	RANDH	time	AHFAM	time	AHSAR	time
50	29090.22	272.79	29328.30	1.31	29090.23	46.73	29090.23	273.47	29090.22	72.35	29090.22	100.88
100	16559.32	143.30	16759.88	1.59	16569.18	93.57	16562.20	145.16	16566.57	51.46	16552.22	72.74
150	12045.58	178.18	12100.59	1.68	12054.59	45.88	12049.74	175.09	12051.25	46.01	12026.43	77.38
200	9361.34	176.85	9409.70	2.34	9385.97	46.82	9365.61	178.43	9362.59	47.74	9359.05	64.23
250	7747.84	206.06	7781.28	2.51	7761.73	50.49	7748.78	209.76	7743.91	40.05	7742.67	62.97
300	6629.41	224.28	6674.16	2.08	6649.03	52.03	6629.98	227.09	6627.53	44.80	6623.81	107.64
350	5743.23	279.56	5784.00	3.13	5766.73	55.75	5738.79	280.16	5736.31	50.02	5723.47	93.28
400	5030.44	326.14	5093.38	2.72	5058.47	60.09	5026.96	324.66	5017.05	72.92	5009.67	116.64
450	4478.42	440.36	4525.08	3.06	4500.72	63.81	4480.79	452.71	4481.81	88.14	4476.72	112.18
500	4052.17	489.50	4086.64	3.59	4068.32	65.72	4053.38	495.88	4053.94	92.40	4048.96	194.46

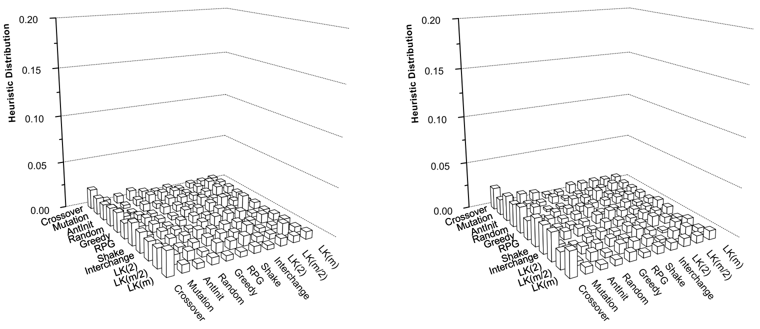
are extracted. AHSAR obtains best solution for all benchmark instances. When running time is concerned, we can see that both AHSAR and AHFAM run faster than ACO. Although ACO obtains best quality among meta heuristics presented, high complexity of LK(m) that ACO employs makes it quite time consuming. With help of the heuristic information η in our algorithm, running time is significantly reduced. We can also observe the effecacy of the pheromone matrix, in that both AHSAR and AHFAM achieve solutions with higher quality than RANDH in less computation time. Besides, the comparison between AHFAM and AHSAR implies the effect of the space reduction mechanism. With the reduction mechanism, the combination of intensification and diversification contributes to the better quality of the solution obtained. Although AHSAR runs slower than AHFAM for some instances, the quality-time trade-off is worthy.

More insights can be gained by analyzing the distribution of LLHs over different instances. Fig. 2 depicts probabilities of selecting different combinations of LLHs during one run of AHSAR. From Fig. 2, we can see that given different problem instances, the distribution of LLH combinations exhibits different patterns. For example, over orlib40 and fl1400 with $p = 200$, the probability of selecting $\langle \text{LK}(2), \text{AntInit} \rangle$ varies from 40% to 22%. However, LLH distribution of AHFAM doesn't exhibit different pattern over different instances, as is illustrated in Fig. 3. The observation implies that our new algorithm is self-adaptive. Besides, Fig. 2 implies the effect of the heuristic matrix η and the elite ant



(a) ORLIB40 with $m = 900, p = 90$ (b) FL1400 with $m = 1400, p = 200$

Fig. 2. ARSAR's distribution of LLHs over different instances



(a) ORLIB40 with $m = 900, p = 90$ (b) FL1400 with $m = 1400, p = 200$

Fig. 3. AHFAM's distribution of LLHs over different instances

strategy. From Fig. 2 we can see that the LK(m) operator is seldom chosen, in that its complexity is too high; while the probability of selecting LLHs such as Random and RPG is also low, due to their poor performance.

5 Conclusion

In this paper a new ant based hyper heuristic with space reduction is developed. The main contribution of this paper can be concluded as follows. (1) A new ant model is proposed. We explicitly follow the definition of meta heuristics by dividing LLHs according to their functionalities, and the search space of LLHs is reduced by replacing the fully connected graph constructed by all LLHs with a bipartite graph derived by two subsets of LLHs, which can be explored more effectively. (2) We apply the proposed algorithm to the p-median problem for the first time, which demonstrate the generality of hyper heuristics. (3) Numerical results show that our algorithm outperforms the meta heuristics from which the new algorithm is derived.

Acknowledgments. Our work is partially supported by the Natural Science Foundation of China under Grant No. 60805024, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20070141020.

References

1. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A Classification of Hyper-heuristics Approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science. Springer, Heidelberg (2009) (in press)
2. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A Graph-based Hyper-heuristic for Educational Timetabling Problems. European Journal of Operational Research 176(1), 177–192 (2007)

3. Privosnik, M.: The Scalability of Evolved On Line Bin Packing Heuristics. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 2530–2537 (2007)
4. Burke, E.K., Kendall, G., Silva, D.L., O'Brien, R., Soubeiga, E.: An Ant Algorithm Hyperheuristic for the Project Presentation Scheduling Problem. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2263–2270 (2005)
5. Cuesta-Cañada, A., Garrido, L., Terashima-Marín, H.: Building Hyper-heuristics Through Ant Colony Optimization for the 2D Bin Packing Problem. In: Knowledge-Based Intelligent Information and Engineering Systems, pp. 654–660. Springer, Heidelberg (2005)
6. Chen, P.C., Kendall, G., Berghe, G.: An Ant Based Hyper-heuristic for the Travelling Tournament Problem. In: IEEE Symposium on Computational Intelligence in Scheduling, SCIS 2007, pp. 19–26 (2007)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1992)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B 26(1), 29–41 (1996)
9. Glover, F.: Tabu search – Part I. ORSA J. on Computing 1, 190–206 (1989)
10. Glover, F.: Tabu search – Part II. ORSA J. on Computing 2, 4–32 (1990)
11. Hansen, P., Mladenović, N.: Variable Neighborhood Search: Principles and Applications. European Journal of Operational Research 130(3), 449–467 (2001)
12. Feo, T.A., Resende, M.G.C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6(2), 109–133 (1995)
13. Taillard, É.D., Gambardella, L.M., Gendreau, M., Potvin, J.Y.: Adaptive Memory Programming: A Unified View of Metaheuristics. European Journal of Operational Research 135(1), 1–16 (2001)
14. Osman, I.H., Laporte, G.: Metaheuristics: A Bibliography. Annals of Operations Research 63(5), 511–623 (1996)
15. Taillard, É.D.: Ant Systems. In: Pardalos, P., Resende, M.G.C. (eds.) Handbook of Applied Optimization, pp. 130–137. Oxford Univ. Press, Oxford (2002)
16. Hansen, P., Mladenović, N.: Variable Neighborhood Search for the p-Median. Location Science 5(4), 207–226 (1997)
17. Kochetov, Y., Levanova, T., Alekseeva, E., Loresh, M.: Large Neighborhood Local Search for the p-Median Problem. Yugoslav Journal of Operations Research 15(1), 53–63 (2005)
18. Alp, O., Erkut, E., Drezner, Z.: An Efficient Genetic Algorithm for the p-Median Problem. Annals of Operations Research 122, 21–42 (2003)
19. Teitz, M.B., Bart, P.: Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. Operations Research 16(5), 955–961 (1968)
20. Resende, M.G.C., Werneck, R.F.: A Hybrid Heuristic for the p-Median Problem. Journal of Heuristics 10(1), 59–88 (2004)
21. Beasley, J.E.: A Note on Solving Large p-Median Problems. European Journal of Operational Research 21, 270–273 (1985)
22. Reinelt, G.: TSPLIB–A Traveling Salesman Problem Library. ORSA Journal on Computing 3, 376–384 (1991)

A Study of Multi-parent Crossover Operators in a Memetic Algorithm

Yang Wang, Zhipeng Lü*, and Jin-Kao Hao

LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers Cedex 01, France
{yangw,lu,hao}@info.univ-angers.fr

Abstract. Using unconstrained binary quadratic programming problem as a case study, we investigate the role of multi-parent crossover operators within the memetic algorithm framework. We evaluate the performance of four multi-parent crossover operators (called MSX, Diagonal, U-Scan and OB-Scan) and provide evidences and insights as to why one particular multi-parent crossover operator leads to better computational results than another one. For this purpose, we employ several indicators like population entropy and average solution distance in the population.

Keywords: multi-parent crossover, memetic algorithm, unconstrained quadratic programming, performance analysis.

1 Introduction

Memetic algorithms (MA) are known to be one of the highly effective metaheuristic approaches for solving a large number of constraint satisfaction and optimization problems [1]. One of the most important features of a MA is the crossover operator for generating offspring solutions. In general, meaningful crossover operators help to create healthy diversification in the population and to avoid a premature convergence of the population.

In this paper, we provide a case study of multi-parent crossover operators within memetic algorithms for the unconstrained binary quadratic programming (UBQP) that can be written

$$\text{UBQP: Maximize } f(x) = x'Qx$$
$$x \text{ binary}$$

where Q is an n by n matrix of constants and x is an n-vector of binary variables.

The formulation UBQP is notable for its ability to represent a wide range of important problems [2]. The literature reports a number of evolutionary and memetic algorithms with two-parent crossover operators for solving the UBQP problem ([3,4,5,6]). However, one finds no studies concerning multi-parent crossover operators for UBQP, where multi-parent crossover operators generate offspring solutions by combining more than two parent solutions. In this work, we are particularly interested in investigating the role of multi-parent crossover operators as well as a number of related important questions: why does one particular multi-parent crossover operator lead to better computational results than

* Corresponding author.

another one? What are the main characteristics of a good multi-parent crossover operator? To what extent can the crossover operators influence the performance of the memetic algorithms?

Without claiming to answer all these questions, we present an experimental analysis of various multi-parent crossover operators within a memetic algorithm. For this purpose, we use four multi-parent crossover operators, respectively called MSX, Diagonal, U-Scan and OB-Scan. The last three ones are well known in the literature while the first one is recently proposed in [7]. The analysis shows that the computational results are strongly correlated with the characteristics of the corresponding crossover operators, such as the entropy and average solution distance of the population, the average solution quality in the population. Furthermore, the analysis sheds light on how a tradeoff between local search and crossover operator can be achieved when using different crossover operators.

2 Multi-parent Crossover within Memetic Algorithms

2.1 Main Scheme and Initial Population

This study is based on the general memetic framework described in Algorithm 1 that alternates between a combination operator and a local improvement procedure. The combination operator (Section 2.4) is used to generate new offspring solutions while the local improvement procedure based on tabu search (Section 2.2) aims at optimizing each offspring solution. As soon as an offspring solution is improved by tabu search, the population is accordingly updated based on two criteria: the solution quality and the diversity of the population. The individuals of the initial population are generated randomly (i.e., each variable x_i of the n-vector x receives a value of 0 or 1 with equal probability).

Algorithm 1. Pseudo-code of the memetic algorithm

```
1: Input: matrix $Q$
2: Output: the best solution $x^*$ found so far
3: $P = \{x^1, \ldots, x^p\} \leftarrow$ Population_Initialization( )
4: for $i = \{1, \ldots, p\}$ do
5:     $x^i \leftarrow$ Tabu_Search($x^i$)
6: end for
7: $x^* = arg\ max\{f(x^i) | i = 1, \ldots, p\}$
8: repeat
9:     randomly choose a subset of individuals $E$ ($|E| \in [4, 8]$) from $P$
10:    $x^0 \leftarrow$ Crossover_Operator($E$)
11:    $x^0 \leftarrow$ Tabu_Search($x^0$)
12:    if $f(x^0) > f(x^*)$ then
13:        $x^* = x^0$
14:    end if
15:    $P \leftarrow$ Pool_Updating($x^0, P$)
16: until a stop criterion is met
```

2.2 Tabu Search Procedure

In this paper, we employ a simple tabu search algorithm as our local search procedure. Our tabu search procedure uses a *neighborhood* defined by the simple *one-flip move*, which consists of changing (flipping) the value of a single variable x_i to its complementary value $1 - x_i$. The implementation of this neighborhood uses a fast incremental evaluation technique [8] to calculate the cost (move value) of transitioning to each neighboring solution.

Tabu search incorporates a *tabu list* as a "recency-based" memory structure to assure that solutions visited within a certain span of iterations, called the tabu tenure, will not be revisited [9]. In our implementation, we elected to set the tabu tenure as $TabuTenure(i) = tt + rand(10)$, where tt is a given constant $(n/100)$ and rand(10) takes a random value from 1 to 10. Our tabu search method stops when a given number α of moves are reached, called *depth* of tabu search.

2.3 Pool Updating

In our memetic algorithm, after an offspring x^0 is obtained by the crossover operator and improved by tabu search, we decide whether the improved offspring should be inserted into the population, replacing the existing *worst* solution. For this purpose, we define a quality-and-distance goodness score of the offspring x^0 with respect to the population. The main idea is to favor the inclusion of x^0 in the population if x^0 is "good enough" (in terms of its objective function evaluation) and is not too *similar* to any solution currently in the population.

Our aim is not only to maintain a pool of good quality solutions but also to emphasize the importance of the diversity of the solutions to avoid a premature convergence of the population. Therefore, if the goodness score of the offspring solution is good enough, it will have high probability to replace the worst solution in the population. Interested readers are referred to [7] for more details.

2.4 Combination Operators

In this paper, we use four multi-parent crossover (or combination) operators to generate offspring solutions, including a "logic" multi-parent combination operator (MSX), a diagonal multi-parent crossover (Diagonal), a multi-parent uniform scanning crossover (U-Scan), a multi-parent occurrence based scanning crossover (OB-Scan). Note that except MSX which is recently proposed for UBQP [7], the last three ones have been widely used for other combinatorial optimization problems in the literature [10,11].

All the four combination operators used in our algorithm are applied on a set E of s ($|E| = s$) parent solutions randomly selected from the current population, i.e., $E = \{x^{(1)}, \ldots, x^{(s)}\}$, where $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$. In our implementation, we set s to be a random number between 4 and 8.

MSX Crossover (MSX): we define a weight $w(i)$ for the solution $x^{(i)}$ and a strength value $Strength(j)$ for variable x_j as: $w(i) = 1/\sum(i) = 1/\sum_{j=1}^{n} x_j^{(i)}$ and $Strength(j) = \sum_{i=1}^{s} w(i) x_j^{(i)}$.

The value $Strength(j)$ gives a relative indication of the tendency of the solutions in E to favor $x_j = 1$ or $x_j = 0$. That is, we may say that the larger the value of $Strength(j)$, the greater is the degree that "E favors $x_j = 1$". Then, we take an average of the $sum(i)$ values over E to get a value for the number of x_j components that should be 1 in an "average" solution, denoted by $Avg = \sum_{i=1}^{s} sum(i)/s$.

Thus, the variables with the first Avg largest $Strength$ values receive assignment 1 and other variables receive assignment 0. In practice, it is preferable to shift Avg slightly in one direction or another to increase the diversity of the generated offspring solutions [7].

Diagonal Crossover (Diagonal): it is a generalization of the one-point crossover. For s parent solutions, diagonal crossover divides each parents into s sections through $s-1$ crossover points. Each section has the same length except the last section containing the surplus variables when divided unequally. The offspring is constructed through extracting in a diagonal way respectively one section from each parent. The formal definition is given as follows.

Given set E with s solutions and $s-1$ crossover points $\{y_1, y_2, ..., y_{s-1}\}$, where $y_i = i * n/s$ and $0 < i < s$. Diagonal crossover reproduces the offspring $c = (c_1, c_2, ...c_s)$ by

$$c_k = \begin{cases} x_j^{(1)}, 1 \le j < y_1 & k = 1; \\ x_j^{(k)}, y_{k-1} \le j < y_k & 1 < k < s; \\ x_j^{(s)}, y_{s-1} \le j \le n & k = s. \end{cases} \tag{1}$$

Uniform Scanning Crossover (U-Scan): this is a generalization of the two-parent uniform crossover. U-Scan uses a scheme that one of the parents selected randomly determines the value of the offspring. Thus each parent has the same probability to be the dominator of the value inherited by the offspring. It breaks the limitation of traditionary two parents and can extend the number of parents to an arbitrary number.

Given set E with s solutions, U-Scan generates offspring solution $c = (c_1, c_2, ... c_n)$ as follows. Value c_j is obtained by $c_j = x_j^{(i)}$ where $x_j^{(i)}$ denotes the jth value of parent $x^{(i)}$ and i is randomly selected from 1 to s with probability $1/s$.

Occurrence-Based Scanning Crossover (OB-Scan): OB-Scan relies on parental occurrence on determining the offspring values. Generally speaking, each parent votes and the values inherited will be the one favored by the majority of parents. As for UBQP problem, the value equals either one or zero, thus we record the frequency of each variable's value equal to one appearing in the parents. If this frequency surpasses or is less than the half of the number of parents, then the value of offspring in this position is assigned to one or zero, respectively. Otherwise, the variable is assigned to be one or zero randomly. The following gives a formal definition of OB-Scan.

Given set E, OB-scan reproduces the offspring $c = (c_1, c_2, ...c_n)$ by

$$c_j = \begin{cases} 0, & \sum_{i=1}^{s} x_j^{(i)} < s/2; \\ 1, & \sum_{i=1}^{s} x_j^{(i)} > s/2; \\ rand(0,1), & \text{otherwise.} \end{cases} \qquad (2)$$

where $rand(0,1) \in \{0,1\}$ is a binary random function.

3 Experimental Results

3.1 Instances and Experimental Protocol

To evaluate the MSX, Diagonal, U-Scan and OB-Scan crossover operators, we carry out experiments on a set of 15 large random instances with 3000 to 5000 variables from the literature [12]. Our algorithm is programmed in C and compiled using GNU GCC on a PC running Windows XP with Pentium 2.66GHz CPU and 512MB RAM. Given the stochastic nature of the algorithm, problem instances are independently solved 10 times. The stop condition for a single run is respectively set to be 5, 10 and 20 minutes on our computer for instances with 3000, 4000 and 5000 variables, respectively. Note that when performing experiments on each crossover, the only difference consists in the crossover operator and other components of the algorithm are kept unchanged. The parameters are set as follows: population size $p = 30$, *depth* of tabu search $\alpha = 2n$. The experimental results are summarized in Tables 1 and 2.

3.2 Computational Results

Tables 1 and 2 report the best objective values (in parentheses number of hits over 10 runs) and the average objective values using the four crossover operators,

Table 1. Computational results on the 15 large random instances with 3000 to 5000 variables: best values (succ rate)

Instance	MSX	U-Scan	OB-Scan	Diagonal
p3000.1	3931583(9)	**3931583(10)**	3931583(8)	3931583(9)
p3000.2	5193073(10)	5193073(10)	5193073(10)	5193073(10)
p3000.3	5111533(7)	5111533(7)	5111533(7)	5111533(6)
p3000.4	5761822(10)	5761822(10)	5761822(9)	5761822(10)
p3000.5	**5675625(6)**	5675625(1)	5675598(1)	5675625(4)
p4000.1	6181830(10)	6181830(10)	6181830(10)	6181830(10)
p4000.2	**7801355(7)**	7801355(6)	7801355(4)	7801355(6)
p4000.3	7741685(9)	7741685(9)	7741685(9)	7741685(7)
p4000.4	**8711822(10)**	8711822(9)	8711822(7)	8711822(9)
p4000.5	8908979(4)	**8908979(7)**	8908979(3)	8908979(2)
p5000.1	8559015(1)	**8559312(1)**	8559210(3)	8559210(4)
p5000.2	10835437(1)	10835832(3)	10835437(3)	**10835437(5)**
p5000.3	10488783(10)	**10489137(3)**	10489137(1)	10489137(1)
p5000.4	12251211(1)	12251211(2)	12251211(2)	12251211(1)
p5000.5	**12731803(10)**	12731803(1)	12731803(1)	12731803(1)

respectively. We observe that MSX and U-Scan perform slightly better in terms of the best objective values since both two crossovers obtain the best values for 4 out of the 15 instances. OB-Scan seems to be the worst in terms of the best objective value and the success rate. Table 2 indicates that MSX seems to be superior to other three crossover operators in terms of the average objective values since it reaches the best results for 7 out of the 15 instances. Moreover, U-Scan and Diagonal perform quite well on 3 and 2 instances, respectively. In addition, the results also disclose that the performance of various crossover operators depend on the instances to be solved. For example, MSX operator dominates the other three ones on instances p3000.5 and p5000.5; U-Scan obtains excellent results on instance p5000.3; Diagonal performs the best on instances p5000.2.

Table 2. Computational results on the 15 large random instances with 3000 to 5000 variables: average objective function values over 10 runs

Instance	MSX	U-Scan	OB-Scan	Diagonal
p3000.1	3931522	**3931583**	3931368	3931418
p3000.2	5193073	5193073	5193073	5193073
p3000.3	**5111403**	5111307	5111292	5111194
p3000.4	5761822	5761822	5761784	5761822
p3000.5	**5675360**	5675041	5674950	5675130
p4000.1	6181830	6181830	6181830	6181830
p4000.2	**7801170**	7800556	7799766	7800731
p4000.3	7741493	**7741653**	7741557	7741541
p4000.4	**8711822**	8711775	8711410	8711582
p4000.5	**8908438**	8908189	8907347	8906880
p5000.1	8558848	8558945	8558916	**8558963**
p5000.2	10834470	**10835107**	10834762	10834987
p5000.3	**10488783**	10487995	10487865	10487941
p5000.4	12250012	12250161	12249935	**12250559**
p5000.5	**12731803**	12730255	12730454	12730563

4 Analysis

The above computational results show that for certain instances, some crossover operators perform better than other ones in terms of the solution quality. In this section, we attempt to explain what causes the effectiveness or weakness of a crossover operator. For this purpose, we introduce the following evaluation criteria to characterize the search capacity of different crossover operators: population entropy, average solution distance and average solution quality in the population. We argue that a good crossover operator should help the population to maintain a good diversity (high entropy and high average solution distance) and good average solution quality. We also perform an experiment to show how different crossover operators and local search jointly influence the performance of the memetic algorithms.

As an example, the experiments are presented on the large random instance p5000.5. The stopping criterion is the number of generations which is limited to 100. From Tables 1 and 2, one observes that for this instance MSX performs the best, while U-Scan and OB-Scan are much worse than MSX and even Diagonal.

4.1 Evolution of Solution Quality

We first study one of the most important characteristics for the four crossover operators, i.e., the solution gaps to the best known value evolving with the generation iterations, denoted by g_b. g_b is defined as the average value of solution gaps between the best solution in the current population and the best known objective value over 10 independent runs. The smaller is this value, the higher quality the best solution in the population has. Figure 1 shows how this value evolves with the generation iterations for the four crossover operators.

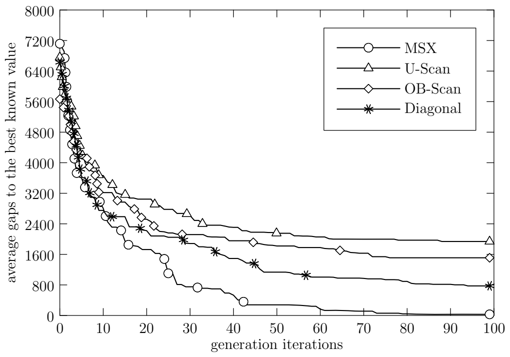


Fig. 1. Best population solution quality evolving with the generation iterations

One observes that at the first generations, the four crossover operators have no clear difference in terms of this criterion. However, with the search progresses, MSX performs much better than other three ones. Note that Diagonal also performs very well and U-Scan is the worst among the four operators. This observation coincides very well with the results reported in Tables 1 and 2, showing the advantage of the crossover operators of MSX and Diagonal, as well as the weakness of U-Scan and OB-Scan for this problem instance.

4.2 Population Entropy and Distance

In our second experiment, we observe the two characteristics of the four multi-parent crossover operators in terms of the population diversity: the *population entropy* vs. the number of generations; the *average solution distance* in the population vs. the number of generations.

The entropy, taking into account the value of each variable in each individual of the population P, is calculated as follows [13]:

$$entropy(P) = \frac{-\sum_{i=1}^{n}\sum_{j=0}^{1}\frac{n_{ij}}{p}log\frac{n_{ij}}{p}}{nlog2} \tag{3}$$

where n is the number of variables and n_{ij} is the number of times the variable x_i is set to j in P. In this definition, $entropy(P) \in [0, 1]$. $entropy(P) = 0$ indicates a population of identical individuals whereas $entropy(P) = 1$ means that all possible assignments are almost uniformly distributed in the population.

The average solution distance in the population is calculated:

$$\bar{d}(P) = \frac{2}{p(p-1)} \sum_{i=1}^{p} \sum_{j=i+1}^{p} d_{ij} \tag{4}$$

where d_{ij} is the Hamming distance between any two solutions $x^{(i)}$ and $x^{(j)}$ in the current population P.

Figure 2 shows how the population entropy (left) and average solution distance of the population (right) evolve with the number of generations. We see that the population diversity measured in terms of these two characteristics is better preserved during the evolution process for MSX and Diagonal than for U-Scan and OB-Scan, especially after the first 60 generations.

Following the spirit of scatter search and path-relinking, an efficient solution combination operator is one that ensures not only high quality solutions but also a good diversity of the population. In other words, the diversification of the population induced by MSX and Diagonal allows the algorithm to benefit from a better exploration of the search space and prevents the population from stagnating in poor local optima.

4.3 Tradeoff between Intensification and Diversification

In this section, we turn our attention to study another important aspect of the memetic algorithms, i.e., the tradeoff between local search and crossover operators. In fact, the performance of the memetic algorithm is influenced by the value of the *depth* of tabu search α. Under a limited computational resource, the *depth* of tabu search reflects the relative proportion of combination operators and tabu search in the algorithm. In this section, we analyze the influence of the parameter α on the performance of the memetic algorithm.

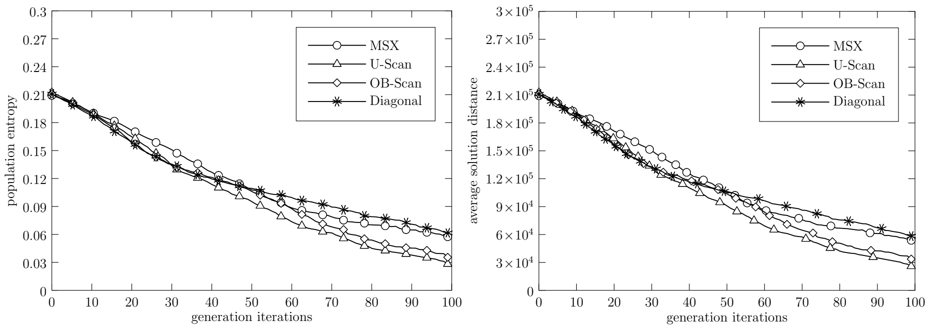


Fig. 2. Comparisons of the four crossover operators in terms of population diversity

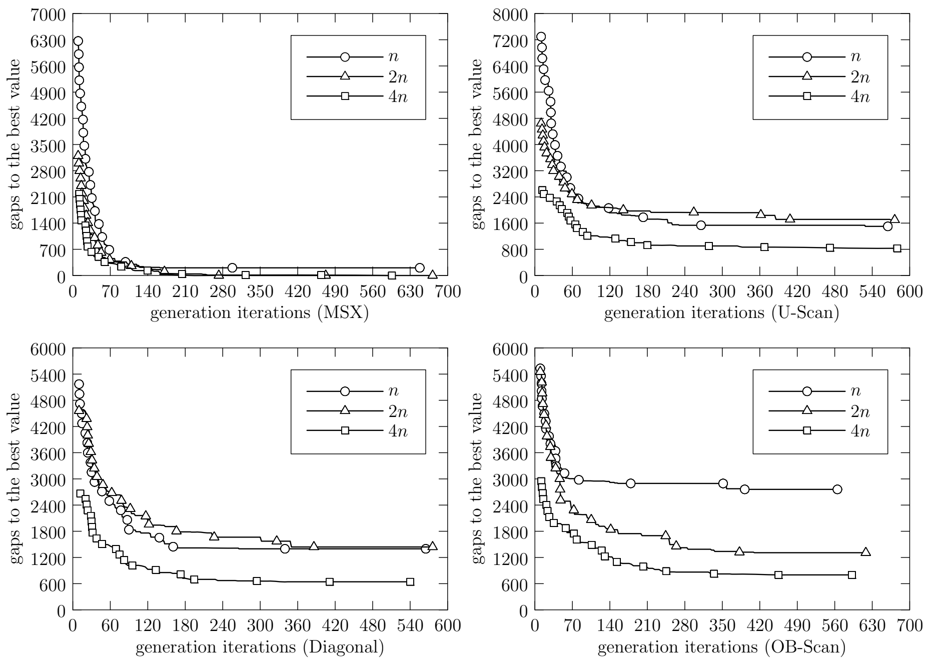


Fig. 3. Tradeoffs between TS and crossover operator

To implement this experiment, we consider 3 different values of the parameter α: $\alpha = n$, $\alpha = 2n$ and $\alpha = 4n$. For each value, we perform 10 independent runs, each run being given 20 minutes CPU time. Figure 3 shows the average evolution of the best solution gaps during the search obtained with these three α values and four crossover operators.

From Figure 3, we first notice that the memetic algorithm performs much worse with $\alpha =$ n and $\alpha =$ 2n than with $\alpha =$ 4n in three cases (Diagonal, U-Scan and OB-Scan), which means that tabu search is an essential part in the memetic algorithm when using these three operators and stronger tabu search can eclipse the role of the crossover. However, when it comes to the MSX crossover operator, one observes that MSX is not really sensitive to various α values, showing that MSX plays a real driving role for the search process. This experiment shows a clear advantage of MSX and the importance of setting an appropriate α value for other crossover operators in order to achieve a desired tradeoff between intensification and diversification.

5 Conclusions

Understanding and explaining the performance of crossover operators within a memetic algorithm is an important topic. In this paper, we presented an attempt to analyze the intrinsic characteristics of four crossover operators for the UBQP problem. To this end, we employed several evaluation indicators to characterize the search capability of a crossover operator. The experimental analysis allowed us to understand to some extent the relative advantages and weaknesses of the four studied crossover operators within the memetic framework.

Acknowledgement

We are grateful for comments by the referees that help us to improve the paper. The work is partially supported by "Pays de la Loire" Region (France) through RaDaPop and LigeRO projects (2009-2012).

References

1. Moscato, P.: Memetic algorithms: a short introduction. In: New Ideas in Optimization, pp. 219–234. Mcgraw-Hill Ltd., Maidenhead (1999)
2. Kochenberger, G.A., Glover, F., Alidaee, B., Rego, C.: A unified modeling and solution framework for combinatorial optimization problems. OR Spectrum 26, 237–250 (2004)
3. Borgulya, I.: An evolutionary algorithm for the binary quadratic problems. Advances in Soft Computing 2, 3–16 (2005)
4. Katayama, K., Tani, M., Narihisa, H.: Solving large binary quadratic programming problems by an effective genetic local search algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), pp. 643–650. Morgan Kaufmann, San Francisco (2000)
5. Lodi, A., Allemand, K., Liebling, T.M.: An evolutionary heuristic for quadratic 0-1 programming. European Journal of Operational Research 119(3), 662–670 (1999)
6. Merz, P., Katayama, K.: Memetic algorithms for the unconstrained binary quadratic programming problem. BioSystems 78, 99–118 (2004)
7. Lü, Z., Hao, J.K., Glover, F.: A study of memetic search with multi-parent crossover for UBQP. In: Cowling, P., Merz, P. (eds.) EvoCOP 2010. LNCS, vol. 6022, pp. 154–165. Springer, Heidelberg (2010)
8. Glover, F., Kochenberger, G.A., Alidaee, B.: Adaptive memory tabu search for binary quadratic programs. Management Science 44, 336–345 (1998)
9. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
10. Eiben, A.E., Raué, P.E., Ruttkay, Z.: Genetic algorithms with multi-parent recombination. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 78–87. Springer, Heidelberg (1994)
11. Ting, C.K.: Design and analysis of multi-parent genetic algorithms, PhD Thesis, Univerity of Paderborn (2005)
12. Palubeckis, G.: Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Annals of Operations Research 131, 259–282 (2004)
13. Fleurent, C., Ferland, J.A.: Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, pp. 619–652 (1996)

A Hybrid Genetic Algorithm for the Traveling Salesman Problem Using Generalized Partition Crossover*

Darrell Whitley, Doug Hains, and Adele Howe

Colorado State University, Fort Collins, CO 80524

Abstract. We present a hybrid genetic algorithm that incorporates the Generalized Partition Crossover (GPX) operator to produce an algorithm that is competitive with the state of the art for the Traveling Salesman Problem (TSP). GPX is respectful, transmits alleles and is capable of tunneling directly to new local optima. Our results show that the hybrid genetic algorithm quickly finds optimal and near optimal solution on problems ranging from 500 to 1817 cities using a population size of 10. It is also superior to Chained-LK given similar computational effort. Additional analysis shows that all the edges found in the globally optimal solution are present in a population after only a few generations in almost every run. Furthermore the number of unique edges in the population is also less than twice the problem size.

Keywords: Traveling Salesman Problem, Generalized Partition Crossover, Hybrid Genetic Algorithm, Chained-LK.

1 Introduction

Chained Lin-Kernighan (Chained LK) [1] is a highly competitive local search algorithm for the Traveling Salesman Problem that uses a carefully designed operator called the "double bridge move." In this paper, we present an evolutionary algorithm that yields better results than Chained LK and that can be shown to possess several desirable characteristics. Our hybrid Genetic Algorithm combines local search with a new recombination operator, Generalized Partition Crossover (GPX), which is a generalization of the Partition Crossover operator (PX) [2].

PX is respectful and transmits alleles: this means that the children of parent solutions are guaranteed to have all edges that are found in both parents (respect) and any edge found in an offspring can also be found in one of the parents (transmits alleles) [3]. Additionally the operator has a property which we call

* This effort was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number FA9550-08-1-0422. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

"tunneling": the offspring of parents that are locally optimal are also locally optimal with high probability.

Partition Crossover (PX) partitions a graph G constructed from the union of two parent tours. If a partition of cost 2 exists in this graph, PX is able to construct two children which are distinct from the parents in $O(n)$ time, where n is the number of vertices in graph G. If there are multiple ways to partition graph G into subgraphs, where each partition has cost 2, PX uses only one of these partitions. But empirically we have found that multiple partitions exist when recombining solutions that are already locally optimal. GPX exploits *all* partitions of cost 2 with no significant increase to the $O(n)$ running time of the original PX operator; the resulting recombination is still respectful and transmits alleles.

When we embed GPX in a hybrid GA, we can show improvements in efficiency and effectiveness over Chained LK. We demonstrate its performance in experiments in which we carefully control computational effort to provide a fair comparison. On small 500 city problems the hybrid Genetic Algorithm often quickly finds the global optimum. It occasionally finds the global optimum on larger problem instances using a very modest amount of computational effort as compared to Chained LK.

More importantly, we analyze the cases when the hybrid GA does *not* easily find the global optimum. We find that the edges in the global optimum which may be missing from the best tour are present in other members of the population. Furthermore, the number of unique edges in the population is relatively small compared to the total number of edges in the cost matrix. In addition, the edges from the global optimum which are missing from the best solutions are predominantly contained within a single subgraph which is the largest "component" of the graph that is being broken apart during the recombination.

2 Generalized Partition Crossover

To recombine two Hamiltonian circuits, Partition Crossover partitions a graph $G = (V, E)$ where V is the set of vertices (i.e., cities) of an instance of a TSP and E is the union of the edges found in two parents. An edge in E can be classified as either a *common* edge or an *uncommon* edge. An edge in E is a *common* edge if it is found in both parents; an edge is an *uncommon* edge if it is found in only one parent.

Whitley et al. [2] prove that if graph G contains at least one partition of cost 2, then it is possible to create at least two offspring in $O(n)$ time which are Hamiltonian circuits distinct from the parents. Figure 1(a) shows a graph G created from two parents. The edges from one parent are represented by solid lines and those from the other by dashed lines. When the common edges are deleted, the graph breaks into 3 subgraphs. There are two partitions of this graph with cost 2 (the heavy dark lines).

The original PX operator constructs two children using only *one* of the partitions of cost 2 in G. PX could construct two children using partition A in

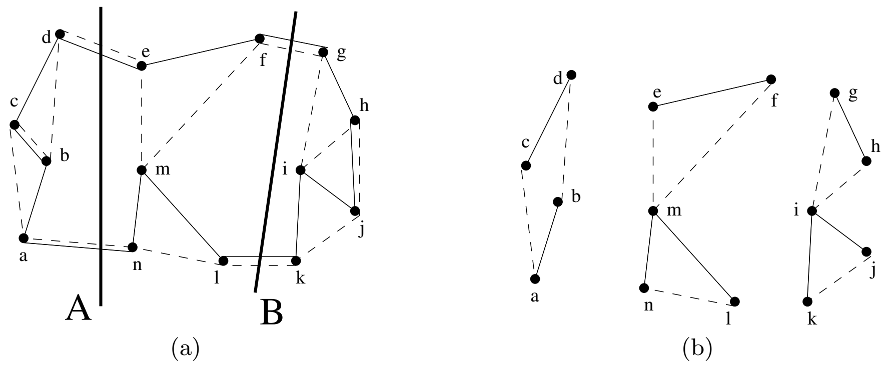


Fig. 1. An example of (a) The graph G created from the union of two parent tours with two partitions shown by the heavy dark lines and (b) The graph G_u, constructed by deleting the common edges between the two parent tours from G

Figure 1 by taking the solid edges from the left of A and the dashed edges from the right, and the second child by taking the dashed edges from the left of A and the solid edges from the right. Two different children would be constructed if PX used partition B in a similar manner. Whitley, Hains and Howe [2] prove that Partition Crossover is respectful and transmits alleles.

Generalized partition crossover (GPX), makes use of all partitions of cost 2 in a single recombination with no significant increase to the $O(n)$ running time of PX. We recombine solutions by creating a subgraph of G, $G_u = (V, E_u)$, where V is the vertex set of the original TSP instance and E_u is the set of uncommon edges found in E. Typically, G_u is made up of multiple disconnected subgraphs. We use Breadth First Search on G_u to find each connected subgraph of G_u; this has $O(n)$ cost, because the degree of any vertex is at most 4, and each vertex is processed only once. Some additional bookkeeping is needed to track which partitions have cost 2 for graph G. When all the partitions of cost 2 are applied, the graph G is broken into k pieces which we will define to be *partition components*; not all connected subgraphs in G_u yield feasible partition components because they may not yield partitions of cost 2.

Figure 1(b) shows an example of the graph G_u created from the graph G shown in Figure 1(a). GPX creates children tours by using the common edges from G and taking either the dashed or solid edges from each of the partition components. The path followed by a tour within a partition component is independent of the path followed by a tour in any other partition component. Thus, within each component, a new tour could follow the path of the "dashed" parent or the "solid" parent.

The GPX Theorem

Let graph G be constructed by unioning the vertices and edge found in two Hamiltonian Circuits for some instance of the TSP. If graph G can be separated into

k partition components of cost 2, then there are $2^k - 2$ distinct offspring; every recombination is both respectful and transmits alleles.

Proof
By construction, all of the common edges connecting partition components are inherited. Within a partition component, a path followed by one of the parents is followed. This also means that all common edges within a partition component are inherited. Therefore the operator is respectful: all common edges are inherited. Generalized Partition Crossover "transmits alleles" because it only uses edges found in the graph G.

Since offspring inherit the common edges; *partitions are also always inherited.* This means we could recombine the parents using a single partition A, then recombine the children again using another partition B to produce grandchildren. By separating all partitions of cost 2, Generalized Partition Crossover is equivalent to iterative applications of Partition Crossover and the offspring must be Hamiltonian circuits.

Let s be a string of k bits, one bit for each partition component. Let bit $s_i = 0$ if a tour follows the path of the "dashed" parent in partition component i. Let bit $s_i = 1$ if a tour follows the path of the "solid" parent in partition component i. Clearly, there are 2^k possible tours (and bit strings) that can be constructed by GPX, but 2 of these represent the parents. Thus, if there are k partition components, there are $2^k - 2$ possible offspring tours that are respectful and that transmit alleles. □

Since the objective function is linear, and all of the offspring are tours, if we make a greedy choice in each partition component by deciding whether the dashed-path or solid-path is shortest, we can also construct the shortest tour possible of the $2^k - 2$ possible offspring by making k greedy choices within each partition component. This is also accomplished in $O(n)$ time.

2.1 GPX with Local Search

We run local search to generate an initial population and to further optimize the offspring. We tested different local search operators. In our previous study [2] looking at randomly chosen local optima produced by the application of 2-opt, we found that Partition Crossover was feasible more than 90 percent of the time. Since PX and GPX are closely related operators, when one is feasible the other is feasible. When randomly chosen local optima are generated by the application of 3-opt, we found that Generalized Partition Crossover is feasible in 100 percent of the cases tested across all of the TSP instances studied in this paper; in more than half of all cases, the offspring are still locally optimal under 3-opt.

To ascertain the number of partition components available to GPX, we recombined 50 random local optima generated using 2-opt [4], 3-opt and Lin-Kernighan search [5] (LK-search). The results are presented in Table 1. The instances rand500 and rand1500 are random Euclidean instances and att532, nrw1379 and u1817 are from the TSPLIB. The number of cities in each instance is indicated by the numerical suffix.

Table 1. Average number of *partition components* used by GPX in 50 recombinations of random local optima found by 2-opt, 3-opt and LK-search

Instance	rand500	att532	nrw1379	rand1500	u1817
2-opt	2.6 ± 0.1	3.3 ± 0.2	3.2 ± 0.2	3.7 ± 0.3	5.0 ± 0.3
3-opt	9.42 ± 0.4	10.5 ± 0.5	11.3 ± 0.5	24.9 ± 0.2	26.2 ± 0.7
LK-search	4.5 ± 0.2	5.3 ± 0.2	5.2 ± 0.3	10.6 ± 0.3	13.3 ± 0.4

Our data indicate 3-opt induces more partition components than 2-opt because it induces more subtours made up of common edges that can be used to partition the graph. The big valley hypothesis [6] supports a strong correlation between the number of common edges shared with the global optimum and the evaluation of a tour. Tours produced by LK-search have even more *total* common edges than 3-opt, but some partitions are now absorbed into common subtours. Thus, there are a smaller number of common subtours (and fewer partitions) than 3-opt but the common subtours become longer for LK.

Sometimes there are more than 20 partition components under 3-opt. Using more than 20 partition components, one recombination is selecting the best of more than 1 million solutions, most of which are local optima. Nevertheless, working with LK-search gets us closer to the global optimum. In the remainder of this paper we will only employ LK-search.

3 The Algorithms: Descriptions and Comparisons

Our hybrid GA is described in Figure 2. The initial population is produced by randomly generating t tours and applying the same LK-search procedure used in step 5. The algorithm was run for a fixed number of generations.

The version of LK-search used is the implementation from Applegate et al. [1] with don't-look bits and uses the default neighborhood list size and search depth. By restricting moves (and clever programming), one iteration of LK-search is faster than a full exploration of the 2-opt neighborhood. The ordering and choice of neighbors is non-deterministic, meaning LK-search may improve upon a tour with subsequent calls until the potential improving moves under LK-search have been exhausted. Only one call to LK-search is done in Step 5.

One of the first questions we considered was whether to use truncation selection (always keep the t best solutions), or to try to preserve diversity by keeping offspring containing edges that are under-represented in the population. When GPX is applied in a greedy fashion, we generate two offspring. The first offspring is the greedy offspring: the shortest path in each partition component is selected. Usually, there are many small partition components, and one large partition component that is typically 20% of the tour. The second offspring also inherits the shortest path in all of the partition components, except for the largest partition component; in this component the offspring inherits the path *not* used by the first greedy offspring. Thus, t recombinations produces $2t$ offspring.

```
Let $P1$ be a randomly generated population of size $t$;
Let $P2$ be a temporary child population of size $t$;
For each member of $P1$: apply LK-search and evaluate;
  1. Attempt to recombine the best tour of $P1$ with the remaining $t-1$ tours
        using GPX; this generates a set of up to $2t$ offspring.
  2. If recombine was not feasible between the best tour and tour $i$, mutate
        tour $i$ using a double bridge move and place in population $P2$;
  3. Place the best solution found so far in population $P2$;
  4. From the set of offspring, select offspring to fill population $P2$;
  5. For each member of population $P2$: apply LK-search and evaluate;
  6. $P1 = P2$; If stopping condition not met, goto 1.
```

Fig. 2. Algorithm for the hybrid GA; the GA is generational, but elitist

We developed a strategy called *diversity selection* that uses an edge weighting function d to quantify the diversity of edges contributed to the population by each tour. For tour s_i in the population,

$$d(s_i) = \sum_{e(j,k) \in s_i} \frac{1}{M(j,k)}$$

where $e(j,k)$ is an edge from city j to k and $M(j,k)$ is the number of times $e(j,k)$ appears in the population. We then retain tours from among the offspring with the highest summed edge diversity, $d(s_i)$.

The use of diversity selection means that the GA must be generational and that offspring replace parents, because parents typically have higher diversity than offspring. In empirical studies, a generational GA using diversity selection consistently produced much better results than keeping the t tours with lowest cost. Thus, we used "diversity selection" (step 4) in the hybrid GA.

We retain the *best* tour found so far (step 3) in the population of offspring. If GPX fails to recombine two tours, we then apply one double-bridge move to tour i where i is *not* the best tour in the population, and directly place this "mutated" tour in the population of offspring (step 2). The remaining members of the offspring population are selected by diversity selection (step 4).

3.1 Comparisons: The Hybrid GA with GPX versus Chained-LK

The hybrid GA was run using a population of 10 tours. The hybrid GA used LK-search as the local search method and Generalized Partition Crossover. We then compare the minimum tour found using the hybrid Genetic Algorithm against the best tour found using Chained Lin-Kernighan. Both methods used exactly the same implementation of LK-search using identical parameters.

Chained Lin-Kernighan is one of the best performing local search heuristics for the TSP [1]. Chained LK applies LK-search to a single tour and then uses a double bridge move [7] to perturb the solution; Chained-LK then reapplies LK-search. Since the population size is 10, the hybrid GA uses 10 applications

Table 2. Average percentage of the cost of the minimum tour found above the globally optimal cost averaged over 500 experiments using Chained LK and the hybrid GA

Generation $\longrightarrow$		5	10	20	50
Instance	Algorithm	60 LK calls	110 LK calls	210 LK calls	510 LK calls
rand500	GA w/ GPX	0.29 ± 0.01	0.17 ± 0	0.1 ± 0	0.05 ± 0
	Chained-LK	0.30 ± 0.01	0.19 ± 0.01	0.13 ± 0	0.09 ± 0
att532	GA w/ GPX	0.29 ± 0	0.18 ± 0	0.12 ± 0	0.07 ± 0
	Chained-LK	0.30 ± 0.01	0.21 ± 0.01	0.13 ± 0	0.08 ± 0
nrw1379	GA w/ GPX	0.63 ± 0	0.48 ± 0	0.34 ± 0	0.23 ± 0
	Chained-LK	0.62 ± 0.01	0.46 ± 0.01	0.32 ± 0	0.19 ± 0
rand1500	GA w/ GPX	0.71 ± 0.01	0.52 ± 0.01	0.36 ± 0	0.22 ± 0
	Chained-LK	0.73 ± 0.01	0.54 ± 0.01	0.39 ± 0.01	0.25 ± 0
u1817	GA w/ GPX	1.61 ± 0.01	1.26 ± 0.01	0.95 ± 0.01	0.63 ± 0.01
	Chained-LK	2.08 ± 0.02	1.61 ± 0.02	1.19 ± 0.01	0.83 ± 0.01

Table 3. The number of times the global optimum is found by each algorithm after 1010 calls to LK-search over 50 experiments

	rand500	att532	nrw1379	rand1500	u1817
Hybrid GA	50/50	26/50	1/50	12/50	1/50
Chained-LK	38/50	16/50	1/50	2/50	0/50

of LK-search each generation; therefore, Chained LK is allowed to do 10 double-bridge moves and apply the LK-search 10 times for every generation that the hybrid GA is allowed to execute. This means that each algorithm is allowed to call LK-search exactly the same number of times. The hybrid GA has the additional cost of recombination, but this cost is $O(n)$ with a small constant and the computation is very small compared to one iteration of LK-search. Furthermore, applying LK-search after a double bridge move is more expensive that applying LK-search after recombination: the double bridge move is a disimproving move, and recombination using GPX solutions is typically an improving move. Thus, the run times are approximately the same. (Exact comparisons of run times are difficult because LK-search is integrated into the Chained-LK code, while the hybrid GA uses a simple but unoptimized interface to call the LK-search.)

Table 2 lists the average percentage of the cost of the minimum tour found compared to the cost of the global optimum for each problem instance. The hybrid GA was allowed to run for 50 generations in these experiments.

After 510 calls each to LK-search, the hybrid GA using GPX yields better results on all of the problems except nrw1379. This is remarkable because the hybrid GA must optimize 10 solutions and the best solution must be optimized 10 times faster than Chained-LK to obtain a better result.

If each algorithm is run longer, the performance of the hybrid Genetic Algorithm is increasingly better than Chained LK. Table 3 shows how many times (out of 50 attempts) that each method finds the global optimum after 1010 calls to LK-search (which is 100 generations for the hybrid GA).

Table 4. Results obtained by running the hybrid GA for only 5 generations and *without* mutation (double-bridge moves). "Global Edges in population" is the number of edges found in the global optimum that are also present in the population. "Global Edges in Minimal Tour" is the number of edges shared in common between the best solution found and the global optimum. "Unique Edges in Population" is the total number of distinct edges found in the population at the end of generation 5. The number of edges in each instance is indicated by the numerical suffix of the instance name.

Instance	Global Edges in Population	Global Edges in Minimum Tour	Unique Edges in Population
rand500	500 ± 0	449.68 ± 1.98	941.56 ± 1.56
att532	532 ± 0	464.1 ± 2.11	979.54 ± 1.47
nrw1379	1378.9 ± 0.04	1162.3 ± 3.44	2709.34 ± 2.25
rand1500	1500 ± 0	1301.02 ± 4.15	2871.9 ± 3.14
u1817	1815.12 ± 0.18	1562.44 ± 3.22	3616.92 ± 4.71

4 The Power of a Population

While it is encouraging that the hybrid GA is able to yield performance that exceeds that of Chained-LK, is this really the best way to exploit the population of solutions that is being generated by the hybrid GA? To explore this question, we ran the hybrid GA again. However, we turned off the mutation operator in step 2. This means that step 4 now selects $t-1$ offspring to place in population $P2$. During the first few generations, recombination is almost always feasible.

We ran 50 trials of the hybrid GA for only 5 generations. At generation 5 we record the minimum tour found, the number of unique edges in the population and the number of edges in the population that also appear in the global optimum. When there is no mutation (i.e., double bridge moves) the hybrid GA converges very fast, but it also loses diversity and gets "stuck" after about 5 generations.

Nevertheless, the hybrid GA is already finding very good solutions after only 5 generations: for rand500 and att532, it found the global optimum in 2 out of 50 trials, using only 50 recombinations and only 50 calls to LK-search. The convergence to the global optimum is extremely fast in these exceptional cases.

As the data shows in table 4, the edges found in the global optimum are all present in the population in the majority of the runs. On instances rand500, att532 and rand1500 all of the edges found in the global optimum were also in the population on every single run. The population therefore contains all of the edges needed to construct the globally optimal solution after only 5 generations.

Furthermore, the results for all of the TSP instances show that the total number of unique edges in the population was always less than $2n$ after 5 generations. Assume that we merge all 10 members of the population after 5 generations into a single graph. We can now search this reduced graph for a minimal Hamiltonian circuit. The search space is dramatically smaller than that of the original TSP instances. This means that the optimization problem has been reduced to

Table 5. Percentages were averaged over 50 trials. These results were captured during recombination during the 5th generation. Common Edges can appear inside of components, or between components of Graph G_u. Uncommon edges appear only inside of components of G_u.

	att532	u1817
Common Edges also found in the global optimum	406.75	1407.90
Uncommon edges in largest component also found in global optimum	87.39	268.62
Uncommon in all other components also found in global optimum	0.04	0.96
Total edges in the largest component	102.11	289.12

finding the minimal Hamiltonian Circuit of length n in a graph with only $2n$ edges.

4.1 Where Are the Global Edges?

We next look at those edges that appear in the global optimum, but which do not appear in the best tour in the population. We already know that typically all of the edges found in the global optimum are present in the population after 5 generations. Since we recombine the best tour with all members of the population, those edges that are shared with the global but not found in the minimal tour must be classified as uncommon by GPX and will appear in the graph G_u during at least one recombination.

Because of the way GPX performs crossover, edges that appear in the same partition component cannot be chosen on an individual level. Either all the edges from one parent are chosen from that component or all the edges from the other parent are chosen.

We want to determine if the uncommon globally optimal edges are spread out among different partition components in G_u or if they appear in the same component. If a majority of the uncommon global edges appear in the same component, then this means it will be impossible for GPX (working without any form of mutation) to reassemble these edges and reach the global optimum.

We looked at the trials from the previous experiments and found in the majority of recombinations the uncommon globally optimal edges fell into a single partition component which was larger than the rest. In table 5 we report the number of uncommon globally optimal edges which fell into this large partition component, the size of this component, and the number of uncommon global edges which fell into other components; we also report the number of shared common global edges observed during recombination. Results for two instances, att532 and u1817, are shown in table 5.

As can be seen from Table 5, the majority of edges that are not found in the best solution but that are found in the global optimal solution appear as uncommon edges that are largely contained in the largest component during the recombination process. Nevertheless, most of the edges that are found in the globally optimum actually appear as *common* edges (the first row in Table 5) during recombination, meaning these edges will be passed onto the children.

5 Conclusions and Future Work

A new recombination operation has been developed for the TSP. GPX is a generalization of the previously described PX operator. Both operators are respectful and transmits alleles. GPX in a hybrid GA with LK-search is capable of finding better tours than Chained LK using double bridge moves. Additionally, we find that the hybrid genetic algorithm using GPX and LK-search is capable of finding globally optimal solutions in a relatively small number of generations.

Additional analysis shows that all the edges found in the globally optimal solution are present in a population after only a few generations in almost every case. Furthermore, the number of unique edges in the population is also less than twice the problem size.

When critical edges are concentrated in a single partition component, GPX is not able to "re-assort" these edges. However, this represents a challenge as well as an opportunity. Instead of needing to optimize over the entire search space, effort can be focused on optimizing a small subregion of the search space. Future research will examine how best to exploit this knowledge.

References

1. Applegate, D., Cook, W., Rohe, A.: Chained Lin-Kernighan for large traveling salesman problems. INFORMS Journal on Computing 15(1), 82–92 (2003)
2. Whitley, D., Hains, D., Howe, A.: Tunneling between optima: partition crossover for the traveling salesman problem. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 915–922. ACM, New York (2009)
3. Radcliffe, N., Surry, P.: Fitness variance of formae and performance predictions. In: Whitley, D., Vose, M. (eds.) FOGA - 3, pp. 51–72. Morgan Kaufmann, San Francisco (1995)
4. Croes, G.: A method for solving traveling-salesman problems. Operations Research, 791–812 (1958)
5. Lin, S., Kernighan, B.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research, 498–516 (1973)
6. Boese, K.D., Kahng, A.B., Muddu, S.: A new adaptive multi-start technique for combinatorial global optimizations. Operations Research Letters 16, 101–113 (1994)
7. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: A case study in local optimization. In: Aarts, E.H.L., Lenstra, J. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. John Wiley and Sons Ltd., Chichester (1997)

A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-model

Saúl Zapotecas Martínez* and Carlos A. Coello Coello**

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, México
saul.zapotecas@gmail.com, ccoello@cs.cinvestav.mx

Abstract. The development of multi-objective evolutionary algorithms (MOEAs) assisted by meta-models has increased in the last few years. However, the use of local search engines assisted by meta-models for multi-objective optimization has been less common in the specialized literature. In this paper, we propose the use of a local search mechanism which is assisted by a meta-model based on support vector machines. The local search mechanism adopts a free-derivative mathematical programming technique and consists of two main phases: the first generates approximations of the Pareto optimal set. Such solutions are obtained by solving a set of aggregating functions which are defined by different weighted vectors. The second phase generates new solutions departing from those obtained during the first phase. The solutions found by the local search mechanism are incorporated into the evolutionary process of our MOEA. Our experiments show that our proposed approach can produce good quality results with a budget of only 1,000 fitness function evaluations in test problems having between 10 and 30 decision variables.

1 Introduction

Evolutionary algorithms (EAs) have been successfully adopted for solving multi-objective optimization problems (MOPs) in a wide variety of engineering and scientific problems [1]. However, in real-world applications is common to find objective functions which are very expensive to evaluate (computationally speaking). This considerably limits the application of EAs. This has motivated the development of numerous strategies for reducing the number of fitness function evaluations when using EAs [2]. From such strategies, the use of meta-models has been one of the most commonly adopted. Several authors have reported the use of surrogate models which aim to model a function by means of a simple linear regression, polynomial regression or by more elaborated models such as Artificial Neural Networks (ANNs), Radial Basis Functions (RBFs), Support Vector Machines (SVMs), Gaussian processes (also known as Kriging), among others.

* The first author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN.

** The second author acknowledges support from CONACyT project no. 103570.

Most of this work, however, focuses on single-objective optimization problems, and relatively few refer to multi-objective optimization tasks. In this paper, we present a strategy which combines an approximation function model (based on support vector machines) combined with a local search engine (which adopts a non-gradient mathematical programming technique) and a multi-objective evolutionary algorithm (MOEA). Our goal was to reduce the number of fitness function evaluations, while still producing reasonably good approximations of the Pareto optimal set.

The remainder of this paper is organized as follows. In Section 2, we present a brief survey of previous related work reported in the specialized literature. In Section 3, we describe in detail our proposed approach. In Section 4, we show the results of our proposal. Finally, in Section 5 we present our conclusions and provide some possible paths for future research.

2 Previous Related Work

The incorporation of meta-models in EAs to approximate the real fitness function of a problem, aiming to reduce the total number of fitness evaluations performed has been studied by several researchers. However, most of these approaches have been developed to deal with single-objective optimization problems (see for example [2]). Here, however, our review of previous work will focus only on MOEAs.

Ong et al. [3] proposed an approach that incorporates surrogate models for solving computationally expensive problems with constraints. The authors used a combination of a parallel EA coupled with sequential quadratic programming in order to find optimal solutions of an aircraft wing design problem. A local surrogate model based on RBFs is the strategy adopted to approximate the objective and the constraint functions.

Emmerich and Naujoks [4] proposed several metamodel-assisted MOEAs. Gaussian field (Kriging) models fitted by results from previous evaluations are used in order to pre-screen candidate solutions and decide whether they should be evaluated or rejected. Three different rejection mechanisms were proposed and integrated into MOEA variants (NSGA-II and ϵ-MOEA).

In [5], Knowles proposed "ParEGO", which consists of a hybrid algorithm based on a single optimization model (EGO) and a Gaussian process, which is updated after each function evaluation, coupled to an evolutionary algorithm. EGO is a single-objective optimization algorithm that uses Kriging to model the search landscape from the previously visited solutions.

Isaacs et al. [6] proposed a MOEA with spatially distributed surrogate models based on RBFs. In this approach, the objective functions are analyzed with their actual values for the initial population and then periodically, at every few generations. The approach maintains an external archive of these actual objective function values, since these values are used to train the surrogate models. The data points are divided into multiple partitions using clustering techniques (the k-means algorithm). The surrogate model is built for each partition using a fraction of the points lying in that partition. The rest of the points in the partition are used as validation data to decide the prediction accuracy of the surrogate model.

Finally, Georgopoulou and Giannakoglou [7] proposed a metamodel-assisted memetic algorithm for multi-objective optimization. This approach uses several RBFs, each of them corresponding to a small portion of the search space. The local search mechanism uses a function which corresponds to an ascent method that incorporates gradient values provided by the metamodels. Each RBF is retrained by considering the current offspring, parent and elite populations. The performance of this approach was evaluated with three benchmark problems and a combined cycle power plant problem. This approach outperformed a conventional MOEA in all the test problems adopted.

3 Our Proposed Approach

In this section, we present a new *Multi-Objective Meta-Model Assisted Memetic Algorithm* (MO-MAMA) which incorporates a local search mechanism based on non-gradient mathematical programming techniques. Our algorithm is characterized by using an approximation model based on support vector regression [8]. Additionally, our approach adopts an external archive $\mathcal{A}$ and a solutions set $\mathcal{R}$ (obtained by the local search mechanism) to create the offspring population in the EA. The meta-model is trained with the set $\mathcal{D}$, which consist of all the solutions evaluated with the real fitness function values obtained up to the current generation. The details of this approach are described next.

3.1 The Multi-objective Meta-model Assisted Memetic Algorithm

Initially, we create a sample $\mathcal{S}$ of size $2N$ (where N is the population size) which is randomly distributed in the search space using the Latin hypercube sampling method [9]. The initial population $\mathcal{P}_0$ is defined by N solutions randomly chosen from $\mathcal{S}$. Then, the normal evolutionary process of the MOEA is carried out. The proposed approach uses the current population $\mathcal{P}_t$, a set of solutions $\mathcal{R}_t$ (obtained by the local search mechanism) and a (bounded) external archive $\mathcal{A}_t$ (defined by all the nondominated solutions found throughout the evolutionary process) to create the offspring population $\mathcal{Q}_t$ at generation t. The next population $\mathcal{P}_{t+1}$ is obtained by selecting N individuals from $\mathcal{P}_t \cup \mathcal{Q}_t$ according to Pareto ranking. This procedure is called *SelectNextPopulation* in the algorithm of Figure 1, which shows the complete scheme of our proposed MO-MAMA. Its details are explained next.

Archiving Solutions: Our algorithm uses an external archive $\mathcal{A}$ which stores all the nondominated solutions found at each generation of the MOEA. The archive $\mathcal{A}$ is bounded according to the population size. Thus, the maximum number of solutions in $\mathcal{A}$ is N. Since we are interested in obtaining a well-distributed set of solutions along the Pareto front, we adopted a strategy based on the k-means algorithm [10]. At each generation, the archive $\mathcal{A}$ is updated with the new nondominated solutions found in the population $\mathcal{P}$. If the number

```
// $t_{max}$ = maximum number of generations
1. $t = 0, \mathcal{A} = \emptyset$;
2. Generate $\mathcal{S}$ of size $2N$ // using the Latin Hyper-cubes method;
3. $Evaluate(\mathcal{S})$;            // using the real fitness function
4. $\mathcal{P}_t = \{x_i \in \mathcal{S}\}$ such that: $x_i$ is randomly chosen from $\mathcal{S}$ and $|\mathcal{P}_t| = N$;
5. $\mathcal{R}_t = \mathcal{S} \setminus \mathcal{P}_t$;
6. $\mathcal{A} = UpdateArchive(\mathcal{R}_t, \mathcal{A})$;
7. $\mathcal{D} = \mathcal{S}$;
8. while $(t < t_{max})$do
9.     $\mathcal{A} = UpdateArchive(\mathcal{P}_t, \mathcal{A})$;
10.    $\mathcal{Q}_t = CreateOffspring(\mathcal{P}_t, \mathcal{R}_t, \mathcal{A}_t)$;
11.    $Evaluate(\mathcal{Q}_t)$;      // using the real fitness function
12.    $\mathcal{D} = \mathcal{D} \cup \mathcal{Q}_t$;
13.    $\mathcal{P}_{t+1} = SelectNextPopulation(\mathcal{P}_t, \mathcal{Q}_t)$;
14.    $\mathcal{R}_{t+1} = SurrogateLocalSearch(\mathcal{P}_t, \mathcal{A})$;
15.    $t = t + 1$;
16. end while
```

Fig. 1. Main algorithm of our proposed MO-MAMA

of solutions is greater than N, then we define k-means ($k = N$) from $\mathcal{A}$. In this way, the archive is updated with the nearest solutions to each mean. This procedure is called *UpdateArchive* in the algorithm of Figure 1.

Generating Offspring Population: We consider the set $\mathcal{D}$ as the set of all solutions obtained by the MOEA, and $\mathcal{R}$ as the set of solutions obtained by the local search mechanism. Furhermore, we assume that our approach will eventually converge to the Pareto optimal set (or, at least, to a reasonably good approximation of it). Therefore, in the last generations of the algorithm, a well-distributed sample of the Pareto set is achieved and maintained in $\mathcal{D}$. For this, the improvement mechanism (which approximates solutions to the Pareto optimal set) generates solutions, which, when evaluated into the meta-model, correspond to good approximations of the real fitness values. Furthermore, since the set $\mathcal{R}$ is the result of an improvement procedure, we consider that both the $\mathcal{R}$ set and the $\mathcal{A}$ set (the nondominated set) have solutions of similar quality. Based on the previous discussion, crossover takes place between each individual of the population $\mathcal{P}$ (the current population) and an individual which can be chosen from either $\mathcal{R}$ or $\mathcal{A}$. Therefore, we define the parents for the crossover operator according to the following procedure:

$$\begin{aligned} parent^1 &= x_i \in \mathcal{P} \quad \forall i = 1, \ldots, N \\ parent^2 &= \begin{cases} y \in \mathcal{R}, & \textbf{if } \left(g < 1 - \frac{|\mathcal{A}|}{2N}\right) \\ y \in \mathcal{A}, & \textbf{otherwise} \end{cases} \end{aligned} \tag{1}$$

where g is a uniformly distributed random number within $(0, 1)$ and y is a solution randomly chosen from $\mathcal{A}$ or $\mathcal{R}$. Clearly, when the archive pool $\mathcal{A}$ is full, $|\mathcal{A}| = N$ and equation (1) guarantees to choose a solution from either $\mathcal{R}$ or $\mathcal{A}$ (both have the same probability). The mutation operator is applied (with a certain probability) to each child generated by the crossover operator. In this

```
// $\mathcal{P}$ = current population
// $\mathcal{R}$ = set of solutions obtained by the local search mechanism
// $\mathcal{A}$ = external archive
1. $\mathcal{Q} = \emptyset$;
2. forall $(x \in \mathcal{P})$do
3.     $parent^1 = x$;
4.     Define $parent^2$ according to equation (1);
5.     Generate $child^1$ and $child^2$ performing $SBX(parent^1, parent^2)$;
6.     $y^1 = PBM(child^1)$ and $y^2 = PBM(child^2)$;
7.     $\mathcal{Q} = \mathcal{Q} \cup \{y^1, y^2\}$;
8. end forall
9. return $\mathcal{Q}$;
```

Fig. 2. Creating the offspring population ($CreateOffspring(\mathcal{P}, \mathcal{R}, \mathcal{A})$)

work, we adopted the genetic operators from the NSGA-II [11] (Simulated Binary Crossover (SBX) and Parameter-Based Mutation (PBM)). Figure 2 shows the complete procedure for creating the offspring population.

Local Search Mechanism: The main goal of the local search mechanism incorporated into our meta-model is to find new solutions nearby the solutions provided by the MOEA (such solutions should be at least nondominated with respect to the current and previous populations). While the local search engine explores promising areas into the meta-model, the MOEA performs a broader exploration of the search space. All this procedure is called *SurrogateLocalSearch* within the algorithm of Figure 1.

Approximating Solutions: There exist several mathematical programming methods designed for solving multi-objective optimization problems (see e.g., [12,13]). Here, we are interested in solving the *weighted Tchebycheff problem* which is of the form:

$$\min_{x \in \mathbb{R}^n} \max_{i=1,\ldots,k} \{w_i |f_i(x) - z_i^*|\} \tag{2}$$

where z^* denotes the ideal vector, w is a vector in $\mathbb{R}^k$ such that $\mathbf{0} < w$ and $\sum_{i=1}^k w_i = 1$ (a convex combination of weights). It is well known that, for each Pareto optimal point there exists a weighting vector $\mathbf{0} < w \in \mathbb{R}^k$ such that it is the optimum solution of (2). Unfortunately, if the solution of the Tchebycheff problem is not unique, the solutions generated will be weakly Pareto optimal. In order to identify the Pareto optimal solutions, the following *augmented weighted Tchebycheff* problem is suggested:

$$\min_{x \in \mathbb{R}^n} \max_{i=1,\ldots,k} \{w_i |f_i(x) - z_i^*|\} + \rho \sum_{i=1}^{k} |f_i(\boldsymbol{x}) - z_i^\star| \tag{3}$$

where ρ is a sufficiently small positive scalar and z^* represents the utopian vector.

Initially, a set of n_w well-distributed weighted vectors $W \subset \mathbb{R}^k$ is defined (for this task, we use the method proposed by Zhang and Li [14]). The approximate solutions to the Pareto optimal set are obtained by solving the Tchebycheff problem for each weighted vector. For each weighted vector $w_j \in W$, a set of solutions λ_j is found, which consists of all the solutions evaluated so far into the meta-model by solving the above Tchebycheff problem. The utopian vector z^* is constructed with the minimum of each objective function at the current generation. Moreover, here, we use the well-known *pattern search* (or Hooke and Jeeves) algorithm [15], in order to solve each Tchebycheff problem. Clearly, all the candidate solutions are evaluated into the surrogate model. The initial search point x_s for solving the first problem corresponding to the weighted vector w_1, is defined according to the next equation:

$$x_s = x^* \in \{\mathcal{P}_t \cup \mathcal{A}\}, \text{ such that } x^* \text{ minimizes equation (3)} \tag{4}$$

where $\mathcal{P}_t$ and $\mathcal{A}$ are the population and the external archive at the current generation, respectively. The remaining sets λ_j $(j = 2, \ldots, n_w)$ are obtained by solving the Tchebycheff problem for the weighted vector w_j. The initial search point for obtaining λ_j is given by the decision vector which minimizes the Tchebycheff problem for the weighted vector w_{j-1}. Therefore, we define the set Λ as the union of all the sets λ found by solving the n_w Tchebycheff problems, that is:

$$\Lambda = \bigcup_{j=1}^{n_w} \lambda_j \tag{5}$$

Generating New Solutions: We consider Λ to be the set of solutions found by the above process. Furthermore, we consider:

$$P(\exists p \in \mathbb{R}^n : ||q^* - p|| < \delta \text{ and } q^* \not\prec p) = 1 \tag{6}$$

for any small $\delta \in \mathbb{R}_+$. Here, q^* is at least a locally nondominated solution. That is, the probability that p is nondominated with respect to q^* is equal to one, which implies that p is also nondominated. We generate more approximate solutions using an evolutionary algorithm available within the meta-model. The differential evolution (DE) [16] algorithm with a *DE1/rand/bin* strategy is adopted for this task. Furthermore, the following dominance rule is used to select the new individuals for the next generation:

$$x_{i,g+1} = \begin{cases} x^*_{i,g} & \textbf{if } (x^*_{i,g} \prec x_{i,g}) \\ & \textbf{or } (x^*_{i,g} \text{ and } x_{i,g} \text{ are nondominated}) \\ x_{i,g} & \textbf{otherwise} \end{cases} \tag{7}$$

where x_i is a solution in the current population, x^* is the test vector and g is the current iteration of the DE algorithm. For more details about DE see [17]. The initial population is given by $\mathcal{G}_0 = \Lambda$. Each new individual $x_{i,g+1}$ is stored (or not) in an external archive $\mathcal{L}$ according to the dominance rule. The archive strategy can make that the set of solutions $\mathcal{L}$ increases or decreases its size. Given the probability defined by equation (6), we generate more nondominated solutions from $\mathcal{L}$. Thus, the next population for the DE algorithm is defined by $\mathcal{G}_{g+1} = \mathcal{L}$.

Since all the solutions in the archive $\mathcal{L}$ are nondominated, we can say that the algorithm has converged (at least to a local Pareto front) when it has obtained N different nondominated solutions from the evolutionary process. That is:

$$\textbf{if } |\mathcal{L}| = N \textbf{ then} \qquad \text{stop the DE algorithm} \tag{8}$$

Therefore, the solutions set $\mathcal{R}$ obtained by our local search mechanism is given by $\mathcal{R} = \mathcal{L}$. However, this stopping criteria is not always satisfied. Thus, we can define the $\mathcal{R}$ set by selecting N individuals from $\Lambda \cup \mathcal{L}$ using Pareto ranking after a certain number of iterations.

4 Comparison of Results

In order to assess the performance of our proposed approach, we compare it with respect to NSGA-II [11]. The test problems adopted are the ZDT (Zitzler-Deb-Thiele) test suite [18] (except from ZDT5, which is a binary test problem). We adopted three performance measures to assess our results: Inverted Generational Distance ($\mathcal{IGD}$) [19], Spacing ($\mathcal{S}$) [20] and the Set Coverage ($\mathcal{SC}$) [18].

4.1 Experimental Setup

As indicated before, we compared our proposed approach with respect to the NSGA-II. For each MOP, we performed 25 independent runs with each approach. The parameters used in the algorithms are shown below.

Since our approach adopts the same genetic operators included in the NSGA-II (SBX and PBM), we adopted the same parameter values for these operators in both algorithms, that is: crossover index $\eta_c = 15$ and mutation index $\eta_m = 20$. Furthermore, for both algorithms we used: crossover probability $P_c = 1.0$, mutation probability $P_m = \frac{1}{n}$ (where n represents the number of decision variables of the MOP) and a population size $N = 100$.

The Hooke-Jeeves algorithm was implemented with: $\delta_i = \frac{up_i - low_i}{2}$ (up_i and low_i are the upper and lower bounds of the i^{th} decision variable component, respectively), the reduction factor was set to $\alpha = 2$ and $\varepsilon = 1 \times 10^{-3}$. The differential evolution algorithm was implemented using a weighting factor $F = 0.5$ and a crossover constant $CR = 1.0$. Finally, for the approximation phase, we set $n_w = 5$ (which is equal to 5% of the population size) as the number of weighted vectors which define the number of Tchebycheff problems. We should consider that more weight vectors implies more local search and with this, greater computational effort.

4.2 Discussion of Results

Our results are summarized in Tables 1 to 3. Each table displays both the average (showing the best results in **boldface**) and the standard deviation (σ) of

Table 1. Results for $\mathcal{IGD}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**0.000068** (0.000036)	0.055665 (0.005467)
ZDT2	**0.000186** (0.000400)	0.065110 (0.006909)
ZDT3	**0.000965** (0.000028)	0.055609 (0.006253)
ZDT4	0.151272 (0.033721)	**0.143483** (0.022090)
ZDT6	**0.000483** (0.000210)	0.023264 (0.001469)

Table 2. Results for $\mathcal{S}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**0.020216** (0.013534)	1.285481 (0.848476)
ZDT2	**0.034953** (0.062620)	1.690465 (1.171313)
ZDT3	**0.022872** (0.007906)	1.455995 (1.258330)
ZDT4	**6.522628** (8.981982)	19.108133 (24.636678)
ZDT6	0.415136 (0.320719)	**0.226461** (0.162008)

Table 3. Results for $\mathcal{SC}$ metric (MO-MAMA vs NSGA-II)

MOP	MO-MAMA *average* (σ)	NSGA-II *average* (σ)
ZDT1	**1.000000** (0.000000)	0.000000 (0.000000)
ZDT2	**0.979200** (0.048656)	0.000000 (0.000000)
ZDT3	**1.000000** (0.000000)	0.000000 (0.000000)
ZDT4	0.673600 (0.093590)	**0.684000** (0.073103)
ZDT6	**1.000000** (0.000000)	0.000000 (0.000000)

each performance measure, for each of the test problems adopted. Each run was restricted to $1,000$ fitness function evaluations. These results clearly show that our proposed approach (MO-MAMA) outperformed the NSGA-II in most of the test problems adopted (except for ZDT4), not only with respect to $\mathcal{IGD}$ but also with respect to $\mathcal{SC}$. It is worth noticing that the NSGA-II performed better with respect to $\mathcal{S}$, which indicates that it produced solutions with a better distribution. However, a better distribution of solutions is relevant only when a good approximation of the true Pareto front has been achieved. Since in our case, we were emphasizing efficiency (i.e., only a fairly limited number of fitness function evaluations was allowed). Furthermore, according to the Wilcoxon rank-sum test [21], our MO-MAMA is significantly better than NSGA-II over the $\mathcal{IGD}$ metric (which is the most important metric that we considered in this work) in most of the adopted test problems (except for ZDT4) with a

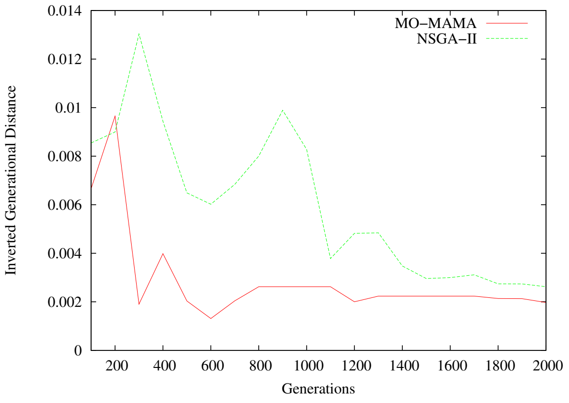


Fig. 3. Convergence for $\mathcal{IGD}$ metric over ZDT1 problem using $2,000$ real fitness function evaluations

significance level of 0.05. In the other hand, for the ZDT4 problem the Wilcoxon test did not show a significant variation. Therefore, we considered these results to be satisfactory. Finally, a picture of the convergence for the $\mathcal{IGD}$ metric in the ZDT1 problem is shown in Figure 3.

5 Conclusions and Future Work

We have proposed a multi-objective memetic algorithm assisted by support vector machines, with the aim of performing an efficient exploration of the search space. Our local search engine was based on a weighted Tchebycheff function and the Hooke-Jeeves method was adopted as our minimizer for each problem defined by each weighted vectors under consideration. Our proposed approach was found to be competitive with respect to the NSGA-II over a set of test functions taken from the specialized literature, when performing only $1,000$ fitness function evaluations.

As part of our future work, we plan to use our approach in problems having more objectives (three or more) and we aim to experiment with other search engines (e.g., with multi-objective scatter search [22]). The introduction of alternative approaches to improve the uniform distribution of our solutions as well as the use of more difficult test problems (e.g., the Deb-Thiele-Laumanns-Zitzler (DTLZ) test problems [23] and the Walking-Fish-Group (WFG) test problems [24]) is also part of our future work. Finally, we are also interested in testing our approach with real-world problems having computationally expensive objective functions, and that is indeed part of our ongoing research.

References

1. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
2. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9(1), 3–12 (2005)
3. 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)
4. Emmerich, M.T.M., Naujoks, B.: Metamodel-assisted multiobjective optimisation strategies and their application in airfoil design. In: Adaptive Computing in Design and Manufacture VI, Berlyn, Germany, pp. 249–260. Springer, Heidelberg (2004)
5. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)
6. Isaacs, A., Ray, T., Smith, W.: An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS (LNAI), vol. 4828, pp. 257–268. Springer, Heidelberg (2007)
7. Georgopoulou, C.A., Giannakoglou, K.C.: A multi-objective metamodel-assisted memetic algorithm with strengthbased local refinement. Engineering Optimization 41(10), 909–923 (2009)

8. Vapnik, V., Golowich, S.E., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. In: Advances in Neural Information Processing Systems, vol. 9, pp. 281–287. MIT Press, Cambridge (1997)
9. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21(2), 239–245 (1979)
10. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
11. 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)
12. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston (1999)
13. Hillermeier, C.: Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach. Birkhäuser, Basel (2000)
14. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation 11(6), 712–731 (2007)
15. Hooke, R., Jeeves, T.A.: "Direct search" solution of numerical and statistical problems. J. ACM 8(2), 212–229 (1961)
16. Storn, R.M., Price, K.V.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, Berkeley, CA (1995)
17. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution. A Practical Approach to Global Optimization. Springer, Berlin (2005), ISBN 3-540-20950-6
18. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
19. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
20. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts (1995)
21. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin 1(6), 80–83 (1945)
22. Nebro, A.J., Luna, F., Alba, E., Dorronsoro, B., Durillo, J.J., Beham, A.: AbYSS: Adapting Scatter Search to Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 12(4), 439–457 (2008)
23. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, pp. 105–145. Springer, USA (2005)
24. 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)

Theoretically Investigating Optimal μ-Distributions for the Hypervolume Indicator: First Results for Three Objectives

Anne Auger[1], Johannes Bader[2], and Dimo Brockhoff[1]

[1] TAO Team, INRIA Saclay, LRI, Paris Sud University, 91405 Orsay Cedex, France
firstname.lastname@inria.fr

[2] Computer Engineering and Networks Lab, ETH Zurich, 8092 Zurich, Switzerland
johannes.bader@tik.ee.ethz.ch

Abstract. Several indicator-based evolutionary multiobjective optimization algorithms have been proposed in the literature. The notion of *optimal μ-distributions* formalizes the optimization goal of such algorithms: find a set of μ solutions that maximizes the underlying indicator among all sets with μ solutions. In particular for the often used hypervolume indicator, optimal μ-distributions have been theoretically analyzed recently. All those results, however, cope with bi-objective problems only. It is the main goal of this paper to extend some of the results to the 3-objective case. This generalization is shown to be not straight-forward as a solution's hypervolume contribution has not a simple geometric shape anymore in opposition to the bi-objective case where it is always rectangular. In addition, we investigate the influence of the reference point on optimal μ-distributions and prove that also in the 3-objective case situations exist for which the Pareto front's extreme points cannot be guaranteed in optimal μ-distributions.

1 Introduction

Several evolutionary multiobjective optimization (EMO) algorithms have been proposed to tackle multiobjective optimization problems. Among them, the indicator-based algorithms are the most recent developments [17,6,13]. These algorithms often explicitly optimize a unary quality indicator which maps a set of solutions to a single real value. This not only allows to decouple preference articulation from the search algorithm [17] but also transforms the multiobjective problem into a single-objective one: the goal is no longer to find or approximate the so-called Pareto front, but to find a solution set of fixed size (typically the population size μ) that maximizes the indicator. Therefore, it is important to characterize these solution sets to understand the optimization goal implicitly defined by a given indicator. In particular when benchmarking algorithms on certain test functions, it is highly useful to know the largest possible indicator value achievable with μ points. Throughout the paper, and in line with [2], we use the term *optimal μ-distribution* for those sets of μ solutions optimizing a given indicator.

One of the most often used quality indicators within indicator-based EMO algorithms is the hypervolume indicator or $\mathcal{S}$-metric which maps a set of solutions to the *size of the objective space covered* [18]. It has the nice property of being a refinement of the

Pareto dominance relation [19] which implies that the optimal μ-distributions contain only solutions that are mapped to the Pareto front [10]. The question of *how* the optimal μ-distributions are spread over the Pareto front, interestingly, has only gained attention recently. Besides specific results on optimal μ-distributions in the case of linear Pareto fronts [9,5], optimal μ-distributions have been theoretically investigated in more detail in [2,1] for bi-objective problems. The main results are an exact characterization of optimal μ-distributions for problems with arbitrary linear Pareto fronts and a limit result in terms of a density for general front shapes that can be described by a continuous and differentiable function f. The density result proves that the empirical density of points converges to a density proportional to the square root of the negative of the first derivative of the front. In other words, it is only the slope of the front which determines how the points that maximize the hypervolume indicator are distributed—independent of the second derivative, i.e., whether the front is convex or concave. It has also been proven in [2] that for certain types of fronts, no finite reference point of the hypervolume indicator allows to have the extreme points in the optimal μ-distribution; for the remaining cases, it has been shown where to place the reference point such that the extremes are included. Later, the relation between optimal μ-distributions for the hypervolume indicator and the approximation ratio has been investigated theoretically as well [11,7]. However, also in these studies, the results are restricted to only two objectives. The main reason why almost no results about optimal μ-distributions for 3-objective problems are known[1] is that the geometry of the hypervolume becomes more complicated in higher dimensions. We will see later on that, e.g., the hypervolume contributions of single points are not anymore simple rectangles or cuboids if 3-objective problems are considered and that all solutions can have an influence on the optimal placement of one point—in comparison to the local property proven in [2] for bi-objective problems.

Contributions of this paper. In this paper, we present for the first time theoretical results about optimal μ-distributions for the hypervolume indicator for more than 2 objectives, in contrast to [2,1,11,7] where only bi-objective problems were tackled. Besides fundamental results on the existence and the monotonicity of optimal μ-distributions (Sec. 3), we prove fundamental, yet often not obvious statements about the shape of the hypervolume contribution of a single solution (Sec. 4) and investigate their implications on optimal μ-distributions—in particular on the influence of the reference point (Sec. 5). More specifically, we prove that situations exist (and characterize them) for which the extreme points of the Pareto front will never be contained in an optimal μ-distribution for 3-objective problems which covers the results for the bi-objective case of [2]. The results show in particular, that the investigation of optimal μ-distributions is, indeed, more difficult for 3-objective problems than in the case of 2 objectives.

2 Preliminaries

Without loss of generality (w.l.o.g.), we consider minimization problems where the vector-valued objective function is defined as $\mathcal{F}\colon X \to \mathbb{R}^k$ and k is the number of objectives. In this paper, $k = 3$ most of the time. We say $\mathcal{F}$ maps a solution $x \in X$

[1] The only exception is a conjecture in [3] about the influence of the dimension on the density.

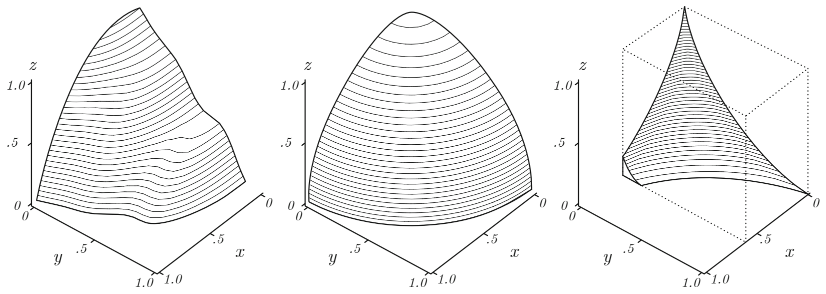


Fig. 1. 3-dimensional fronts implicitly described by $f_{3d}(x, y, z) = 0$ and restricted to the cube $[0, 1]^3$ (left and middle), and to $[0, 0.6] \times [0, 1] \times [0, 1]$ (right). **Left:** $f_{3d} = x^3 + (.1 \cdot (\sin(5\pi \cdot y) + 10)) \cdot y + z - 1$; **Middle:** $f_{3d} = x^2 + y^2 + z^2 - 1$; **Right:** $f_{3d} = x^{2/3} + y^{2/3} + z^{2/3} - 1$.

from the decision space X to its objective vector $\mathcal{F}(x) = (\mathcal{F}_1(x), \ldots, \mathcal{F}_k(x)) \in \mathbb{R}^k$ within the objective space $\mathcal{F}(X) \subseteq \mathbb{R}^k$. As the single objective functions $\mathcal{F}_i$, in general, cannot be simultaneously minimized and therefore no single optimal solution exists, we denote the sought *set* of so-called *Pareto-optimal solutions* (or *Pareto set*) as the set $\{x \in X \mid \nexists y \in X\colon y \preceq x \text{ and } x \npreceq y\}$. Thereby, the relation $\preceq$ is defined as: $x \preceq y$ if and only if $\mathcal{F}_i(x) \leq \mathcal{F}_i(y)$ for all $1 \leq i \leq k$ and we say, x is weakly dominating y if $x \preceq y$. The image of the Pareto set is called *Pareto front* or *front* for short. Note that in the remainder of this paper, we make an abuse of terminology and use the term *solution* both for a solution in the decision space and for its corresponding objective vector. Moreover, in order to increase readability, we also define $\preceq$ on objective vectors.

The Hypervolume Indicator. The hypervolume indicator of a solution set has been introduced as *the size of the objective space covered* [18]. Here, we formalize the hypervolume indicator $I_H(A, r)$ for sets of objective vectors $A \subseteq \mathbb{R}^k$ and a reference point $r \in \mathbb{R}^k$ according to [2] to ease readability compared to defining I_H for solutions in X as in the original paper: $I_H(A, r) = \lambda\big(\bigcup_{a \in A} C(a, r)\big)$ where $C(a, r) = \{z \in \mathbb{R}^k \mid a \preceq z \preceq r\}$ is the (hyper-)cuboid containing all objective vectors that are weakly dominated by a and themselves weakly dominate r. λ is the Lebesgue measure.

Notations for 3-objective problems. For the specific case of 3-objective problems, we assume the Pareto front to be implicitly describable as the points $(x, y, z) \in \mathbb{R}^3$ for which a function $f_{3d} : \mathbb{R}^3 \to \mathbb{R}$ is zero[2]: $f_{3d}(x, y, z) = 0$. W.l.o.g., we restrict the front to a cuboid $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\max}, z_{\min}]$, see Fig. 1. Besides a few exceptions of disrupted fronts, the Pareto fronts of well-known test problems, e.g., from the DTLZ [8], IHR [13], or WFG [12] test suites, can be described as assumed. Furthermore, we denote the reference point of the hypervolume by $r = (r_1, r_2, r_3)$. For some proofs, we will need an explicit description of the front, i.e., in terms of $z = f(x, y)$ (resp. $y = f(x, z)$, $x = f(y, z)$). Note that it is not always possible to find an explicit representation of an implicit equation $f_{3d}(x, y, z) = 0$. However, it is possible locally

[2] In addition, in order to describe a Pareto front, the partial derivatives of f_{3d} with respect to the first, second, and third variableare not supposed to change their sign, see for example [15].

assuming regularity of f_{3d} as stated by the implicit function theorem [14]. Therefore, assuming an explicit representation is not very restrictive.

3 General Results on Optimal μ-Distributions in 3-Objective Problems

In this section, we generalize some basic results of [2] about the existence of optimal μ-distributions and their monotonicity in μ to the 3-objective case. The proofs comprise the same ideas as in the bi-objective case though they are a bit more technical.

Theorem 1 (Existence of optimal μ-distributions for 3-objective problems). *Assume a 3-objective problem and assume that the front is described explicitly by a 2-dimensional function f, i.e., points of the Pareto front satisfy $z = f(x, y)$ (or $y = f(x, z)$ or $x = f(y, z)$). If the function f is continuous, there exists (at least) one set of μ points maximizing the hypervolume.*

Proof. Assume w.l.o.g. that the front is described via $z = f(x, y)$. Let $p_1, \ldots, p_\mu$ be μ points of $\mathbb{R}^3$. A point p_i writes as $(x_i, y_i, f(x_i, y_i))$. Since f is continuous, the mapping $((x_1, y_1), \ldots, (x_\mu, y_\mu)) \to \lambda(\bigcup_i C((x_i, y_i, f(x_i, y_i)), r))$, where $C((x_i, y_i, f(x_i, y_i)), r)$ is the cuboid with space diagonal defined by the extremes p_i and r, is continuous according to the Lebesgue dominated convergence theorem [4]. Moreover I_H is upper bounded by the hypervolume of the entire front. From the Extreme Value Theorem, there exists a set of μ points maximizing the hypervolume indicator. □

Note that the previous theorem states the existence but not the uniqueness, which cannot be guaranteed in general and that, in principle, the result can be easily generalized to the weighted hypervolume of [16]. A set of points maximizing the hypervolume whose existence is proven in the previous theorem will be called *optimal μ-distribution*. The associated value of the hypervolume is denoted as $\overline{I_H^\mu}$.

The following proposition establishes that the hypervolume of optimal $(\mu + 1)$-distributions is strictly larger than the hypervolume of optimal μ-distributions in the case of 3-objective problems and when the Pareto front contains at least $\mu + 1$ distinct points. This result is a generalization of Lemma 1 in [2].

Proposition 1 (Strict monotonicity in μ of the optimal hypervolume value). *Let $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\max}, z_{\min} \in \mathbb{R}$, $f_{3d} : \mathbb{R}^3 \to \mathbb{R}$, and let $P = \{(x, y, z) \in \mathbb{R}^3 \mid f_{3d}(x, y, z) = 0 \wedge (x_{\min} \leq x \leq x_{\max}) \wedge (y_{\min} \leq y \leq y_{\max}) \wedge (z_{\min} \leq z \leq z_{\max})\}$ describe the corresponding Pareto front. Let μ_1 and $\mu_2 \in \mathbb{N}$ with $\mu_1 < \mu_2$, then*

$$\overline{I_H^{\mu_1}} < \overline{I_H^{\mu_2}}$$

holds if P contains at least $\mu_1 + 1$ elements (x_i, y_i, z_i) for which $x_i < r_1$, $y_i < r_2$, and $z_i < r_3$ holds where $r = (r_1, r_2, r_3)$ is the hypervolume's reference point.

Proof. To prove the proposition, it suffices to show the inequality for $\mu_2 = \mu_1 + 1$ where we denote μ_1 by μ for readability. Assume the optimal μ-distribution is $D_\mu = \{(x_1^\mu, y_1^\mu, z_1^\mu), \ldots, (x_\mu^\mu, y_\mu^\mu, z_\mu^\mu)\}$ with $x_i^\mu, y_i^\mu, z_i^\mu \in \mathbb{R}$ and $f_{3d}(x_i^\mu, y_i^\mu, z_i^\mu) = 0$ for all

$1 \leq i \leq \mu$. Since P contains at least $\mu + 1$ elements, the set $P \backslash D_\mu$ is not empty and we can pick any $p_{\text{new}} = (x_{\text{new}}, y_{\text{new}}, z_{\text{new}}) \in P \backslash D_\mu$ to define a set $S = D_\mu \cup \{p_{\text{new}}\}$. As $I_H(D_{\mu_2}) \geq I_H(S)$ holds, it remains to prove that $I_H(S) > I_H(D_\mu)$. To this end, let us sort the points in D_μ with respect to each objective and pick the solution with the smallest x- (y-, z-) value which is larger than x_{new} (y_{new}, z_{new}) and denote it by $\overline{x}$ ($\overline{y}$, $\overline{z}$). If such a solution does not exist in D_μ, we set $\overline{x}$ to r_1 ($\overline{y}$ to r_2, $\overline{z}$ to r_3):

$$\begin{aligned}
\overline{x} &= \min \left\{ \{ x_i^\mu \mid (x_i^\mu, y_i^\mu, z_i^\mu) \in D_\mu \wedge x_{\text{new}} < x_i^\mu < r_1 \}, r_1 \right\} \\
\overline{y} &= \min \left\{ \{ y_i^\mu \mid (x_i^\mu, y_i^\mu, z_i^\mu) \in D_\mu \wedge y_{\text{new}} < y_i^\mu < r_2 \}, r_2 \right\} \\
\overline{x} &= \min \left\{ \{ z_i^\mu \mid (x_i^\mu, y_i^\mu, z_i^\mu) \in D_\mu \wedge z_{\text{new}} < z_i^\mu < r_3 \}, r_3 \right\}
\end{aligned}$$

Then, all objective vectors within $H_{\text{new}} := [x_{\text{new}}, \overline{x}) \times [y_{\text{new}}, \overline{y}) \times [z_{\text{new}}, \overline{z})$ are weakly dominated by p_{new} but are not dominated by any vector in D_μ. Furthermore, H_{new} is not a null set (i.e. has a strictly positive Lebesgue measure) since $x_{\text{new}} < \overline{x}$, $y_{\text{new}} < \overline{y}$, and $z_{\text{new}} < \overline{z}$. This additional contribution makes $I_H(S)$ strictly larger than $I_H(D_\mu)$. □

Although the result is proven only for the 3-objective case, the generalization to an arbitrary number of objectives is straightforward though technical such that we refrain from presenting it here. Moreover, the same monotonicity directly follows for the weighted hypervolume indicator of [16] by replacing *Lebesgue* by *weighted Lebesgue*.

4 Geometrical Properties of the Hypervolume Contributions of Single Solutions in 3-Objective Problems

As we have seen so far, some basic results about optimal μ-distributions can be easily transferred to the 3-objective case. For some other results of [2], mainly regarding the exact distribution of μ solutions that maximize the hypervolume indicator, generalizations to higher dimensions are more difficult. The main reason is the fact that the optimal placement of a single solution is not determined by only two neighbors anymore as it is the case for bi-objective problems, see [2, Proposition 1]. As we will see in this section, the hypervolume contribution of a single solution in a 3-objective scenario can be influenced by all other solutions. The stated properties of the possible *shape* of a solution's hypervolume contribution will be used in the following section to generalize a non-trivial result of [2] about the absence of the extreme points in optimal μ-distributions to the 3-objective case.

Before we investigate the general shape of the hypervolume dominated by a single solution, let us define a geometrical object to be a *generalized cylinder* if there exists one coordinate axis for which all cross sections of the geometrical object, perpendicular to this axis, yield the same 2-dimensional shape and the corresponding projections of the cross sections along this axis on the coordinate system are the same. The usual cylinder with a circular cross section is one specific case of such a generalized cylinder when oriented along a coordinate axis. Figure 3 shows an example of a generalized cylinder with a steplike cross section. With this definition, we can state the first result about the volume solely dominated by a single solution in 3 objectives, see Fig. 2:

Lemma 1. *Given a set $A \subseteq \mathbb{R}^3$ of 3-dimensional objective vectors, the hypervolume solely dominated by a single point $a \in A$ is an axis-aligned cuboid, with the point itself*

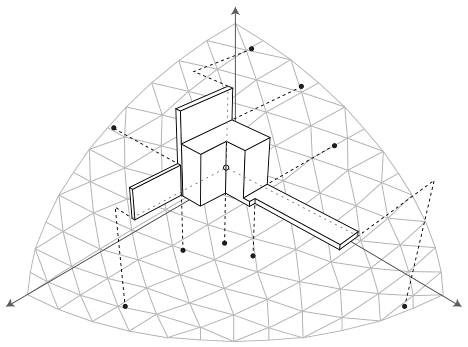


Fig. 2. Hypervolume contribution of a point (unfilled circle) on a three dimensional sphere function. The remaining nine points (black circles) all affect the shape of the contribution.

and the reference point as the end points of one of the cuboid's space diagonals, from which three generalized cylinders are cut—one parallel to each coordinate axis with steplike base areas, which depend on the other points in A.

Proof. The points that are weakly dominated by a specific 3-dimensional point $p = (x, y, z) \in \mathbb{R}^3$ and that weakly dominate the hypervolume's reference point form a cuboid $[x, r_1] \times [y, r_2] \times [z, r_3]$ with the point p as one corner and the reference point $r = (r_1, r_2, r_3)$ as the other end of its space diagonal. If we investigate now the points that are solely dominated by the point p, we have to subtract from this cuboid all solutions that are weakly dominated by other points $a = (a_1, a_2, a_3) \in A$, i.e., by the corresponding cuboids of which one corner is also the reference point. This gives the following set of points that are solely weakly dominated by p which can be obtained by deleting a general cylinder with steplike base area in each dimension from the cuboid associated to p: $([x, r_1] \times [y, r_2] \times [z, r_3]) \setminus \left(\bigcup_{\mathbf{a}\in A}[a_1, r_1] \times [a_2, r_2] \times [a_3, r_3]\right)$. $\square$

Note that the previous result does not only characterize the special shape of the hypervolume contribution of a single solution but also that this hypervolume contribution can be influenced by an arbitrary number of other solutions in a set A.

Interestingly, the shape of the space solely dominated by a single solution is becoming a generalized cylinder itself if we consider *extreme solutions* of a solution set A, see Fig. 3 (left). A solution a^i is thereby called extreme with respect to A (or extreme point of A) and objective $\mathcal{F}_i$, if no other solution in A has larger values in objective $\mathcal{F}_i$, i.e., $a^i \in \operatorname{argmax}\{a' \in A \mid \nexists a'' \in A\colon \mathcal{F}_i(a') < \mathcal{F}_i(a'')\}$. Note that extreme points are not unique in the 3-objective case in general and that their objective values do not always coincide with the values $x_{\max}$, $y_{\max}$, and $z_{\max}$, see, e.g., Fig. 1. We denote the obtained maximal values of extreme points in the three dimensions as $\overline{x}$, $\overline{y}$, and $\overline{z}$ respectively.

Lemma 2. *Given a set of 3-dimensional objective vectors $A \subseteq \mathbb{R}^3$. An extreme point of A, i.e., a point with the largest objective value among all points in A for (at least) one objective, solely dominates a region the shape of which is itself a generalized cylinder with a steplike base area.*

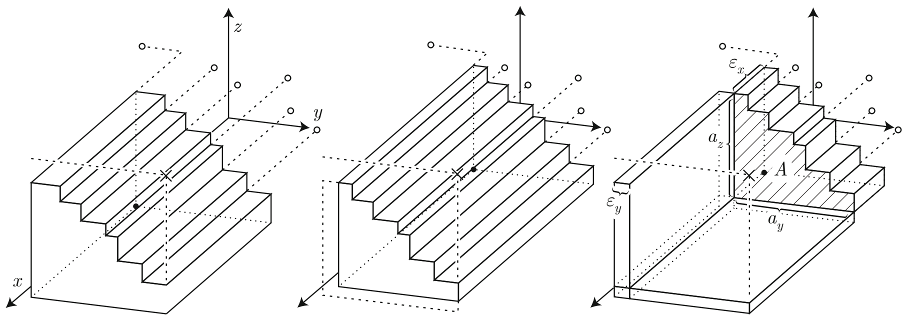


Fig. 3. Illustration of the hypervolume contribution of an extreme point in x-direction. **Left:** hypervolume solely dominated by the extreme point p_{xmax} which is depicted by a black circle; **Middle:** hypervolume solely dominated by the moved point $p_{\text{xmax}} - \varepsilon$; **Right:** illustration of benefit and deficit in hypervolume if we move the extreme point towards $p_{\text{xmax}} - \varepsilon$; in all three plots, the reference point is depicted by a cross and the remaining points influencing the extreme point's hypervolume contribution are depicted by unfilled circles.

Proof. Let us consider w.l.o.g. only one extreme point in x-direction and its hypervolume contribution, i.e., a point $p_{\text{xmax}} = (\overline{x}, y, z)$ with the largest x-value $\overline{x}$ among μ solutions on the front. Without any other point, the hypervolume contribution of p_{xmax} would be again the cuboid from above with the point itself as one corner and the hypervolume's reference point as the other end of the cuboid's space diagonal starting at p_{xmax}. Due to other incomparable solutions on the front, this cuboid is pruned in a specific way. To investigate how the hypervolume contribution of p_{xmax} is influenced by other points on the front, we consider the projection of all points to the y-z-plane, see Fig. 4. Two statements can be easily proven: (i) no point in the lower left region of p_{xmax} exists (otherwise it would be dominating p_{xmax} due to its better objective values in x-, y-, and z-direction) and (ii) all other solutions dominate a cuboid themselves and therefore cut this cuboid from the extreme point's cuboid (all solutions obviously dominate a volume that is a cuboid and the points are in addition not worse in x-direction and therefore their dominated volume is reaching in x-direction over the entire cuboid of p_{xmax}). This results in a volume solely dominated by p_{xmax}, that has a steplike projection and is a general cylinder in x-direction, see the leftmost plot of Fig. 3. □

5 Fronts for Which It Is Impossible to Obtain the Extreme Points

Given the knowledge about the shape of the hypervolume solely dominated by extreme solutions obtained above, we are able to generalize another result on optimal μ-distributions of [2] to the 3-objective case: There are cases where no finite reference point allows to have an extreme point of the Pareto front contained in optimal μ-distributions. In the 3-objective case, this corresponds to the cases where the (finite) partial derivative of the front at an extreme with respect to the first (second, third) axis is perpendicular to the first (second, third) axis:

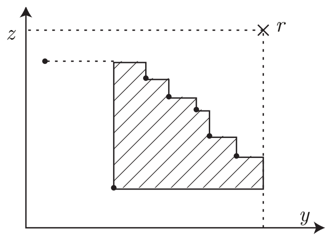


Fig. 4. Projection of all points to the y-z-plane

Theorem 2. *Let f_{3d} be a continuous and differentiable function describing the front with gradient ∇f_{3d} continuous in $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$. If the gradient $\nabla f_{3d}(\overline{x}, y, z) = (\partial_1 f_{3d}(\overline{x}, y, z), \partial_2 f_{3d}(\overline{x}, y, z), \partial_3 f_{3d}(\overline{x}, y, z))$ of the front at an extreme point $(\overline{x}, y, z)$ (at an extreme point $(x, \overline{y}, z)$, or at $(x, y, \overline{z})$) is finite, i.e., the single components of the gradient are finite, and the gradient is perpendicular to the x-axis (y-, z-axis), i.e., if $\nabla f_{3d}(\overline{x}, y, z) \cdot (1, 0, 0) = 0$ $(\nabla f_{3d}(x, \overline{y}, z) \cdot (0, 1, 0) = 0$, $\nabla f_{3d}(x, y, \overline{z}) \cdot (0, 0, 1) = 0)$, the corresponding extreme point is not included in any optimal μ-distribution with $\mu \geq 1$.*

Proof. W.l.o.g., we consider only the case of the extreme point $p_{\text{xmax}} := (\overline{x}, y, z)$ where $\nabla f_{3d}(p_{\text{xmax}}) \cdot (1, 0, 0) = 0$ and therefore $\partial_1 f_{3d}(p_{\text{xmax}}) = 0$. The proof idea is similar to the bi-objective case in [2]: we consider the hypervolume that we gain and the hypervolume that we lose if we move the extreme point p_{xmax} towards larger values of y and z. To this end, we use the notations of Fig. 3. In particular, we move p_{xmax} by a small value $\varepsilon_x > 0$ in x direction towards smaller x-values and at the same time parallel to the plane defined by the x-axis and the gradient $\nabla f_{3d}(p_{\text{xmax}})$ along the front:

$$p_{\text{xmax}} - \varepsilon = p_{\text{xmax}} - \begin{pmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \end{pmatrix} = p_{\text{xmax}} - \begin{pmatrix} \varepsilon_x \\ 0 \\ 0 \end{pmatrix} + \nu \begin{pmatrix} 0 \\ \partial_2 f(p_{\text{xmax}}) \\ \partial_3 f(p_{\text{xmax}}) \end{pmatrix} \tag{1}$$

where $\nu \in \mathbb{R}$ and the last equality follows from the assumption that the gradient $\nabla f(p_{\text{xmax}})$ is perpendicular to the x-axis. Note that at least one of the values ε_y and ε_z has to be negative since all points on the Pareto front are incomparable and a point with all three objectives smaller than p_{xmax} would therefore not lie on the front.

If we choose ε_x small enough, i.e., as long as there is no other point $p' = (x', y', z')$ with $x' > \overline{x} - \varepsilon_x$, $y' > y$, and $z' > z$ among the μ solutions under consideration, the hypervolume contribution of the moved extreme point keeps its shape, see Fig. 3. According to the rightmost plot of Fig. 3, we denote the area of the y-z-projection of the new point's hypervolume contribution as A, by a_z the height of this area in z-direction, and by a_y the length of this area in y-direction. Then, the benefit and deficit in hypervolume if we move the extreme point from p_{xmax} to $p_{\text{xmax}} - \varepsilon$ can be written as

$$\text{benefit: } \varepsilon_x \cdot A \qquad \text{deficit: } (\varepsilon_y \cdot a_z + \varepsilon_z \cdot a_y + \varepsilon_y \cdot \varepsilon_z) \cdot (r_1 - \overline{x}) \enspace .$$

Now, it remains to be shown that the ratio between deficit and benefit goes to zero when ε_x converges to zero. To this end, we decompose the ratio R between deficit and benefit as $R = \frac{\text{deficit}}{\text{benefit}} = \frac{\varepsilon_y \cdot a_z \cdot (r_1 - \overline{x})}{\varepsilon_x \cdot A} + \frac{\varepsilon_z \cdot a_y \cdot (r_1 - \overline{x})}{\varepsilon_x \cdot A} + \frac{\varepsilon_y \cdot \varepsilon_z \cdot (r_1 - \overline{x})}{\varepsilon_x \cdot A}$. Because A is lower bounded by a constant and a_x and a_z are upper bounded by a constant, showing that $\lim_{\varepsilon_x \to 0} \frac{\varepsilon_y}{\varepsilon_x} = 0$ and $\lim_{\varepsilon_x \to 0} \frac{\varepsilon_z}{\varepsilon_x} = 0$ will directly prove that the ratio R converges to zero. W.l.o.g., we are only going to prove $\lim_{\varepsilon_x \to 0} \frac{\varepsilon_y}{\varepsilon_x} = 0$ in the following. The proof of $\lim_{\varepsilon_x \to 0} \frac{\varepsilon_z}{\varepsilon_x} = 0$ can be done in the same way by exchanging the roles of ε_y and ε_z.

Assuming $\partial_2 f_{3d}(p_{\text{xmax}}) \neq 0$ (otherwise, $\lim_{\varepsilon_x \to 0} \frac{\varepsilon_y}{\varepsilon_x} = 0$ follows directly), we know from Eq. 1 that $\nu = \frac{\varepsilon_y}{\partial_2 f_{3d}(p_{\text{xmax}})}$ and therefore that

$$\varepsilon_z = \varepsilon_y \frac{\partial_3 f_{3d}(p_{\text{xmax}})}{\partial_2 f_{3d}(p_{\text{xmax}})} \quad . \tag{2}$$

Since the gradient of f_{3d} is continuous within the cuboid restricting the front, we can expand $f_{3d}(p_{\text{xmax}} - \varepsilon)$ with the Taylor formula as $f_{3d}(p_{\text{xmax}} - \varepsilon) = f_{3d}(p_{\text{xmax}}) - \nabla f_{3d}(p_{\text{xmax}}) \cdot \varepsilon + \mathcal{O}(||\varepsilon||^2)$ which indicates that $\nabla f_{3d}(p_{\text{xmax}}) \cdot \varepsilon - \mathcal{O}(||\varepsilon||^2) = 0$ for any $\varepsilon \geq 0$ as f_{3d} equals zero for all points on the front by definition. From $\nabla f_{3d}(p_{\text{xmax}}) \cdot \varepsilon - \mathcal{O}(||\varepsilon||^2) = 0$ we can conclude that $\lim_{\varepsilon \to 0} \nabla f_{3d}(p_{\text{xmax}}) \cdot \varepsilon = \lim_{\varepsilon \to 0} \mathcal{O}(||\varepsilon||^2)$ and even $\lim_{\varepsilon \to 0} (\nabla f_{3d}(p_{\text{xmax}}) \cdot \varepsilon / ||\varepsilon||) = \lim_{\varepsilon \to 0} \mathcal{O}(||\varepsilon||) = 0$. Since $\partial_1 f_{3d}(p_{\text{xmax}}) = 0$ and the other partial derivatives $\partial_2 f_{3d}(p_{\text{xmax}})$ and $\partial_2 f_{3d}(p_{\text{xmax}})$ are finite and constant, the previous equation can be rewritten as

$$\lim_{\varepsilon \to 0} \left(\left(\partial_2 f_{3d}(p_{\text{xmax}}) \cdot \varepsilon_y + \partial_3 f_{3d}(p_{\text{xmax}}) \cdot \varepsilon_z\right) / \sqrt{\varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2} \right) = 0 \quad .$$

Using (2) in the previous equation and factorizing the numerator and denominator by ε_y we obtain

$$\lim_{\varepsilon \to 0} \frac{\partial_2 f_{3d}(p_{\text{xmax}}) + \frac{(\partial_3 f_{3d}(p_{\text{xmax}}))^2}{\partial_2 f_{3d}(p_{\text{xmax}})}}{\sqrt{\frac{\varepsilon_x^2}{\varepsilon_y^2} + 1 + \frac{(\partial_3 f_{3d}(p_{\text{xmax}}))^2}{(\partial_2 f_{3d}(p_{\text{xmax}}))^2}}} = 0$$

which implies that $\lim_{\varepsilon \to 0} \frac{\varepsilon_y}{\varepsilon_x} = 0$. □

Remark. Note that the case covered by the previous theorem is not an artificial case but obtained for some well-known test problems, e.g., DTLZ2–4 [8] and WFG4–9 [12]. It also covers the bi-objective case proven in [2] which, however, used a slightly different notation due to a simpler description of the front shape.

6 Conclusions

Obtaining optimal μ-distributions for a certain quality indicator I, i.e., a set of μ solutions maximizing I, is the optimization goal of several indicator-based multiobjective evolutionary algorithms [2]. In particular, the hypervolume indicator, among others employed in the SMS-EMOA [6] and the MO-CMA-ES [13], received interest as a selection criterion in multiobjective algorithms due to its property of being a refinement of the Pareto dominance relation. However, theoretical investigations of optimal μ-distributions for the hypervolume indicator are rare and limited to the bi-objective case so far [9,5,2,1,11,7].

Here, we obtain first theoretical results on optimal μ-distributions for the hypervolume indicator in 3-objective scenarios. It turns out that the hypervolume contribution of a single point has a specific shape that is not as simple as in bi-objective problems anymore—indicating that *all* solutions have an influence on where to optimally place a solution instead of only *two* solutions in the bi-objective case. Besides generalizations of basic statements of [2] to the 3-objective case, we prove in particular that also in 3-objective problems there are situations where no finite reference point can ensure the extreme solutions of the Pareto front within an optimal μ-distribution.

Acknowledgments. This work has been in part supported by the French national research agency (ANR) within the SYSCOMM project ANR-08-SYSC-017 and within the COSINUS project ANR-08-COSI-007-12.

References

1. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Investigating and Exploiting the Bias of the Weighted Hypervolume to Articulate User Preferences. In: Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 563–570. ACM, New York (2009)
2. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the Hypervolume Indicator: Optimal μ-Distributions and the Choice of the Reference Point. In: Foundations of Genetic Algorithms (FOGA 2009), pp. 87–102. ACM, New York (2009)
3. Bader, J.: Hypervolume-Based Search for Multiobjective Optimization: Theory and Methods. PhD thesis, ETH Zurich, Switzerland (2010)
4. Bartle, R.G.: The Elements of Integration and Lebesgue Measure. Wiley, Chichester (1995)
5. Beume, N., Fonseca, C.M., Lopez-Ibanez, M., Paquete, L., Vahrenhold, J.: On the Complexity of Computing the Hypervolume Indicator. IEEE T. Evolut. Comput. 13(5), 1075–1082 (2009)
6. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. Eur. J. Oper. Res. 181, 1653–1669 (2007)
7. Bringmann, K., Friedrich, T.: The Maximum Hypervolume Set Yields Near-optimal Approximation. In: Genetic and Evolutionary Computation Conference, GECCO 2010, pp. 511–518. ACM, New York (2010)
8. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multi-Objective Optimization. In: Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, ch. 6, pp. 105–145. Springer, Heidelberg (2005)
9. Emmerich, M., Deutz, A., Beume, N.: Gradient-Based/Evolutionary Relay Hybrid for Computing Pareto Front Approximations Maximizing the S-Metric. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HCI/ICCV 2007. LNCS, vol. 4771, pp. 140–156. Springer, Heidelberg (2007)
10. Fleischer, M.: The Measure of Pareto Optima. Applications to Multi-Objective Metaheuristics. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 519–533. Springer, Heidelberg (2003)
11. Friedrich, T., Horoba, C., Neumann, F.: Multiplicative Approximations and the Hypervolume Indicator. In: Genetic and Evolutionary Computation Conference (GECCO 2009), pp. 571–578. ACM, New York (2009)
12. Huband, S., Hingston, P., Barone, L., While, L.: A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit. IEEE T. Evolut. Comput. 10(5), 477–506 (2006)
13. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)

14. Rudin, W.: Principles of Mathematical Analysis, 3rd edn. McGraw-Hill, New York (1976)
15. Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Symposium on Applied Computing, pp. 351–357. ACM, New York (1999)
16. Zitzler, E., Brockhoff, D., Thiele, L.: The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)
17. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. 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. 832–842. Springer, Heidelberg (2004)
18. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE T. Evolut. Comput. 3(4), 257–271 (1999)
19. Zitzler, E., Thiele, L., Bader, J.: On Set-Based Multiobjective Optimization. IEEE T. Evolut. Comput. 14(1), 58–79 (2009)

Convergence Rates of (1+1) Evolutionary Multiobjective Optimization Algorithms

Nicola Beume[1], Marco Laumanns[2], and Günter Rudolph[1]

[1] Fakultät für Informatik, Technische Universität Dortmund, Germany
{nicola.beume,guenter.rudolph}@tu-dortmund.de
[2] IBM Research – Zurich, Switzerland
mlm@zurich.ibm.com

Abstract. Convergence analyses of evolutionary multiobjective optimization algorithms typically deal with the convergence in limit (stochastic convergence) or the run time. Here, for the first time concrete results for convergence rates of several popular algorithms on certain classes of continuous functions are presented. We consider the algorithms in the version of using a (1+1) selection scheme. Then, SMS-EMOA and IBEA$_{\epsilon+}$ achieve linear convergence rate, proved by showing algorithmic equivalence to the single-objective (1+1)-EA with self-adaptation, whereas NSGA-II and SPEA2 have a sub-linear convergence rate, proved by reducing them to a multiobjective algorithm with known properties.

Keywords: multiobjective optimization, convergence rate, hypervolume, self-adaptation.

1 Introduction

Research on evolutionary algorithms is developed further for single-objective optimization than for multi-objective optimizers. A common hope is that the understanding of evolutionary multiobjective optimization algorithms (EMOA) can profit from the bases acquired for the single-objective case. Here, we transfer knowledge on the convergence of the single-objective (1+1)-EA to gain insights into the convergence behavior of complex EMOA.

Convergence properties of EMOA are yet not well understood. More recently, theory concentrated on the convergence or runtime of simple EMOA on special discrete problems, considering whether and how quickly the Pareto set is reached. For the case of a continuous search space $\mathbb{R}^n$ only a few results exist for specialized algorithms, the first obtained by Rudolph [1]. He showed that a multiobjective (1+1)-EA that accepts incomparable points with probability $\frac{1}{2}$ converges with probability 1 to the Pareto set if the step size is chosen proportional to the distance to the Pareto set, while two other step size concepts fail. Hanne [2] considered stochastic convergence of EMOA with different selection schemes, the possibilities of temporary fitness deterioration, and on problems with unattainable solutions. A recent subject of interest has been whether a certain distribution on the Pareto front can be obtained that is optimal regarding specified preferences.

Despite these advances, the convergence rate in continuous space remains a neglected topic. Teytaud [3] shows that the convergence rate scales badly with increasing number of objectives entailing that any comparison-based EMOA performs hardly better than random search for a large number of objectives. Also a general lower bound for the convergence time is given.

In this paper we consider popular EMOA in the simple version of using a (1+1) selection scheme. For (strongly) convex quadratic objective functions, the order of the convergence rate is calculated, whereas SMS-EMOA and IBEA$_{\epsilon+}$ reach a linear convergence rate. This is to the best of our knowledge the first time that a linear convergence rate is shown for a multiobjective evolutionary algorithms that does not use an explicit weighting of objectives.

The next section introduces the technical background of our topic. Section 3 shows the linear convergence rate for SMS-EMOA and IBEA, whereas Section 4 gives the negative results for NSGA-II and SPEA2. We summarize our findings in section 5 and give hints on future research.

2 Preliminaries

2.1 Single-Objective Optimization with the (1+1)-EA

Let $f : \mathbb{R}^n \to \mathbb{R}$ be the objective function to be minimized. The $(1+1)$ Evolutionary Algorithm (EA) (cf. Alg. 1) minimizes $f(\cdot)$ by drawing an n-dimensional random vector from a multivariate standard normal distribution that is scaled by factor σ and then added to the current position. If the new point is better it is accepted, otherwise it is rejected. Then the scaling factor is adapted and this sequence is run again.

Algorithm 1. $(1+1)$-Evolutionary Algorithm with Self-Adaptation

```
1  choose $X^{(0)} \in \mathbb{R}^n$ and $\sigma^{(0)} > 0$, set $t = 0$ and $k = 0$
2  **repeat**
3      draw $Z^{(t)}$ from a multivariate standard normal distribution
4      $Y^{(t)} = X^{(t)} + \sigma^{(t)} Z^{(t)}$
5      **if** $f(Y^{(t)}) \leq f(X^{(t)})$ **then**
6          $X^{(t+1)} = Y^{(t)}$ ; increment $k$
7      **else** $X^{(t+1)} = X^{(t)}$
8      $\sigma^{(t+1)} = \texttt{adapt}(\sigma^{(t)}, t, k ; \delta, p_s, \gamma)$
9      increment $t$
10 **until** *termination criterion fulfilled*
```

The self-adaptation mechanism considered here (Procedure 2) is termed the $\frac{1}{5}$-success rule that has some parameters: the observation interval $\delta > 0$, the success probability $p_s = \frac{1}{5}$ and the adaptation factor $\gamma > 1$. If the self-adaptation procedure is properly parameterized a remarkable result has been proven by Jägersküpper [4,5]:

Theorem 1. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a quadratic function $f(x) = x'Ax + b'x + c$ with positive definite matrix A whose condition number is bounded. The* $(1+1)$-*EA with self-adaptation as in Procedure 2 using $\delta = \Theta(n)$, $p_s = \frac{1}{5}$ and $c \geq 2$*

Procedure 2. `adapt`$(\sigma, t, k\,; \delta, p_s, \gamma)$

```
1 if $t \mod \delta \neq 0$ then return $\sigma$
2 $q_s = k/\delta$ ;  $k = 0$
3 if $q_s \geq p_s$ then return $\sigma \times \gamma$ else return $\sigma/\gamma$
```

halves the distance to the optimum in $O(n)$ iterations in expectation, provided that $\sigma^{(0)} = \Theta(D/n)$ where D is the distance to the optimum after initialization.

In other words: under the conditions of the theorem the (1+1)-EA with self-adaptation minimizes every strongly convex quadratic function with linear convergence rate, i.e., the approximation error decreases exponentially fast.

2.2 Multi-objective Optimization with the SMS-EMOA

We consider unconstrained multiobjective optimization problems $\min f(x) : \mathbb{R}^n \to \mathbb{R}^d$ where $f(x) = (f_1(x), \ldots, f_d(x))$ maps an n-dimensional vector of the search space to a d-dimensional vector of the objective space.

A strict partial order, called Pareto dominance, holds in the objective space based on the coordinate-wise total order: a point $p = (p_1, \ldots, p_d)$ weakly dominates a point q (written as $p \preceq q$) iff $p_i \leq q_i$ holds for all $1 \leq i \leq d$. A point p dominates q iff $p \preceq q$ and $p \neq q$. Two distinct points $p \neq q$ are incomparable $(p \parallel q)$ iff neither point dominates the other. Considering a set $A \subseteq \mathbb{R}^d$, points of A that are not dominated by any other of A are referred to as non-dominated in A or the minima of A. Those points that are non-dominated regarding the whole objective space are Pareto-optimal and called the Pareto front. The set of their preimages in the search space is named the Pareto set.

In the continuous domain only an approximation of the Pareto set can be expected to be achieved. In order to compare the results of different EMOA, several quality measures exist, typically rewarding quantity, closeness to the Pareto set, and high diversity. Among these, the hypervolume indicator (or S-metric or Lebesque measure) by Zitzler and Thiele [6] is of outstanding importance due to its consistency with the Pareto dominance relation, cf. [7].

Definition 1. *Let $\{v^{(1)}, v^{(2)}, \ldots, v^{(\mu)}\} \subset \mathbb{R}^d$, $d \geq 2$ be a finite set of elements, which are mutually incomparable w. r. t. to the dominance relation $\preceq$. Let $r \in \mathbb{R}^d$ indicate the* reference point *with $v^{(i)} \prec r$ for all $i = 1, \ldots, \mu \in \mathbb{N}$. The quantity*

$$H(v^{(1)}, \ldots, v^{(\mu)}; r) = \mathsf{Leb}\left(\bigcup_{i=1}^{\mu} [v^{(i)}, r]\right) \tag{1}$$

is termed the dominated hypervolume *or* S-metric *where* $\mathsf{Leb}(\cdot)$ *denotes the Lebesgue measure in $\mathbb{R}^d$.*

If $d = 2$, provided the elements $v^{(1)}, \ldots, v^{(\mu)}$ have been labeled in ascending order of their first component, i.e., $v_1^{(1)} < v_1^{(2)} < \ldots < v_1^{(\mu)}$, equation (1) specializes to

$$H(v^{(1)}, \ldots, v^{(\mu)}; r) = (r_1 - v_1^{(1)})\,(r_2 - v_2^{(1)}) + \sum_{i=2}^{\mu} (r_1 - v_1^{(i)})\,(v_2^{(i-1)} - v_2^{(i)}). \tag{2}$$

The SMS-EMOA [8] is a steady-state, i.e. $(\mu+1)$, EMOA that aims to maximize the population's dominated hypervolume by incorporating it in the selection operator. The selection starts with non-dominated sorting in order to determine the worst front. Among these points, the one contributing least to the dominated hypervolume of the set is discarded. The hypervolume contribution of a point is defined as the dominated hypervolume that is exclusively dominated by the point and thus would get lost when the point was discarded. The calculation of the hypervolume requires the specification of a reference point r. Yet, it is no exogenous parameter of the SMS-EMOA but chosen automatically. For each objective function, the maximal value among the $\mu+1$ points is determined. The reference point is constructed by these maxima plus 1. The decisive properties that will be utilized to prove the linear convergence rate are: (1) The reference point is not static throughout the optimization process but dynamically adapted in each generation. (2) Those points that are the worst ones in the population regarding an objective function have a distance to the reference point of exactly 1 w.r.t. that worst objective (cf. Fig. 1, left).

The SMS-EMOA does not specify a certain variation operator. It has mainly been considered using SBX recombination and polynomial mutation (see e.g. [9]). Here, we consider Gaussian mutation with self-adaptation as detailed in the following section.

Section 3.3 contains the analysis of IBEA$_{\epsilon+}$ [10] that performs non-dominated sorting and afterwards selects among the worst points using the additive ϵ-indicator $I_{\epsilon+}$. Section 4 deals with NSGA-II [9] that firstly uses non-dominated sorting, and afterwards a particular density measure, the crowding distance, as the secondary selection criterion. SPEA2 [11] as well applies a selection criterion based on the Pareto dominance relation by counting dominated and dominating solutions for each point. Among the incomparable ones, again a kind of density measure comes into play, namely a k-nearest neighbor method.

3 Linear Convergence Rates

3.1 (1+1)-SMS-EMOA on 2-Objective Problems

If the $(\mu+1)$-SMS-EMOA is instantiated with $\mu = 1$ it reduces to the algorithm described below (see Alg. 3).

Theorem 2. *The* $(1+1)$*-SMS-EMOA with self-adaptation applied to applied to a bi-objective optimization problem* $\min\{f : \mathbb{R}^n \to \mathbb{R}^2\}$ *is algorithmically equivalent to a* $(1+1)$*-EA with self-adaptation applied to the minimization of the single-objective function* $f^s : \mathbb{R}^n \to \mathbb{R}$ *with* $f^s(x) = \frac{1}{2}\,(f_1(x) + f_2(x))$.

Proof. The $(1+1)$-SMS-EMOA differs from the $(1+1)$-EA only in the additional determination of the reference point $R^{(t)}$ which is required in the seemingly more complex acceptance criterion. Evidently, it is sufficient to show that the $(1+1)$-SMS-EMOA accepts/rejects a new point if it would be accepted/rejected by the $(1+1)$-EA with the scalarized objective function (cf. Fig. 1 (right) for its regions of acceptance or rejection).

Algorithm 3. (1+1)-SMS-EMOA with Self-Adaptation

```
1  choose $X^{(0)} \in \mathbb{R}^n$ and $\sigma^{(0)} > 0$, set $t = 0$ and $k = 0$
2  repeat
3      draw $Z^{(t)}$ from a multivariate standard normal distribution
4      $Y^{(t)} = X^{(t)} + \sigma^{(t)} Z^{(t)}$
5      $R^{(t)} = (\max\{f_1(X^{(t)}), f_1(Y^{(t)})\} + 1, \max\{f_2(X^{(t)}), f_2(Y^{(t)})\} + 1)'$
6      if $f(Y^{(t)}) \prec f(X^{(t)})$ or
         $\big(f(Y^{(t)}) \parallel f(X^{(t)})$ and $H(f(Y^{(t)}); R^{(t)}) > H(f(X^{(t)}); R^{(t)})\big)$ then
7          $X^{(t+1)} = Y^{(t)}$ ; increment $k$
8      else $X^{(t+1)} = X^{(t)}$
9      $\sigma^{(t+1)} = \texttt{adapt}(\sigma^{(t)}, t, k\,; \delta, p_s, c)$
10     increment $t$
11 until termination criterion fulfilled
```

The mutated individual y is accepted if it dominates its parent x, i.e., $f(y) \prec f(x)$. This implies $f^s(y) < f^s(x)$ and the $(1+1)$-EA would accept y:

$$\begin{aligned} f(y) \prec f(x) &\Leftrightarrow f_1(y) < f_1(x) \;\wedge\; f_2(y) < f_2(x) \\ &\Rightarrow f_1(y) + f_2(y) < f_1(x) + f_2(x) \\ &\Leftrightarrow \frac{1}{2}\, f_1(y) + \frac{1}{2}\, f_2(y) < \frac{1}{2}\, f_1(y) + \frac{1}{2}\, f_2(y) \\ &\Leftrightarrow f^s(y) < f^s(x) \end{aligned}$$

Moreover, the mutated individual y is also accepted if it is incomparable to its parent x, i.e., $f(y) \parallel f(x)$, but has a larger dominated hypervolume. This condition also implies $f^s(y) < f^s(x)$ which is easily seen as follows: Since $f(y) \parallel f(x)$ we have to distinguish two cases.

1. $f_1(x) > f_1(y) \wedge f_2(x) < f_2(y)$

   According to Algorithm 3 the reference point is $r = (f_1(x) + 1, f_2(y) + 1)'$, here. Recall that $H(v; r) = (r_1 - v_1)\,(r_2 - v_2)$ for a single point. It follows:

   $$\begin{aligned} H_x &= H(f(x); r) = [\, f_1(x) + 1 - f_1(x)\,]\,[\, f_2(y) + 1 - f_2(x)\,] = f_2(y) - f_2(x) + 1 \\ H_y &= H(f(y); r) = [\, f_1(x) + 1 - f_1(y)\,]\,[\, f_2(y) + 1 - f_2(y)\,] = f_1(x) - f_1(y) + 1 \end{aligned}$$

   The new point y is accepted if

   $$\begin{aligned} H_y \overset{!}{>} H_x &\Rightarrow f_1(x) - f_1(y) + 1 > f_2(y) - f_2(x) + 1 \\ &\Leftrightarrow f_1(x) - f_1(y) > f_2(y) - f_2(x) \\ &\Leftrightarrow f_1(x) + f_2(x) > f_1(y) + f_2(y) \\ &\Leftrightarrow f^s(x) > f^s(y)\,. \end{aligned}$$

   Thus, in this particular situation the $(1+1)$-SMS-EMOA would accept y and so would do the $(1+1)$-EA.

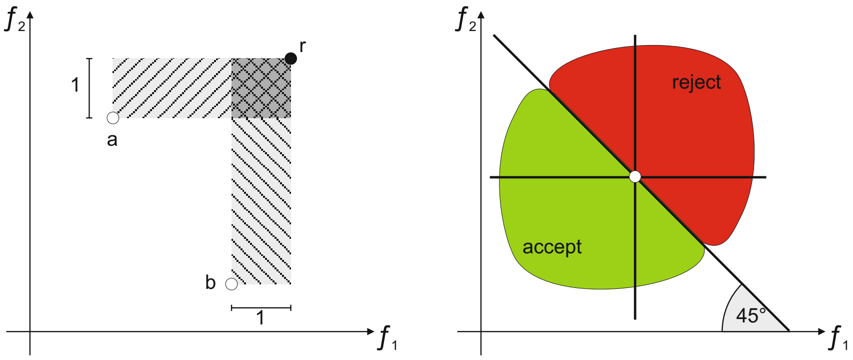


Fig. 1. Left: Dominated hypervolume of the points a and b w.r.t. the reference point r, whereas the hypervolume contribution is shaded in light gray. Right: Regions of acceptance or rejection for the substitute weighted sum function.

2. $f_1(x) < f_1(y) \wedge f_2(x) > f_2(y)$
   Now the reference point is $r = (f_1(y)+1, f_2(x)+1)'$ yielding a dominated hypervolume of
   $H_x = H(f(x);r) = [\,f_1(y)+1-f_1(x)\,]\,[\,f_2(x)+1-f_2(x)\,] = f_1(y)-f_1(x)+1$
   $H_y = H(f(y);r) = [\,f_1(y)+1-f_1(y)\,]\,[\,f_2(x)+1-f_2(y)\,] = f_2(x)-f_2(y)+1$
   The new point y is accepted if

$$\begin{aligned} H_y \overset{!}{>} H_x &\Rightarrow f_2(x) - f_2(y) + 1 > f_1(y) - f_1(x) + 1 \\ &\Leftrightarrow f_2(x) - f_2(y) > f_1(y) - f_1(x) \\ &\Leftrightarrow f_1(x) + f_2(x) > f_1(y) + f_2(y) \\ &\Leftrightarrow f^s(x) > f^s(y) \end{aligned}$$

   Again, the $(1+1)$-SMS-EMOA would accept the new point y and so would do the $(1+1)$-EA.

Putting all together we have shown that whenever the $(1+1)$-SMS-EMOA accepts a new element so does the $(1+1)$-EA. Finally we have to preclude that the $(1+1)$-EA accepts elements that are rejected by the SMS-EMOA. Or equivalently, if the $(1+1)$-SMS-EMOA rejects a new element then so must do the $(1+1)$-EA. Notice that the proof of this property is analogous to acceptance case above and therefore omitted here. □

The equivalence of $(1+1)$-SMS-EMOA and the specific $(1+1)$-EA as formulated in Th. 2 holds for all bi-objective problems. For a certain class of problems, we show a linear convergence rate:

Corollary 1. *The* $(1+1)$-*SMS-EMOA with self-adaptation applied to applied to a bi-objective optimization problem* $\min\{f : \mathbb{R}^n \to \mathbb{R}^2\}$ *approaches an element of the Pareto front with linear order of convergence if both objective functions are quadratically convex and at least one of them strongly convex.*

Proof. Since both objective functions are quadratically convex they are of form

$$f_1(x) = \tfrac{1}{2}x'Ax + b'x + c \qquad \text{and} \qquad f_2(x) = \tfrac{1}{2}x'\check{A}x + \check{b}'x + \check{c}$$

with positive semidefinite matrices A and $\check{A}$. Notice that at least one objective function is even strongly convex so that its Hessian matrix is positive definite. Suppose w. l. o. g. that A is positive definite. Since

$$f^s(x) = \frac{1}{2}(f_1(x) + f_2(x)) = \frac{1}{2}\left[\frac{1}{2}x'(A+\check{A})x + (b+\check{b})'x + (c+\check{c})\right]$$

$$\text{and} \qquad x'(A+\check{A})x \;=\; \underbrace{x'Ax}_{>0} + \underbrace{x'\check{A}x}_{\geq 0} \;>\; 0$$

for all $x \in \mathbb{R}^n \setminus \{0\}$, its Hessian matrix is positive definite ensuring that $f^s(x)$ is a strongly convex quadratic function. Now we can invoke Theorem 1 that guarantees linear convergence rate of the $(1+1)$-EA with self-adaptation for $f^s(x)$. Owing to Theorem 2 we know that the $(1+1)$-EA with self-adaptation is algorithmically equivalent to a $(1+1)$-SMS-EMOA for minimizing the bi-objective function $(f_1(x), f_2(x))'$. As a consequence, the $(1+1)$-SMS-EMOA must have linear convergence rate to an element of the Pareto front under the conditions of the corollary. □

3.2 (1+1)-SMS-EMOA beyond Two Objectives

Expectedly, the result from the previous section does not generalize to more than two objectives, which we show by a simple counter-example. First notice that the reference point for two points $f(x)$ and $f(y)$ in objective space $\mathbb{R}^d$ is

$$r = \big(\max\{f_1(x), f_1(y)\} + 1, \max\{f_2(x), f_2(y)\} + 1, \ldots, \max\{f_d(x), f_d(y)\} + 1\big)'$$

where $f : \mathbb{R}^n \to \mathbb{R}^d$, $d \geq 2$. The dominated hypervolume $H(v;r)$ for a single point $v \in \mathbb{R}^d$ and the scalarized objective function $f^s(x)$ used by the single-objective $(1+1)$-EA are, respectively

$$H(v;r) = \prod_{i=1}^{d}[\,r_i - v_i\,] \qquad \text{and} \qquad f^s(x) = \frac{1}{d}\sum_{i=1}^{d} f_i(x)\,.$$

Suppose there are two incomparable points $x, y \in \mathbb{R}^d$ with values $f(x) = (0, 0, \ldots, 0)'$ and $f(y) = (-1, -1, \ldots, -1, d-1+\epsilon)'$ where $\epsilon \in (0,1) \subset \mathbb{R}$. Insertion yields the reference point $r = (1, 1, \ldots, 1, d+\epsilon)$ leading to the dominated hypervolume $H_x = d+\epsilon$ and $H_y = 2^{d-1}$.

The $(1+1)$-SMS-EMOA would accept y if $H_y > H_x$. Notice that $H_y > H_x \Leftrightarrow 2^{d-1} > d+\epsilon$ is true for $d \geq 3$. But the (1+1)-EA would reject y since

$$f^s(y) = \frac{1}{d}\,[(d-1)\cdot(-1) + d - 1 + \epsilon] = \frac{\epsilon}{d} \;>\; 0 \;=\; f^s(x)\,.$$

As a consequence, both algorithms are not algorithmically equivalent for $d \geq 3$ in case of a uniformly weighted scalarized objective function. □

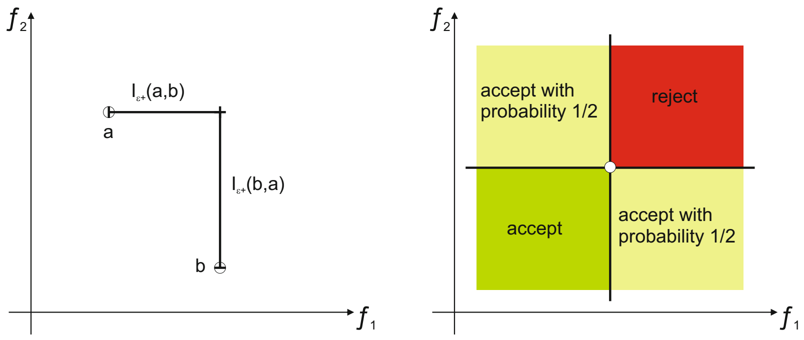


Fig. 2. Left: Values of the additive ϵ indicator $I_{\epsilon+}$ correspond to the hypervolume contributions in the (1+1)-SMS-EMOA. Right: Regions of acceptance, rejection, or random acceptance in case of incomparable points for (1+1)-NSGA-II, (1+1)-SPEA2, and the simple (1+1)-EA from [1]

3.3 (1+1)-IBEA Using the Additive ϵ-Indicator

We show that a (1+1)-IBEA [10] selecting according to the additive ϵ-indicator $I_{\epsilon+}$ [7] performs equal to the (1+1)-SMS-EMOA for two objectives. IBEA$_{\epsilon+}$ prefers non-dominated individuals over dominated ones, so for the case of two comparable individuals, the behavior of acceptance and rejection is clearly equal to the one of the SMS-EMOA. For incomparable individuals, the indicator $I_{\epsilon+}$ comes into play, which is a relative binary indicator, originally defined on two sets of points. For two points, $I_{\epsilon+}(a, b)$ calculates the minimal distance ϵ by which a can be moved in each direction until it is weakly dominated by b.

$$I_{\epsilon+}(a, b) = \min_{\epsilon}\{\forall i \in \{1, \ldots, d\} : f_i(a) + \epsilon \geq f_i(b)\} \tag{3}$$

Obviously large values correspond to valuable individuals, analogously to the hypervolume (contribution). The hypervolume contribution has been shown to reduces to a distance for the (1+1)-SMS-EMOA. This distance is exactly equal to the value of the $I_{\epsilon+}$ (cf. Fig. 2, left). Thus it directly follows:

Corollary 2. *Theorem 2 and Corollary 1 hold as well for the (1+1)-IBEA$_{\epsilon+}$ which selects according to the additive ϵ-indicator $I_{\epsilon+}$.*

4 Sub-linear Convergence Rates

We investigate whether the equivalence of (1+1)-SMS-EMOA and IBEA$_{\epsilon+}$ to the (1+1)-EA is outstanding. To this end, we consider further popular EMOA in the version of using a (1+1) selection scheme.

NSGA-II [9] has been developed with a (μ+μ) selection, and is thus considered for $\mu = 1$. The selection starts by performing non-dominated sorting on the set of parent and offspring. If the individuals are comparable, the dominating one is kept and the dominated one discarded. In case of incomparable individuals the crowding distance is invoked. It rewards individuals with a large distance to their neighbors, and assigns a value of infinity to points at the boundary of the non-dominated front, i.e. those not having neighbors in one dimension. Here, both points are boundary points with equal crowding distance values. Thus, one is chosen to be discarded uniformly at random, so in case of incomparable points, each is accepted with probability 1/2 (cf. Fig. 2, right).

The same result holds for the (1+1)-SPEA2 [11]. For incomparable individuals, there are neither dominated nor dominating ones, thus the raw fitness of both individuals is zero. So, the secondary indicator based on a k-nearest neighbor method is used. The resulting values for the individuals are equal since they both are their only neighbors and distances are symmetrical.

We declare that both algorithms in their (1+1) version are equal to the EMOA considered by Rudolph [1]: Recall that this (1+1)-EA chooses uniformly at random one fitness function for selection. The better individual w.r.t. to the function is kept, the other one discarded. Two incomparable individuals have both worst and best values in interchanged functions. So, choosing a function is equivalent to choosing the preferred individual. Since [1] proves that convergence is given but only with a sub-linear rate for at least one instance from the problem class, we immediately get the following result.

Theorem 3. *The (1+1)-NSGA-II and the (1+1)-SPEA2 have sub-linear convergence rate under conditions for the step sizes given in [1].* □

It is still unclear how this step size rule can be realized in practice and, thus, whether NSGA-II and SPEA2 converge at all for any other known mutation operator. Nevertheless, our result indicates that sub-linear convergence might be the best one can hope for.

5 Conclusions

We showed that the (1+1) versions of SMS-EMOA and IBEA$_{\epsilon+}$ have linear convergence rate on the class of bi-objective problems whose functions all are quadratically convex with at least one being strongly convex. This is the first time that a linear convergence rate could be proved for evolutionary multiobjective optimization algorithms that do not require an explicit weighting of objectives. The convergence rate is proved by reduction to an already analyzed single-objective (1+1)-EA with self-adaptation. The equivalence of the algorithms holds for arbitrary bi-objective problems, and a result regarding the linear convergence rate on a certain class of functions is transferred to the EMOA. By a counter example it is shown that the selection behavior of the EMOA is no longer equal to the single-objective (1+1)-EA for more than two objectives.

(1+1)-NSGA-II and (1+1)-SPEA2 have a sub-linear convergence rate on the considered class of functions due to the fact that their selection operators degenerate to random choice among incomparable individuals.

Future research shall consider how population sizes greater than one influence the convergence properties of different evolutionary multiobjective optimization algorithms.

References

1. Rudolph, G.: On a multi–objective evolutionary algorithm and its convergence to the Pareto set. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, pp. 511–516. IEEE Press, Piscataway (1998)
2. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3), 553–564 (1999)
3. Teytaud, O.: On the hardness of offline multi-objective optimization. Evolutionary Computation 15(4), 475–491 (2007)
4. Jägersküpper, J.: How the (1+1) ES using isotropic mutations minimizes positive definite quadratic forms. Theoretical Computer Science 361(1), 38–56 (2006)
5. Jägersküpper, J.: Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. Theoretical Computer Science 379(3), 329–347 (2007)
6. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms – a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–304. Springer, Heidelberg (1998)
7. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
8. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181(3), 1653–1669 (2007)
9. 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)
10. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. 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. 832–842. Springer, Heidelberg (2004)
11. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), CIMNE, pp. 95–100 (2002)

Tight Bounds for the Approximation Ratio of the Hypervolume Indicator

Karl Bringmann[1] and Tobias Friedrich[2]

[1] Universität des Saarlandes, Saarbrücken, Germany
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. The hypervolume indicator is widely used to guide the search and to evaluate the performance of evolutionary multi-objective optimization algorithms. It measures the volume of the dominated portion of the objective space which is considered to give a good approximation of the Pareto front. There is surprisingly little theoretically known about the quality of this approximation. We examine the multiplicative approximation ratio achieved by two-dimensional sets maximizing the hypervolume indicator and prove that it deviates significantly from the optimal approximation ratio. This provable gap is even exponential in the ratio between the largest and the smallest value of the front. We also examine the additive approximation ratio of the hypervolume indicator and prove that it achieves the optimal additive approximation ratio apart from a small factor $\leqslant n/(n-2)$, where n is the size of the population. Hence the hypervolume indicator can be used to achieve a very good additive but not a good multiplicative approximation of a Pareto front.

1 Introduction

Most real-world optimization problems have to deal with multiple objectives (like time vs. cost) and cannot be easily described by a single objective function. This implies that there is in general no unique optimum, but a possibly very large set of incomparable solutions which forms a Pareto front. Many different multi-objective evolutionary algorithms (MOEAs) have been developed to find a Pareto set of (preferably small) size n which gives a *good approximation* of the Pareto front. A popular way to measure the quality the approximation is the hypervolume indicator. It measures the volume of the dominated space [18]. For a small number of objective, MOEAs which directly use the hypervolume indicator to guide the search are the methods of choice. These include for example the generational MO-CMA-ES [8, 16], the SMS-EMOA [3, 6], and variants of IBEA [17, 19].

One of the reasons why the hypervolume indicator is so popular is that it matches very well with our *intuition* how a good approximation of a Pareto front should look like. However, there is only little known whether maximizing the hypervolume also gives a good approximation of the Pareto front in *strictly mathematical sense.* Considering the wide use of the hypervolume indicator, the question whether it achieves a good approximation appears to be fundamental. The distribution of the points maximizing the hypervolume indicator has been examined by several authors. It was observed that "convex regions may be preferred to concave regions" [13, 18] as well as that HYP is "biased towards the boundary solutions" [5]. In contrast to this, such sets are empirically "well distributed" according to [6, 9, 10] and it was also proven that for the number of points $n \to \infty$ the density of points only depends on the gradient [2].

However, the question whether sets maximizing the hypervolume give an approximation of the Pareto front in the mathematical sense remained open besides two preliminary papers [4, 7]. We follow up on this and study the approximation quality of the hypervolume indicator by classic approximation theory. Which concept from approximation theory is the right measure depends on the problem at hand. As a general rule of thumb, for linear axes this is the additive approximation ratio while for exponential axes this is the multiplicative approximation ratio.

To illustrate this with a small example, consider a knapsack problem (see e.g. [18]) with linearly distributed weights and exponentially distributed profits. In this case a good approximation of the front should be an additive approximation of the weights and a multiplicative approximation of the profits. Within this example the result of this paper is that compared to the optimal set with best possible approximation, sets maximizing the hypervolume only achieve the first aim, not necessarily the latter.

In our previous paper [4], we proved that for all possible Pareto fronts the multiplicative approximation factor achieved by a set of n solutions maximizing the hypervolume indicator is $1+\Theta(1/n)$ (cf. Theorem 3.6)[1]. As this was shown to be *asymptotically equivalent* to the optimal multiplicative approximation factor (cf. Corollary 3.4), we concluded that the hypervolume indicator is guiding the search in the correct direction for sufficiently large n. However, the size n of a population is usually not large. Also, the constant factors hidden by the Θ might still be larger for the set maximizing hypervolume compared to the set with best possible approximation factor.

Our Results

We significantly extend the results of [4]. First, we are now able to give tight bounds on the multiplicative approximation ratio depending on the ratio A/a between the largest and smallest coordinate[2]. Using this notation, the precise result of [4] is the computation of the optimal multiplicative approximation ratio as $1 + \log(A/a)/n$ (cf. Corollary 3.4). We are now able to show that the multiplicative approximation ratio for a set maximizing the hypervolume is strictly

[1] The precise statements of this and the following results of this introduction are slightly more technical. For details see the respective theorems.

[2] The approximation ratio actually depends on the ratios in both dimensions. To simplify the presentation in this introduction, we assume here that the ratio A/a in the first dimension is equal to the ratio B/b in the second dimension.

larger, namely of the order of *at least* $1 + \sqrt{A/a}/n$ (cf. Theorem 3.7). This implies that the dependence of this multiplicative approximation ratio on the ratio A/a can be exponentially worse than in the optimal case. Hence for numerically very wide spread fronts (that is, large A/a) there are Pareto sets which give a much better multiplicative approximation than the Pareto sets which maximize the hypervolume.

Second, we now also analyze the additive approximation ratio of the hypervolume indicator. While the multiplicative approximation factor is determined by the ratio A/a, the additive approximation factor is determined by the width of the domain $A - a$. We prove that the optimal additive approximation ratio is $(A - a)/n$ (cf. Theorem 4.3) and upper bound the additive approximation ratio achieved by a set maximizing the hypervolume by $(A - a)/(n - 2)$ (cf. Theorem 4.5). This is a very strong statement, as apart from the small factor $n/(n-2)$ the additive approximation ratio achieved when maximizing the hypervolume is optimal! This shows that the hypervolume indicator yields a much *better additive than multiplicative* approximation.

2 Preliminaries

We only consider the case of two objectives where there is a mapping from an arbitrary search space to an objective space which is a subset of $\mathbb{R}^2$. Throughout this paper, we will only work on the objective space. For points from the objective space we define the following dominance relation:

$$(x_1, y_1) \preceq (x_2, y_2) \text{ iff } x_1 \leqslant x_2 \text{ and } y_1 \leqslant y_2,$$
$$(x_1, y_1) \prec (x_2, y_2) \text{ iff } (x_1, y_1) \preceq (x_2, y_2) \text{ and } (x_1, y_1) \neq (x_2, y_2).$$

We restrict ourselves to Pareto fronts that can be written as $\{(x, f(x)) \mid x \in [a, A]\}$ where $f\colon [a, A] \to [b, B]$ is a (not necessarily strictly) monotonically decreasing, upper semi-continuous[3] function with $f(a) = B$, $f(A) = b$ for $a < A$, $b < B$. We write $\mathcal{F} = \mathcal{F}_{[a,A]\to[b,B]}$ for the set of all such functions f. We will use the term *front* for both, the set of points $\{(x, f(x)) \mid x \in [a, A]\}$, and the function f.

The condition of f being upper semi-continuous cannot be relaxed further as without it the f lacks symmetry in the two objectives in the following sense: Being upper semi-continuous is necessary and sufficient for the existence of the inverse function $f^{-1}\colon [b, B] \to [a, A]$ defined by setting $f^{-1}(y) := \max\{x \in [a, A] \mid f(x) \geqslant y\}$. Without upper semi-continuity this maximum does not exist in general. Furthermore, this condition implies that there is a set maximizing the hypervolume indicator.

Note that the set $\mathcal{F}$ of fronts we consider is a very general one. Most papers that theoretically examine the hypervolume indicator assume that the front is

[3] Semi-continuity is a weaker property than normal continuity. A function f is said to be upper semi-continuous if for all points x of its domain, $\limsup_{y\to x} f(y) \leqslant f(x)$. Intuitively speaking this means that for all points x the function values for arguments near x are either close to $f(x)$ or less than $f(x)$. For more details see e.g. [15].

continuous and differentiable (e.g. [1, 2, 7]), and are thus not able to give results about discrete fronts, which we can.

Let $n \in \mathbb{N}$. For fixed $[a, A], [b, B] \subset \mathbb{R}$ we call a set $P = \{p_1, \dots, p_n\} \subset [a, A] \times [b, B]$ a *solution set* (of size n) and write $\mathcal{P} = \mathcal{P}_n$ for the set of all such solution sets. A solution set P is said to be *feasible* for a front $f \in \mathcal{F}$, if $y \leqslant f(x)$ for all $p = (x, y) \in P$. We write $\mathcal{P}^f \subseteq \mathcal{P}$ for the set of all solution sets that are feasible for f.

We are now ready to formally define the hypervolume indicator. It was first introduced for performance assessment in multiobjective optimization by [18], but since then also has become a very popular way to guide the search in multi-objective evolutionary optimizers. On a two-dimensional objective space it is defined as follows.

Definition 2.1. *The hypervolume indicator* $\mathrm{HYP}(P)$ *of a solution set* $P \in \mathcal{P}$ *relative to a reference point* $R = (R_x, R_y)$ *is*

$$\mathrm{HYP}(P) := \textsc{vol}\left(\bigcup_{(x,y)\in P} [R_x, x] \times [R_y, y] \right).$$

with $\textsc{vol}(\,\cdot\,)$ *being the usual Lebesgue measure.*

3 Multiplicative Approximation

The standard measure of approximation quality in approximation theory is the multiplicative approximation ratio. We use the multi-objective definition for the multiplicative approximation ratio by [14] which was also used in [4, 7, 11, 12]. Note that here and in the rest of the paper when talking about multiplicative approximation we require $a, b > 0$ as this ratio only makes sense for positive values.

Definition 3.1. *Let* $f \in \mathcal{F}$ *and* $P \in \mathcal{P}^f$. *The solution set* P *is a* multiplicative α-approximation *of* f *if for each* $\hat{x} \in [a, A]$ *there is a point* $p = (x, y) \in P$ *with*

$$\hat{x} \leqslant \alpha\, x \quad \textit{and} \quad f(\hat{x}) \leqslant \alpha\, y$$

where $\alpha \in \mathbb{R}$, $\alpha \geqslant 1$. *The* multiplicative approximation ratio *of* P *with respect to* f *is then defined as*

$$\alpha^*(f, P) := \inf\{\alpha \in \mathbb{R} \mid P \textit{ is a multiplicative } \alpha\textit{-approximation of } f\}.$$

The quality of an algorithm which calculates a solution set of size n for each Pareto front in $\mathcal{F}$ has to be compared with the respective optimal approximation ratio defined as follows.

Definition 3.2. *For fixed* $[a, A]$, $[b, B]$, *and* n, *let*

$$\alpha^*_{OPT} := \sup_{f\in\mathcal{F}} \inf_{P\in\mathcal{P}^f} \alpha^*(f, P).$$

The value α^*_{OPT} is chosen such that every front in $\mathcal{F}$ can be approximated by n points to a ratio of α^*_{OPT}, and there is a front which cannot be approximated better. In [4] the authors showed the following two results.

Theorem 3.3 (from [4]). $\alpha^*_{OPT} = \min\{A/a, B/b\}^{1/n}$.

Corollary 3.4 (from [4]). *For all* $n \geqslant \log(\min\{A/a, B/b\})/\varepsilon$ *and* $\varepsilon \in (0,1)$,

$$\alpha^*_{OPT} \geqslant 1 + \frac{\log(\min\{A/a, B/b\})}{n},$$
$$\alpha^*_{OPT} \leqslant 1 + (1+\varepsilon)\,\frac{\log(\min\{A/a, B/b\})}{n}.$$

We further want to measure the approximation of the solution set of size n maximizing HYP. As there might be several solution sets maximizing HYP, we consider the worst case and use the following definition.

Definition 3.5. *For fixed* $[a, A]$, $[b, B]$, *and* n, *let*

$$\mathcal{P}^f_{HYP} := \left\{P \in \mathcal{P}^f \;\middle|\; \mathrm{HYP}(P) = \max_{Q \in \mathcal{P}^f} \mathrm{HYP}(Q)\right\} \text{ for } f \in \mathcal{F}, \text{ and}$$
$$\alpha^*_{HYP} := \sup_{f \in \mathcal{F}} \sup_{P \in \mathcal{P}^f_{HYP}} \alpha^*(f, P).$$

The set $\mathcal{P}^f_{HYP}$ is the set of all feasible solution sets that maximize HYP on f. The value α^*_{HYP} is chosen such that for every front f in $\mathcal{F}$ every solution set maximizing HYP approximates f by a ratio of at most α^*_{HYP}. Note that this assumes that there is at least one solution set which maximizes the indicator, i.e., the set $\mathcal{P}^f_{HYP}$ is non-empty. That this is indeed the case was proven in [4].

In [4] the authors also examined α^*_{HYP} and showed an upper bound that has the same asymptotic behavior as α^*_{OPT}, but a much larger constant factor.

Theorem 3.6 (from [4]). *Let* $f \in \mathcal{F}$, $n > 4$, *and let* $R = (R_x, R_y) \preceq (0,0)$ *be the reference point. If we have*

- $n \geqslant 2 + \max\left\{\sqrt{A/a}, \sqrt{B/b}\right\}$ or
- $R_x \leqslant -\sqrt{A\,a}/n$, $R_y \leqslant -\sqrt{B\,b}/n$,

then

$$\alpha^*_{HYP} \leqslant 1 + \frac{\sqrt{A/a} + \sqrt{B/b}}{n-4}.$$

The previous paper [4] left open whether (i) the upper bound of Theorem 3.6 is not tight and α^*_{HYP} is actually much closer to the bounds for α^*_{OPT} given in Corollary 3.4 or (ii) α^*_{HYP} is indeed significantly larger than α^*_{OPT}. By giving a lower bound for α^*_{HYP} we can now prove the latter. In the following theorem we restrict ourselves to the case of $A/a = B/b$. We show that in this situation the bound of Theorem 3.6 is tight except for a small constant factor.

Theorem 3.7. *Let* $n \geqslant 4$, $A/a = B/b \geqslant 13$, *and* $R = (R_x, R_y) \preceq (0,0)$ *be the reference point. Then*

$$\alpha^*_{HYP} \geqslant 1 + \frac{2\sqrt{A/a - 1}}{3\,(n-1)}.$$

The proof of this theorem will be provided in the full version of the paper.

4 Additive Approximation

After the previous section showed that sets maximizing the hypervolume have sub-optimal multiplicative approximation ratio we now analyze their additive approximation properties. Analogous to Definition 3.1 we use the following definition.

Definition 4.1. *Let $f \in \mathcal{F}$ and $P \in \mathcal{P}^f$. The solution set P is an* additive α-approximation *of f if for each $\hat{x} \in [a, A]$ there is a point $p = (x, y) \in P$ with*

$$\hat{x} \leqslant x + \alpha \quad \textit{and} \quad f(\hat{x}) \leqslant y + \alpha$$

where $\alpha \in \mathbb{R}$, $\alpha \geqslant 0$. The additive approximation ratio *of P with respect to f is defined as*

$$\alpha^{+}(f, P) := \inf\{\alpha \in \mathbb{R} \mid P \textit{ is an additive } \alpha\textit{-approximation of } f\}.$$

Again, we are interested in the optimal approximation ratio for Pareto fronts in $\mathcal{F}$. Analogous to Definition 3.2 we give the following definition.

Definition 4.2. *For fixed $[a, A]$, $[b, B]$, and n, let*

$$\alpha^{+}_{OPT} := \sup_{f \in \mathcal{F}} \inf_{P \in \mathcal{P}^f} \alpha^{+}(f, P).$$

Analogously to the precise bound $\alpha^{*}_{OPT} = \min\{A/a, B/b\}^{1/n}$ of Theorem 3.3 for the optimal multiplicative approximation ratio, we can prove the following for the optimal additive approximation ratio α^{+}_{OPT}.

Theorem 4.3. $\alpha^{+}_{OPT} = \dfrac{\min\{A - a, B - b\}}{n}$.

The proof of Theorem 4.3 will be provided in the full version of the paper.

In order to compare the optimal additive approximation ratio with the approximation ratio achieved by the hypervolume, we give the following definition analogously to the definition of α^{*}_{HYP} in Definition 3.5.

Definition 4.4. *For fixed $[a, A], [b, B], n$, and $f \in \mathcal{F}$ let*

$$\alpha^{+}_{HYP} := \sup_{f \in \mathcal{F}} \sup_{P \in \mathcal{P}^f_{HYP}} \alpha^{+}(f, P).$$

We can now state the main result of this paper that α^{+}_{HYP} is very close to α^{+}_{OPT}. Similar to the proof of the upper bound for α^{*}_{HYP} of Theorem 3.6 we can prove the following upper bound for α^{+}_{HYP}.

Theorem 4.5. *For all $n > 2$ and $(n - 2)\min\{a - R_x, b - R_y\} \geqslant \sqrt{(A - a)(B - b)}$,*

$$\alpha^{+}_{HYP} \leqslant \frac{\sqrt{(A - a)(B - b)}}{n - 2}.$$

Let us briefly discuss the result before the theorem will be proven in the remainder of this section. First note that the precondition is fulfilled if n is large enough *or* if the reference point is sufficiently far away from (a, b). Hence this is no real restriction. Moreover, compare this result to the bound for the optimal additive approximation ratio of Theorem 4.3. This shows that for $A - a \approx B - b$ and moderately sized n, α^{+}_{HYP} is very close to α^{+}_{OPT}. More precisely, for $A - a \ll B - b$ (or $A - a \gg B - b$) the constant in Theorem 4.5 is the geometric mean of $A - a$ and $B - b$ while in Theorem 4.3 it is instead the minimum of both. As there is a provable gap of log vs. square root of A/a for the multiplicative approximation ratio, this proves that HYP yields a much better additive approximation than a multiplicative one.

Proof of Theorem 4.5. Let P be a solution set maximizing HYP on a front $f \in \mathcal{F}$, i.e., $P \in \mathcal{P}^{f}_{HYP}$. Assume that there are points $p, q \in P$ with $p \prec q$. Such a "redundant" set can maximize HYP only on degenerate fronts: If there is a point $r = (x, f(x))$ on the front which is not dominated by any point in P, then[4] $P' := P + r - p$ would have $\mathrm{HYP}(P') > \mathrm{HYP}(P)$, as it dominates all the space P dominates united with the space r dominates. Thus, there is no such point r and P dominates already the whole front. In this case the approximation ratio $\alpha^{+}(f, P) = 1$ and the inequality we want to show holds trivially. This can only happen for f being a step function with less than n steps.

Hence, for the rest of the proof we can assume that there are no points $p, q \in P$ with $p \prec q$. Then we can write $P = \{p_1, \ldots, p_n\}$, $p_i = (x_i, y_i)$ with $a \leqslant x_1 < \ldots < x_n \leqslant A$ and $B \geqslant y_1 > \ldots > y_n \geqslant b$. Furthermore, we can assume that $y_i = f(x_i)$ as otherwise $P - p_i + p'_i$ with $p'_i = (x_i, f(x_i))$ would have a larger hypervolume than P (this uses that the points in P are mutually non-dominating).

We want to argue about the contribution of a point p to the hypervolume of a solution set P, namely $\mathrm{CON}_P(p) := \mathrm{HYP}(P) - \mathrm{HYP}(P - p)$. In particular we need the minimal contribution of any of the points $p_2, \ldots, p_{n-1}$:

$$\begin{aligned}\textsc{MinCon}(P) &:= \min_{1<i<n} \mathrm{CON}_P(p_i)\\ &= \min_{1<i<n} (x_i - x_{i-1})\,(f(x_i) - f(x_{i+1})).\end{aligned}$$

This value has been (with slightly different notation) examined in [4]. In particular, the authors showed that for $n > 2$

$$\textsc{MinCon}(P) \leqslant \frac{(x_n - x_1)\,(f(x_1) - f(x_n))}{(n-2)^2}.$$

This implies

$$\textsc{MinCon}(P) \leqslant \frac{(A - a)\,(B - b)}{(n-2)^2}. \tag{1}$$

[4] To increase the readability, for a set $P \subset \mathbb{R}^2$ and a point $r \in \mathbb{R}^2$ we define $P + r := P \cup \{r\}$ and $P - r := P \setminus \{r\}$ here and in the remainder of this section.

Table 1. Results for the optimal approximation ratio and upper bounds for the approximation ratios of HYP. See the cited theorems for the precise statements.

	Multiplicative approximation	**Additive approximation**
OPT	$1+\frac{\log(\min\{A/a, B/b\})}{n}$ (Cor. 3.4)	$\frac{\min\{A-a, B-b\}}{n}$ (Thm. 4.3)
HYP	$1+\frac{\sqrt{A/a}+\sqrt{B/b}}{n-4}$ (Thm. 3.6)	$\frac{\sqrt{(A-a)\,(B-b)}}{n-2}$ (Thm. 4.5)

Let $r = (x, f(x))$, $x \in [a, A]$ be an arbitrary point and let $\alpha > 0$ be such that r it not additively approximated by α. We make a case distinction depending on the position of r. Let us first assume that r is an "inner point", i.e., there is an $i \in \{1, \ldots, n-1\}$ with $x_i \leqslant x < x_{i+1}$. As r is not additively approximated by α, we have

$$x > x_i + \alpha \quad \text{and} \quad f(x) > f(x_{i+1}) + \alpha. \tag{2}$$

As P maximizes the hypervolume indicator on f, replacing the point $p \in P$ contributing $\textsc{MinCon}(P)$ to P by the point r must not increase the hypervolume. Therefore,

$$\begin{aligned}\mathrm{HYP}(P) \geqslant \mathrm{HYP}(P+r-p) &= \mathrm{HYP}(P) - \mathrm{CON}_P(p) + \mathrm{CON}_{P+r-p}(r)\\ &\geqslant \mathrm{HYP}(P) - \mathrm{CON}_P(p) + \mathrm{CON}_{P+r}(r),\end{aligned}$$

which in turn implies

$$\textsc{MinCon}(P) = \mathrm{CON}_P(p) \geqslant \mathrm{CON}_{P+r}(r) = (x - x_i)\,(f(x) - f(x_{i+1})) \overset{(2)}{>} \alpha^2.$$

Using equation (1) and taking square roots on both sides gives the desired

$$\alpha < \frac{\sqrt{(A-a)\,(B-b)}}{n-2}.$$

It remains to study the case where $r = (x, f(x))$ is an "outer point" with $x \leqslant x_1$ or $x \geqslant x_n$. It suffices to examine $x \leqslant x_1$ as then the case $x \geqslant x_n$ follows by symmetry in the two objectives.

As r is not approximated by a ratio of α we have $f(x) > f(x_1) + \alpha$. Additionally, replacing the point $p \in P$ contributing $\textsc{MinCon}(P)$ to P by r must not increase the hypervolume, so we have

$$\begin{aligned}\textsc{MinCon}(P) \geqslant \mathrm{CON}_{P+r-p}(r) \geqslant \mathrm{CON}_{P+r}(r) &= (a - R_x)\,(f(x) - f(x_1))\\ &\geqslant (a - R_x)\alpha.\end{aligned}$$

We use equation (1) again and get

$$\alpha \leqslant \frac{(A-a)\,(B-b)}{(a-R_x)\,(n-2)^2} \leqslant \sqrt{(A-a)\,(B-b)}/(n-2),$$

where the second inequality follows from the precondition of the theorem. □

5 Conclusion

Many modern MOEA use the hypervolume indicator to guide the search process. We presented a mathematically rigorous framework to analyze the approximation ratio achieved by sets maximizing the hypervolume. We prove that sets maximizing HYP do *not* give a perfect multiplicative approximation. The proven bounds can be found in Table 1. The multiplicative approximation ratio of HYP is getting large for numerically wide spread fronts with large A/a. On the other hand, we can prove that maximizing HYP gives a close-to-optimal additive approximation.

References

[1] Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Investigating and exploiting the bias of the weighted hypervolume to articulate user preferences. In: Proc. 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 563–570 (2009)
[2] Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point. In: Proc. 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009), pp. 87–102 (2009)
[3] Beume, N., Naujoks, B., Emmerich, M.T.M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181, 1653–1669 (2007)
[4] Bringmann, K., Friedrich, T.: The maximum hypervolume set yields near-optimal approximation. In: Proc. 12th Annual Conference on Genetic and Evolutionary Computation (GECCO 2010), pp. 511–518 (2010)
[5] Deb, K., Mohan, M., Mishra, S.: Evaluating the ε–domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evolutionary Computation 13, 501–525 (2005)
[6] Emmerich, M.T.M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
[7] Friedrich, T., Horoba, C., Neumann, F.: Multiplicative approximations and the hypervolume indicator. In: Proc. 11th Annual Conference on Genetic and Evolutionary Computation (GECCO 2009), pp. 571–578 (2009)
[8] Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation 15, 1–28 (2007)
[9] Knowles, J.D., Corne, D.: Properties of an adaptive archiving algorithm for storing nondominated vectors. IEEE Trans. Evolutionary Computation 7, 100–116 (2003)
[10] Knowles, J.D., Corne, D.W., Fleischer, M.: Bounded archiving using the Lebesgue measure. In: Proc. IEEE Congress on Evolutionary Computation (CEC 2003), vol. 4, pp. 2490–2497.
[11] Kumar, R., Banerjee, N.: Running time analysis of a multiobjective evolutionary algorithm on simple and hard problems. In: Wright, A.H., Vose, M.D., De Jong, K.A., Schmitt, L.M. (eds.) FOGA 2005. LNCS, vol. 3469, pp. 112–131. Springer, Heidelberg (2005)

[12] Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary Computation 10(3), 263–282 (2002)

[13] Lizarraga-Lizarraga, G., Hernandez-Aguirre, A., Botello-Rionda, S.: G-metric: an m-ary quality indicator for the evaluation of non-dominated sets. In: Proc. 10th Annual Conference on Genetic and Evolutionary Computation (GECCO 2008), pp. 665–672 (2008)

[14] Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources. In: Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS 2000), pp. 86–92 (2000)

[15] Rudin, W.: Real and complex analysis, 3rd edn. McGraw-Hill Book Co., New York (1987)

[16] Suttorp, T., Hansen, N., Igel, C.: Efficient covariance matrix update for variable metric evolution strategies. Machine Learning 75, 167–197 (2009)

[17] Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. 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. 832–842. Springer, Heidelberg (2004)

[18] Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evolutionary Computation 3, 257–271 (1999)

[19] Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)

Evolutionary Multiobjective Optimization Algorithm as a Markov System

Ewa Gajda[1], Robert Schaefer[1], and Maciej Smołka[2]

[1] AGH University of Science and Technology, Department of Computer Science
Al. Mickiewicza 30, 30-059 Kraków, Poland
{gajda,schaefer}@agh.edu.pl
http://www.iisg.agh.edu.pl/

[2] Jagiellonian University, Institute of Computer Science
ul. Łojasiewicza 6, 30-348 Kraków, Poland
smolka@ii.uj.edu.pl

Abstract. In the paper we consider the ranking given by the Pareto dominance relation as a basis to create a selection operator for the Evolutionary Multiobjective Optimization Algorithm (EMOA). Assuming that sampling to the next epoch is performed according to the generalized Bernoulli schema with regard to a selected type of the rank selection, a heuristic operator for EMOA is introduced. Having defined the heuristic operator, the transition probability matrix of the uniform Markov chain modeling EMOA can be explicitly obtained as in the Vose's theory of the Simple Genetic Algorithm (SGA). This chain is ergodic if the mixing operator following the EMOA selection operator in each epoch is strictly positive. Moreover, we show that the measure on the space of populations imposed by the EMOA infinite population concentrates on the set of fixed points of the heuristic operator after infinite number of epochs, assuming that the heuristic operator is focusing.

Keywords: evolutionary algorithm, multi-objective optimization, Markov system.

1 Introduction

Evolutionary Multiobjective Optimization Algorithms (EMOAs) have been studied by several groups of researchers. Different types of selection were introduced i.a. by Goldberg in [5], Fonseca and Fleming in [4] and Zitzler and Thiele in [16]. Nondominated sorting was also used by Srinivas and Deb (see. e.g. [12]). Theoretical properties of EMOAs applied to discrete problems were studied i.a. by Rudolph in [9], [10], Hanne in [6] and Laumanns in [7]. Authors of these papers base on the Markov description of populations processing and use an archive in which an approximation of the Pareto front is stored. Convergence with regard to ε-Pareto dominance relation was analyzed by Laumanns in [7].

We build a Markov model of EMOA basing on the introduced heuristic operator, similar to the Vose's genetic operator for the Simple Genetic Algorithm

(SGA) (see e.g. [13]). We assume the set of genes to be finite but general (individuals are not necessarily strings over any alphabet). The proposed heuristic is created with regard to some special types of rank selection. We will analyze asymptotic features of the evolved population according to such EMOA selection rules and mixing operations (crossover, mutation, etc.) that return a strictly positive sampling probability.

Because the asymptotic results obtained for EMOA Markov model are similar as those obtained for SGA by Vose, they can be used for verifying two-phase strategies in the same manner as for the single criteria ones (see e.g. [11]). Such strategies consist of finding the approximation of the connected components of the Pareto set (using EMOA combined with the proper population clustering) in the first phase, and the parallel, detailed search in each of them.

2 Evolutionary Approach to the Multiobjective Optimization

2.1 Pareto Dominance

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

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

defined over some search space U, which might be implicitly defined by constraints. We assume the search space U to be finite $\#U = r < +\infty$ and that all objectives shall be maximized. Therefore we are interested in solving

$$\max \left\{ f(p) = (f_1(p), \dots, f_k(p))^T \,|\, p \in U \right\}. \tag{2}$$

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

$$f(p) \geq f(q) \text{ and } \exists_{i=1,\dots,k}\ f_i(p) \neq f_i(q). \tag{3}$$

Remark 1. The definition can be easily adapted to the minimization problem, when in formula (3) inequality changes form $\geq$ to $\leq$. It can be also adapted to mixed min-max problems by changing inequalities for certain coordinates representing different objective functions.

2.2 Evolutionary Multiobjective Optimization (EMOA)

One of the possible ways of solving (2) is finding the *Pareto set* $\mathcal{P}$ being the set of non-dominated elements from U and its image $f(\mathcal{P}) \subset [0, M]^k$ called the *Pareto front*.

A popular class of stochastic algorithms designed for finding Pareto set is called Evolutionary Multiobjective Optimization (EMOA) (see e.g. [14]). Their simplest instances operate on the single population being the multiset $P = (U, \eta)$ of the search space members called *individuals*, while U is called now *genetic* *universum*. The occurrence function $\eta : U \rightarrow \mathbb{Z}_+ \cup \{0\}$ returns $\eta(i)$ being the number individuals with the genotype $i \in U$ and $\mu = \sum_{i \in U} \eta(i) < +\infty$ stands for the population cardinality.

As other genetic algorithms, EMOA consists in producing the sequence of populations $\{P^t\}$ in the consecutive *genetic epochs* $t = 1, 2, \ldots$ starting from the population P^0 uniformly sampled from U.

Later, we will consider only the scheme (μ, λ) where $\lambda = \mu$, such that the transformation between P^t and its successor P^{t+1} is obtained by the composition of two groups of random operations: selection and mixing. Hereafter (μ, λ) stands for the Schwefel's symbol, where λ is the offspring cardinality and μ is the population cardinality.

While the mixing operations utilized in EMOA do not differ significantly from those applied in other groups of evolutionary algorithms, the EMOA selection is performed with regard to the Pareto dominance (see e.g. [2]). The algorithm terminates after a predefined number of epochs or when another stopping criterion is satisfied (see e.g. [14]).

2.3 Selection Schemes

Several important EMOAs with different selection schemes will be briefly described in the following section. A comparison of these methods can be found e.g. in [14].

The idea of calculating an individual's fitness according to Pareto-dominance was first suggested by Goldberg in [5]. The procedure of NSGA (*Nondominated Sorting Genetic Algorithm*) is based on ranking individuals in an iterative way: firstly nondominated solutions are assigned rank one and temporarily removed from the population. After that, next nondominated solutions are given rank two and so forth. The rank of an individual determines its fitness value. Goldberg's concept was implemented e.g. by Srinivas and Deb [12].

Fonseca and Fleming in [4] proposed a Pareto-based ranking procedure (*FFGA*), where an individual's rank equals the number of solutions by which it is dominated. After sorting population according to the rank, new fitness values are assigned to individuals by interpolating from the best (with the lowest rank) to the worst (with the highest rank) according to some function. Fitness of individuals with the same rank should be equal, so that all of them will be sampled at the same rate. We used this type of selection as a basis for creating the selection operator.

Later on we will refer mainly to NSGA and FFGA selection schemes in preparing the EMOA Markov model. In the following paragraphs we will mention two important strategies which seem to be difficult or impossible to model in the way presented in next sections.

One of these methods is aimed to construct an algorithm in which the *hypervolume measure* (see e.g. [3]) governs the selection operator of an EMOA in order to find a set of solutions well distributed on the Pareto front. Hypervolume measure or S-metric corresponds to the size of dominated space [16]. Individuals are rated according to their contribution to the dominated hypervolume of the current population, therefore ranks are not based on relations between pairs of individuals but on relation between an individual and the whole population.

Strength Pareto Evolutionary Algorithm (SPEA, see [16]) uses a regular population and an external set (archive) into which all nondominated solutions are copied in each iteration. If the size of the archive exceeds a predefined limit, further archive members are deleted by a clustering strategy which preserves the characteristics of the nondominated front. Ranks of solutions are calculated basing on strength values of individuals stored externally. SPEA was later improved and introduced as SPEA2 in [15]. The selection in SPEA cannot be described by our selection operator because of the existence of the archive. In order to model this selection scheme one should consider a different space of states.

3 EMOA Markov Model

3.1 Evolutionary Algorithms with Heuristic

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

$$\Lambda^r = \left\{ x = \{x_p\};\ 0 \le x_p \le 1,\ p \in U,\ \sum_{p \in U} x_p = 1 \right\}. \tag{4}$$

Such construction has several advantages:

1. Although the frequency vector represents unambiguously only the finite populations ($\mu < +\infty$), it is possible to represent also the infinite size populations. We will identify the population with its frequency vector if it does not lead to the ambiguity.
2. Each $x \in \Lambda^r$ being the population frequency vector (possibly infinite one) belongs to $\mathcal{M}(U)$ being the set of probabilistic measures on the set U.
3. The set containing representations of all populations is compact in $\mathbb{R}^r$.

The above settings allow to define the class of evolutionary algorithms which are characterized by the same genetic universum U and fitness as well as the same set of genetic operations that do not depend on the genetic epoch number. They can differ only by the population size μ. This class of EA's may be also characterized as "stationary" because the evolutionary rule does not change during computations.

Definition 2. *The mapping $\mathcal{H} \in C(\Lambda^r \to \Lambda^r)$ will be called the heuristic of the particular class of genetic algorithms if:*

1. *Each coordinate $(\mathcal{H}(x))_p$ is equal to the sampling probability of the individual with the genotype $p \in U$ in the epoch that immediately follows the epoch in which the population $x \in \Lambda^r$ appears.*

2. *The value $\mathcal{H}(x)$ is the expected population in the epoch that immediately follows the epoch in which the population $x \in \Lambda^r$ appeared, for all algorithms from the considered class.*
3. *It stands for the law of evolution of the abstract, deterministic, infinite population algorithm (we assume that it exists in the considered class). In other words, the infinite population algorithm is the dynamic system that starts from a particular initial population $x^0 \in \Lambda^r$ and then passes consecutively by $\mathcal{H}(x^0), \mathcal{H}^2(x^0), \mathcal{H}^3(x^0), \dots$.*

If a particular class of genetic algorithms admits a heuristic operator, we will call those algorithms *the genetic algorithms with heuristic.* The heuristic operator was introduced by Vose and his collaborators for the class of Simple Genetic Algorithms (SGA) (see e.g. [13]). This operator was equivalently called *genetic operator* in this case. Some further comments are contained in [11].

Furthermore, we restrict ourselves to the evolutionary algorithms in which the next epoch population $x^{t+1} \in \Lambda^r$ is obtained by the μ-times sampling with return according to the polynomial scheme (generalized Bernoulli scheme, see e.g. Billingsley [1]), assuming the probability distribution on the set of genotypes. Of course, in case of the GA class with heuristic, the value of $\mathcal{H}(x^t)$ stands for such probability distribution.

Observation 1. *If the next population x^{t+1} is obtained using generalized Bernoulli scheme, then the condition 1 of the Definition 2 implies the condition 2.*

The Observation 1 can be motivated as follows. Let us denote by $P^{t+1} = (U, \eta_{t+1})$ the random variable being the population in the $t+1$ epoch. Because P^{t+1} is obtained using the generalized Bernoulli scheme associated with the probability distribution $\mathcal{H}(x^t) \in \mathcal{M}(U)$, we have that $EP^{t+1} = (U, \bar{\eta}_{t+1})$ with $\bar{\eta}_{t+1}(p) = \mu \; \mathcal{H}(x^t)_p$, where E is the proper expected value operator. Therefore the expected coordinate of the frequency vector x^{t+1} satisfies $(Ex^{t+1})_p = \frac{1}{\mu} \; \bar{\eta}_{t+1}(p) = \mathcal{H}(x^t)_p$ for all $p \in U$.

Observation 2. *If the next population x^{t+1} is obtained using generalized Bernoulli scheme, then the condition 1 of the Definition 2 implies also the condition 3.*

The motivation of the Observation 2 in not so trivial as the previous one. It may be drawn from the following theorem.

Theorem 1. $\forall \; k > 0, \;\; \varepsilon > 0, \;\; \nu < 1 \;\; \exists \; N$ *independent upon* $x^0 \in \Lambda^r$ *such that*

$$\mu > N \Rightarrow \Pr\{\left\|x^t - \mathcal{H}^t(x^0)\right\| < \varepsilon\} > \nu \quad \forall \; t \in [0, k] \cap \mathbb{N}.$$

This theorem is a generalization of the well known Nix and Vose result (see Theorem 2 in [8]) proved originally for the SGA heuristic only. This theorem states that the finite population algorithm spends arbitrarily large number of epochs arbitrarily close to the heuristic trajectory with the probability arbitrarily close to 1 if the population size is large enough, so the heuristic trajectory might be understood as the trajectory of infinite population algorithm in this sense.

Observation 3. *If the particular EA has the heuristic $\mathcal{H}$ and the next epoch population P^{t+1} is obtained using generalized Bernoulli scheme associated with the probability distribution $\mathcal{H}(x^t) \in \mathcal{M}(U)$, then it can be modeled as the stationary Markov chain with the finite space of states X_μ and with the transition probability matrix* $\mathbf{Q}$ *given by the formula similar to the formula introduced by Vose for SGA Markov model (see Theorem 1 in [8])*

$$(\mathbf{Q})_{x,y} = \mu! \prod_{p \in U} \frac{(\mathcal{H}(x)_p)^{\mu y_p}}{(\mu y_p)!} \;\; \forall x, y \in X_\mu. \tag{5}$$

The above observation is a simple issue of the polynomial sampling distribution.

3.2 The EMOA Selection Operator

Let us start with the definition of the *binary Pareto dominance matrix*

$$\varXi \in \{0,1\}^r \times \{0,1\}^r; \quad \varXi_{p,q} = \begin{cases} 1 & \text{if } q \succ p \\ 0 & \text{otherwise.} \end{cases}, \;\; \forall\, p, q \in U. \tag{6}$$

which completely characterizes the Pareto dominance relation among the genotypes from U for the particular multiobjective optimization (2). The above definition is appropriate also for different cases of problems, not only maximization (see Remark 1). NSGA selection scheme can be represented in a similar way but with a different $\varXi$ matrix.

It is easy to observe that the p-th entry of the vector $(\varXi\, \eta)$ represents the number of individuals which dominate the individual with the genotype p belonging to the population $P = (U, \eta)$ (i.e. $\eta(p) > 0$).

Next, we introduce function $\xi : \Lambda^r \to [0,1]^r$ of the form

$$\xi(x) = \varXi\, x\,, \tag{7}$$

so that $\xi(x)_p$ defines the rank of all individuals with the genotype $p \in U$ contained in the population P represented by its frequency vector x.

This function is well defined for both finite and infinite populations. In case of finite population of the cardinality $\mu < +\infty$ the entry $\xi(x)_p$ may be interpreted as the relative number of individuals that dominate the individual with the genotype p because $\xi(x) = \frac{1}{\mu}\,(\mu\, \varXi\, x) = \frac{1}{\mu}\,(\varXi\, \mu\, x) = \frac{1}{\mu}\,(\varXi\, \eta)$.

Observation 4. *It may be easily checked that*

$$\forall x \in \Lambda^r\; \exists p \in U :\; \xi(x)_p = 0,\; x_p > 0.$$

It follows from the fact that there is at least one non-dominated individual in each population.

As usual, it is necessary to introduce the *validating function* in order to obtain the probability distribution of the rank selection

$$g \in C([0,1] \to [0,1]);\; \forall \zeta, \gamma \in [0,1],\;\; \zeta > \gamma \Rightarrow g(\zeta) < g(\gamma). \tag{8}$$

As a simple example of a function correlated with the rank-based fitness assignment method [4] we can take

$$g(\zeta) = 1 - \zeta. \tag{9}$$

To obtain more control on the selection pressure the function

$$g(\zeta) = e^{-\alpha\zeta},\ \alpha \in \mathbb{R}_+ \tag{10}$$

may be chosen.

For technical purposes we introduce a next function $G : [0,1]^r \to [0,1]^r$ such that $G(x)_p = g(x_p),\ p \in U$.

The probability of selecting the individual $p \in U$ from the current EMOA population P represented by the vector $x \in \Lambda^r$ equals to

$$\Pr(p) = \frac{1}{x^T\ G(\xi(x))}\, g((\xi(x))_p)\ x_p. \tag{11}$$

We are now ready to define the selection operator $F : \Lambda^r \to \Lambda^r$ for the EMOA rank selection

$$F(x) = \frac{1}{x^T\ G(\Xi\ x)}\ \mathrm{diag}(x)\ G(\Xi\ x)\,, \tag{12}$$

where $\mathrm{diag}(x)$ denotes the $r \times r$ diagonal matrix with the diagonal x.

Observation 5. *Taking into account the features of the function ξ (see Observation 4) and the features of the function g (see formula (8)) the EMOA rank selection operator (12) as well as the formula (11) are well defined, because the denominator $x^T\ G(\Xi\ x)$ is strictly positive for all $x \in \Lambda^r$.*

Observation 6. *Because g is continuous in its domain the EMOA rank selection operator (12) is continuous on the whole Λ^r. If we additionally assume, that g is continuously differentiable, as in case of both examples (9), (10), then the EMOA rank selection operator is also continuously differentiable on the whole Λ^r.*

Observation 7. *If the validating function g is strictly positive, then the EMOA rank selection is always "soft", which means that each individual from the current population can be selected (with the positive probability) as a parental one. If we relax conditions of g contained in (8), assuming only that g is weakly decreasing ($g(x) \leq g(y)$ for $x > y$) and exists $\gamma \in (0,1)$ so that $g(x) > 0$ only for $x \in [0,\gamma]$, then the EMOA rank selection may become partially hard, elitist one. Such a relaxation does not contradict the well-posedness of the formula (12) because the denominator $x^T\ G(\Xi\ x)$ remains strictly positive for all $x \in \Lambda^r$.*

3.3 The EMOA Heuristic

The selection is followed by the genetic operations (e.g. mutation, crossover) in each EMOA epochs. They can be represented by the *mixing operator* $M \in C^1(\Lambda^r \to \Lambda^r)$. Currently, we do not impose any specific restrictions for this mapping.

The well known example of mixing operator was introduced by Vose and collaborators [13]. It expresses the binary mutation and positional crossover utilized in SGA.

$$M(x)_p = (\sigma_p \, x)^T \mathbf{M} \sigma_p \, x, \ \ \forall \, x \in \Lambda^r, \ p \in U \tag{13}$$

where σ_i stands for the $r \times r$ dimension permutation matrix with the entries $(\sigma_p)_{q,k} = [q \oplus k = p], \ p, q, k \in U$. The entries $\mathbf{M}_{p,q}$ of the symmetric $r \times r$ matrix $\mathbf{M}$ express the probability of obtaining the genotype $0 \in U$ (the genotype being the string of zeros) from the parents $p, q \in U$ by the crossover and mutation.

Similarly, like in case of SGA, the composition

$$\mathcal{H} = M \circ F \tag{14}$$

may be considered as the candidate for the heuristic of the particular class of EMOA considered in this paper.

Observation 8. *As an immediate consequence of its construction, $\mathcal{H}$ is well defined and continuous on the whole Λ^r. Assuming additionally that the function g (see formula (8)) is continuously differentiable on $[0,1]$ we have also that $\mathcal{H}$ is continuously differentiable on the whole Λ^r.*

Observation 9. *Again, as the result of the construction, $\mathcal{H}$ described by (14) satisfies the condition 1 of the Definition 2, because each coordinate of its value $(\mathcal{H}(x))_p$ stands for the probability of sampling the genotype $p \in U$ to the population following the population associated with the frequency vector x. If we moreover assume, that the sampling to the next epoch population is performed according to the generalized Bernoulli model (according to the polynomial probability distribution) then also the conditions 2 and 3 of the Definition 2 are satisfied (see Observations 1 and 2).*

Observation 10. *Immediately from the Observations 9 and 3 it follows that the considered class of EMOA transforming the finite populations of the cardinality $\mu < +\infty$ can be modeled as the stationary Markov chain with the finite space of states X_μ and with the transition probability matrix $\mathbf{Q}$ given by the formula (5).*

3.4 Asymptotic Features

Observation 11. *If the mixing operator (13) is strictly positive, e.g. $M(x)_p > 0, \ \forall x \in \Lambda^r, \ \forall p \in U$, then the Markov chain describing EMOA is ergodic. The algorithm possesses the asymptotic guarantee of success, e.g. it will reach the population (state) which contain all points lying in the Pareto set after an infinite number of epochs.*

Let us denote by $\pi_\mu^t \in \mathcal{M}(X_\mu)$ the probability distribution of the random variable representing EMOA population of the size μ after t epochs. Assuming that EMOA is modeled by the ergodic Markov chain, each sequence $\pi_\mu^0, \pi_\mu^1, \dots$ has a strictly positive limit π_μ that does not depend on the initial distribution π_μ^0, which is the simple issue of the ergodic theorem (see e.g. [1]). Measures π_μ originally defined over X_μ can be easily extended to Λ^r.

Next, according to the Prokhorov theorem (see Theorem 29.3 in [1]), the sequence $\{\pi_\mu\}$ has a weak limit π^* in Λ^r, because Λ^r is compact.

Let us assume now, that the EMOA heuristic is *focusing* i.e. if for all $x \in \Lambda^r$ the sequence $\{\mathcal{H}^t(x)\}$ converges in Λ^r for $t \to +\infty$. Let $w \in \Lambda^r$ be the limit of such sequence for some starting point $x \in \Lambda^r$. The continuity of $\mathcal{H}$ guarantees, that $\mathcal{H}(w) = \mathcal{H}\left(\lim_{t\to+\infty} \mathcal{H}^t(x)\right) = \lim_{t\to+\infty} \mathcal{H}^{t+1}(x) = w$, so w is the fixed point of $\mathcal{H}$. Let us denote the set of fixed points of $\mathcal{H}$ by $\mathcal{K} \subset \Lambda^r$. If $\mathcal{H}$ is focusing, then obviously $\mathcal{K} \neq \emptyset$.

Theorem 2. *Assuming that the EMOA heuristic is focusing and the Markov chains associated with family of EMOA with various population sizes are ergodic, we obtain* $\pi^*(\mathcal{K}) = 1$.

The above theorem is a formal extension of the well known Vose and Nix result (see Theorem 3 in [8]) and it can be proved in an analogous way.

4 Conclusions and Further Research

– EMOA can be modeled as the ergodic Markov chain given some reasonable assumptions upon the type of selection and the presence of mutation in the mixing step of each epoch.
– Alternative selection types might be considered and formalized. The main step to adapt the current model to other EMOA selection schemes will consist of redefining the Pareto dominance matrix (6).
– In the proposed model a particular form of genotypes has not been assumed: the most common form of strings over an alphabet is appropriate but genotypes may be graphs as well.
– It was also proved that EMOA has the heuristic which fixed points are the only ones visited by the infinite population algorithm (see Theorem 2). It is possible to prove the theorem of a fixed point approximation and the theorem of the convergence of sampling measures, similar to those proved for SGA (see Theorems 4.54 and 4.66 in [11]).
– Furthermore, these results might be used for verifying two-phase strategies in the same manner as for the single criteria ones (see e.g. [11]). Such strategies consist in finding the approximation of the connected components of the Pareto set (using EMOA combined with the proper population clustering) in the first phase and the parallel, detailed search in each of them. It seems that the obtained results might also be useful in the analysis of the ε-Pareto dominance problem (see [7]).

References

1. Billingsley, P.: Probability and Measure. John Wiley and Sons, Chichester (1979)
2. Coello Coello, C.A., Lamont, G.B.: Applications of Multi-objective Evolutionary Algorithms. World Scientific, Singapore (2004)

3. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
4. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Genetic Algorithms: Proceedings of the Fifth International Conference, pp. 416–423 (1993)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learnings. Addison-Wesley, Reading (1989)
6. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. European Journal of Operational Research 117(3), 553–564 (1999)
7. Laumanns, M.: Stochastic convergence of random search to fixed size Pareto set approximations (2007)
8. Nix, A.E., Vose, M.D.: Modelling genetic algorithms with Markov chains. Annals of Mathematics and Artificial Intelligence 5, 79–88 (1992)
9. Rudolph, G.: On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In: Proceedings of the 5th IEEE Conference on Evolutionary Computation, pp. 511–516. IEEE Press, Los Alamitos (1998)
10. Rudolph, G., Agapie, A.: Convergence properties of some multi-objective evolutionary algorithms. In: Congress on Evolutionary Computation (CEC 2000), pp. 1010–1016. IEEE Press, Los Alamitos (2000)
11. Schaefer, R., Telega, H.: Foundation of Global Genetic Optimization. Springer, Heidelberg (2007)
12. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation (1994)
13. Vose, M.D.: The Simple Genetic Algorithm. MIT Press, Cambridge (1999)
14. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, ETH Zurich, Switzerland (1999)
15. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), pp. 95–100. International Center for Numerical Methods in Engineering, CIMNE (2002)
16. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto evolutionary algorithm. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)

A Natural Evolution Strategy for Multi-objective Optimization

Tobias Glasmachers, Tom Schaul, and Jürgen Schmidhuber

IDSIA, University of Lugano, Switzerland
{tobias,tom,juergen}@idsia.ch

Abstract. The recently introduced family of *natural* evolution strategies (NES), a novel stochastic descent method employing the natural gradient, is providing a more principled alternative to the well-known covariance matrix adaptation evolution strategy (CMA-ES). Until now, NES could only be used for single-objective optimization. This paper extends the approach to the multi-objective case, by first deriving a $(1+1)$ hillclimber version of NES which is then used as the core component of a multi-objective optimization algorithm. We empirically evaluate the approach on a battery of benchmark functions and find it to be competitive with the state-of-the-art.

1 Introduction

The last decade has seen a shift in research focus from single-objective to multi-objective optimization (MOO) [13,7,2,12,1,5]. While many problems can very naturally be viewed as multi-objective (e.g., minimizing cost while simultaneously maximizing utility), they have traditionally been traded off into a single objective to be optimized. Numerous arguments have been put forward in favor of handling the multiple objectives explicitly, especially in the context of evolutionary computation. For one, the diversity of solutions found is larger than for single-objective optimization with fixed trade-offs, which in turn can improve over the single-objective optimization performance at its own game, as it may allow the search to circumnavigate local optima [7]. Furthermore, in many practical applications it is more advantageous to choose among the non-dominated solutions within the Pareto-front, rather than deciding upon a trade-off a priori and then maximizing it. Among the broad range of MOO algorithms that have been proposed (see e.g. [1] for an overview, omitted here for space reasons), approaches based on evolution strategies [5] are of particular interest for the present paper. They show how algorithms like the covariance matrix adaptation evolution strategy (CMA-ES [4,6]), that shine on non-seperable optimization problems, can be utilized for MOO.

The recently introduced family of *natural evolution strategies* (NES [11,10,9,3,8]), consists in an optimization method that follows a sampled natural gradient of the expected fitness, and as such, provides a more principled alternative to CMA-ES. In this paper we combine the well-founded framework of NES with the proven approach of tackling MOO using evolution strategies. This both significantly broadens the applicability of NES, and establishes a novel, elegant MOO algorithm. Our contribution is two-fold: First, we turn the most recent NES algorithm into a hillclimber. Second, we use this hillclimber as a module of an evolutionary algorithm for multi-objective optimization, following an established scheme. We benchmark both algorithms against their CMA-ES counterparts and obtain competitive results.

2 Natural Evolution Strategies

Natural evolution strategies (NES) [11, 10, 9, 3, 8] are a class of evolutionary algorithms for real-valued optimization. They maintain a Gaussian search distribution with fully adaptive covariance matrix. The principal idea is to adapt the search distribution to the problem at hand by following the natural gradient of expected fitness. Although relying exclusively on function value evaluations, the resulting optimization behavior closely resembles second order optimization techniques. This avoids drawbacks of regular gradients which are prone to slow or even premature convergence. Just like CMA-ES [4, 6], NES algorithms are invariant under monotone transformations of the fitness function and linear transformations of the search space (given that the initial search distribution is transformed accordingly).

In this paper we build upon the most recent NES variant, *exponential NES* (xNES), first presented in [3]. We start with stating its working principles, which are needed later on to cleanly derive its hillclimber variant.

In each generation the algorithm samples a population of $n \in \mathbb{N}$ individuals $x_i \sim \mathcal{N}(\mu, C)$, $i \in \{1, \dots, n\}$, i.i.d. from its search distribution, which is represented by the center $\mu \in \mathbb{R}^d$ and a factor $A \in \mathbb{R}^{d \times d}$ of the covariance matrix $C = AA^T$. These points are obtained by sampling $z_i \sim \mathcal{N}(0, I)$ and setting $x_i = \mu + A \cdot z_i$. In this paper, I always denotes the d-dimensional unit matrix. Let $p(x \,|\, \mu, A)$ denote the density of the search distribution $\mathcal{N}(\mu, AA^T)$. Then,

$$J(\mu, A) = \mathbb{E}[f(x) \,|\, \mu, A] = \int f(x) \; p(x \,|\, \theta) \; dx$$

is the expected fitness under the current search distribution. The so-called 'log-likelihood trick' enables us to write

$$\begin{aligned} \nabla_{(\mu,A)} J(\mu, A) &= \int \Big[f(x) \; \nabla_{(\mu,A)} \log(p(x \,|\, \mu, A)) \Big] \; p(x \,|\, \mu, A) \; dx \\ &\approx \frac{1}{n} \sum_{i=1}^{n} f(x_i) \; \nabla_{(\mu,A)} \log(p(x \,|\, \mu, A)) \ . \end{aligned}$$

Using raw fitness values endangers the algorithm to get stuck on plateaus and to systematically overjump steep optima. Thus, *fitness shaping* [11] is used to normalize the fitness values by shaping them into rank-based utility values $u_i \in \mathbb{R}$, $i \in \{1, \dots, n\}$. For this purpose we order the individuals by fitness, with

Algorithm 1. The xNES Algorithm

```
**Input**: $d \in \mathbb{N}$, $f : \mathbb{R}^d \to \mathbb{R}$, $\mu \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times d}$
$\sigma \leftarrow \sqrt[d]{|\det(A)|}$; $B \leftarrow A/\sigma$
**while** *stopping condition not met* **do**
    **for** $i \in \{1, \dots, n\}$ **do** $z_i \leftarrow \mathcal{N}(0, I)$; $x_i \leftarrow \mu + \sigma B \cdot z_i$
    sort $\{(z_i, x_i)\}$ with respect to $f(x_i)$
    $G_\mu \leftarrow \sum_{i=1}^n u_i \cdot z_i$
    $G_A \leftarrow \sum_{i=1}^n u_i \cdot (z_i z_i^T - I)$; $G_\sigma \leftarrow \mathrm{tr}(G_A)/d$; $G_B \leftarrow G_A - G_\sigma \cdot I$
    $\mu \leftarrow \mu + \eta_\mu \cdot \sigma B \cdot G_\mu$; $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma \cdot G_\sigma)$; $B \leftarrow B \cdot \exp(\eta_B \cdot G_B)$
**end**
```

$x_{1:n}$ denoting the best and $x_{n:n}$ denoting the worst offspring. We then use the "fitness-shaped" gradient $G = \sum_{i=1}^n u_i \cdot \nabla_{(\mu,A)} \log(p(x_{i:n} \mid \mu, A))$ to update the parameters of the search distribution. Typically, the utility values are either non-negative numbers that add to one, or a shifted variant with zero mean.

The xNES algorithm introduces a number of novel techniques for its updates. In each step, the coordinate system is transformed such that the search distribution has zero mean and unit variance. This results in the Fisher information matrix being the unit matrix and the natural gradient coinciding with the 'standard' gradient. The exponential map $M \mapsto \exp(M) = \sum_{n=0}^\infty \frac{1}{n!} M^n$ for symmetric matrices is used to encode the covariance matrix, resulting in a multiplicative form of the covariance matrix update (see [3] for details).

The parameters (μ, A) of the distribution can be split canonically into invariant components. This amounts to a (non-redundant) representation similar to CMA-ES, that is, we split off a global step size variable from the covariance matrix in the form $A = \sigma \cdot B$, with $\det(B) = 1$. We obtain the corresponding gradient components

$$G_\mu = \sum_{i=1}^n u_i \cdot z_i \qquad G_A = \sum_{i=1}^n u_i \cdot (z_i z_i^T - I)$$

with sub-components $G_\sigma = \mathrm{tr}(G_A)/d$ and $G_B = G_A - G_\sigma \cdot I$ (refer to [3] for the full derivation).

Let η_μ, η_σ, and η_B denote learning rates for the different parameter components. Putting everything together, the resulting xNES update rules for the search distribution read

$$\mu \leftarrow \mu + \eta_\mu \cdot \sigma B \cdot G_\mu \qquad \sigma \leftarrow \sigma \cdot \exp(\eta_\sigma \cdot G_\sigma) \qquad B \leftarrow B \cdot \exp(\eta_B \cdot G_B) \ .$$

The full xNES algorithm is summarized in Algorithm 1.

As indicated earlier, xNES is closely related to CMA-ES. However, conceptually xNES constitutes a much more principled approach to covariance matrix adaptation. This is because the updates of all parts of the search distribution, center, global step size, and full covariance matrix, result from the same principle of natural gradient descent.

Algorithm 2. (1 + 1)-xNES

```
Input: $d \in \mathbb{N}$, $f : \mathbb{R}^d \to \mathbb{R}$, $\mu \in \mathbb{R}^d$,
       $A \in \mathbb{R}^{d \times d}$
$\sigma \leftarrow 1$
while stopping condition not met
do
    $z \leftarrow \mathcal{N}(0, I)$
    $x \leftarrow \mu + \sigma A \cdot z$
    if $f(x)$ is better than $f(\mu)$ then
        $G_\mu \leftarrow z$
        $G_A \leftarrow zz^T - I$
        $\mu \leftarrow \mu + 1 \cdot A \cdot G_\mu$
        $A \leftarrow A \cdot \exp(\eta_A \cdot G_A)$
        $\sigma \leftarrow \sigma \cdot \exp(\eta_\sigma^+)$
    else $\sigma \leftarrow \sigma / \exp(\eta_\sigma^-)$
end
```

Algorithm 3. (1 + 1)-xNES with natural gradient descent

```
Input: $d \in \mathbb{N}$, $f : \mathbb{R}^d \to \mathbb{R}$, $\mu \in \mathbb{R}^d$,
       $A \in \mathbb{R}^{d \times d}$
while stopping condition not met
do
    $z \leftarrow \mathcal{N}(0, I)$; $x \leftarrow \mu + A \cdot z$
    if $f(x)$ is better than $f(\mu)$ then
      succ $\leftarrow +$; $z_{1:2} \leftarrow z$; $z_{2:2} \leftarrow 0$
    else
      succ $\leftarrow -$; $z_{1:2} \leftarrow 0$; $z_{2:2} \leftarrow z$
    $G_\mu \leftarrow \sum_{i=1}^{2} u_i^{(\mu)} \cdot z_{i:2}$
    $G_A \leftarrow \sum_{i=1}^{2} u_i^{(A,\text{succ})} (z_{i:2} z_{i:2}^T - I)$
    $\mu \leftarrow \mu + 1 \cdot A \cdot G_\mu$
    $A \leftarrow A \cdot \exp(\eta_A \cdot G_A)$
end
```

3 An Elitist Variant for the NES Family

In this section we introduce (1 + 1)-xNES, a hillclimber variant of xNES. Our goal is to use this algorithm as a building block for a multi-objective optimization scheme, in analogy to the development of the (1 + 1)-CMA-ES. The main motivation for this work is that (1 + 1)-xNES is conceptually simpler and more unified than (1 + 1)-CMA-ES. We apply a number of techniques to xNES that were used to derive (1 + 1)-CMA-ES from its population-based variant. In a second step we show that the resulting algorithm can be derived from the NES principle of following the natural fitness gradient.

The resulting algorithm implements the following principles to adapt its search strategy (as usual, an offspring is considered successful if its fitness is better than the fitness of the parent):

1. A successful offspring becomes the center of the search distribution.
2. Sampling a successful offspring results in a covariance matrix update.
3. Global step size adaptation is used to sustain a success rate of about 1/5.

The elitist (1+1)-xNES algorithm, stated in Algorithm 2, incorporates the above principles into the xNES algorithm in a straightforward way. It is designed such that its state is completely determined by its current search distribution. We use a global step size σ and a factor A to represent the covariance matrix $C = \sigma^2 \cdot A^T A$. We stick to this redundant representation for the sake of clarity, as it allows us to separate the mechanisms of xNES-style covariance matrix adaptation and success rule-based step size adaptation. Also note that the learning rate for the center has been fixed to one in order to satisfy the elitism rule. We set the other learning rates to $\eta_A = 1/4 \cdot d^{1.5}$, $\eta_\sigma^- = 1/5 \cdot d^{1.5}$, and $\eta_\sigma^+ = d^{1.5}$. The form of the dependency of the learning rates on the problem dimension is inspired by the learning rates of xNES, divided by its population size.

In the special case of $(1+1)$-xNES the matrix exponential in the covariance matrix update can be computed analytically (that is, without resorting to iterative matrix decomposition techniques) and in quadratic time. This is a consequence of the special form $M = v \cdot zz^T + w \cdot I$ of the argument: We exploit its eigen-decomposition, which consists of a one-dimensional eigenspace along z with eigenvalue $v \cdot \|z\|^2 + w$, while the space orthogonal to z forms an eigenspace with eigenvalue w. With the definitions $S = (1/\|z\|^2) \cdot zz^T$ and $R = I - S$ we obtain $\exp(M) = \exp(v) \cdot R + \exp(v \cdot \|z\|^2 + w) \cdot S$, which can be computed in $\mathcal{O}(d^2)$ operations. The decisive advantage of this computation is numerical stability, even if the time per generation remains cubic (matrix multiplication).

From a conceptual point of view Algorithm 2 is still unsatisfactory, because the step size adaptation mechanism is not derived from the NES principle of updating the search distribution by following the natural gradient of expected fitness. Fortunately, the NES update scheme (and fitness shaping in particular) is flexible enough to cover the elitist case, including the success-based update rule, as we will see in the following.

In the $(1+1)$-selection scheme we need to assign a rank-based utility value not only to the offspring, but also to the parent, giving us an additional degree of freedom. Using different utility values for different parts of the $(1+1)$-xNES update (analogous to using different learning rates) allows us to derive the full update rule from the principle of natural gradient descent. Note that multiplicative factors in the learning rate and the utility values are exchangable, because they have the exact same effect.

The simplest case is the update of the center μ, which is completely determined by the elitism rule. This leaves us with a single choice, amounting to $(u_1^{(\mu)}, u_2^{(\mu)}) = (1, 0)$ for the utility values. Note that in this notation the utility value u_1 automatically refers to the better individual, such that it may correspond to either parent or offspring. This reflects the intuitive notion that in the $(1+1)$-selection scheme only the better individual is of use (has positive utility), while the worse individual is discarded (has zero utility).

The covariance matrix update is a bit more involved, as here the utility values are success dependent. In the simpler case where the offspring is not successful (the **else** part in Algorithm 2), it does not have an impact on the update, and the corresponding utility value is $u_2^{(A,-)} = 0$. Interestingly, we can use the utility of the parent to encode the shrinking of the global step size. The calculation

$$-\eta_\sigma^- \cdot I = \eta_A \cdot G_A = \eta_A \cdot \left(u_1^{(A,-)} \cdot (00^T - I) + u_2^{(A,-)} \cdot (zz^T - I) \right)$$

shows that the choice $u_1^{(A,-)} = \eta_\sigma^- / \eta_A$ does the job (with $0 \in \mathbb{R}^d$ denoting the zero vector). In case of the offspring being successful the analog calculation

$$\eta_\sigma^+ \cdot I + \eta_A \cdot (zz^T - I) = \eta_A \cdot G_A = \eta_A \cdot \left(u_1^{(A,+)} \cdot (zz^T - I) + u_2^{(A,+)} \cdot (-I) \right)$$

results in $u_1^{(A,+)} = 1$ and $u_2^{(A,-)} = -\eta_\sigma^+ / \eta_A$. These rules amount to a natural adaptation of the notion of utility to the $(1+1)$ elitist selection scheme. This means that Algorithm 2 is fully compatible with the principle of strategy adaptation by following the (utility shaped) gradient of expected fitness. It can be turned into the equivalent Algorithm 3.

4 Experimental Evaluation of $(1 + 1)$-xNES

We compared the $(1 + 1)$-xNES hillclimber to both xNES and $(1 + 1)$-CMA-ES a number of standard benchmark functions. We initialized the algorithms by drawing the initial center of the search distribution from a Gaussian with zero mean and unit variance, and setting the covariance matrix to I. Each algorithm was run until it reached the target fitness of 10^{-10} (-10^3 for the unbounded functions ParabR and SharpR), in which case the trial is counted as a success. A trial is said to fail if it reaches the maximum number of 10^7 iterations or shrinks the search distribution below numerical limits, which amounts to premature convergence. The results are shown in Figure 1.

The plots in Figure 1 show $(1+1)$-xNES practically reaching the performance of $(1+1)$-CMA-ES on some benchmarks, while it falling behind on others, particularly in high dimensions. We attribute this to a better tuning of the parameters of $(1 + 1)$-CMA-ES, and the lack of evolution paths in xNES. Not surprisingly, the new elitist algorithm improves on the original xNES on nearly all unimodal problems studied here. Interestingly, both elitist algorithms have severe problems with the sharp ridge benchmark, on which they prematurely converge due to their too greedy strategy adaptation.

5 Multi-objective NES

We now posess all the ingredients to construct a natural evolution strategy for multi-objective optimization, named MO-NES. Our construction follows the successful scheme developed in [5].

The MO-NES algorithm maintains a population of $(1 + 1)$-xNES hillclimbers, with the goal of maximally approximating the Pareto front. Each generation, the $N \in \mathbb{N}$ hillclimbers generate one offspring each. As in MOO there is no unique notion of success due to multiple contradicting objectives, the selection scheme of the individual hillclimbers becomes meaningless. Instead, we adopt the indicator-based selection scheme used in [5] which consists of two stages. Parents and offspring are merged into a single population and ranked according to (1) the dominance relation, and (2) an indicator (see, e.g, [12]) that permits aggregating the relative value of each individual within its front into a single number (in contrast to the m-dimensional fitness vector).

For this purpose, the population is split into fronts $F_1, \dots, F_k$ using non-dominated sorting. Within each front no individual weakly dominates another, while F_i weakly dominates F_j for $i < j$. Thus, in this notation the set F_1 consists of the non-dominated solutions. A secondary sorting criterion is needed to rank individuals within each front. In this study we use the S-measure or hypervolume contribution [13], which is the Laplace measure of the volume of

Algorithm 4. The MO-NES Algorithm

```
Input: $f : \mathbb{R}^d \to \mathbb{R}^m$, $(\mu_i, \sigma_i, A_i) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{d \times d}$ for $i \in \{1, \dots, N\}$
while stopping condition not met do
    for $i \in \{1, \dots, N\}$ do
        $z_i \sim \mathcal{N}(0, I);\ \mu'_i \leftarrow \mu_i + \sigma_i A_i z_i;\ \sigma'_i \leftarrow \sigma_i;\ A'_i \leftarrow A_i$
    use non-dominated sorting to compute fronts $F_1, \dots, F_k$
    compute the $S$-measure of each individual within its front
    compute ranks $R_1, \dots, R_N, R'_1, \dots, R'_N \in \{1, \dots, 2N\}$
    for $i \in \{1, \dots, N\}$ do
        if $R'_i < R_i$ then
            $\sigma_i \leftarrow \sigma_i \cdot \exp(\eta_\sigma^+);\ \sigma'_i \leftarrow \sigma'_i \cdot \exp(\eta_\sigma^+);\ A'_i \leftarrow A'_i \cdot \exp(\eta_A \cdot [z_i z_i^T - I])$
        else
            $\sigma_i \leftarrow \sigma_i / \exp(\eta_\sigma^-);\ \sigma'_i \leftarrow \sigma'_i / \exp(\eta_\sigma^-)$
    copy best ranked $N$ individuals into $(\mu_i, \sigma_i, A_i)$ for $i \in \{1, \dots, N\}$
end
```

the set dominated by some point in objective space, but not by any other point in the front. The hypervolume depends on a reference point, which is chosen adaptively such that it is dominated by the whole population, and such that the best individuals w.r.t. a single objective are always preferred (which amounts to elitism w.r.t. each single objective). Then selection amounts to keeping the best N out of $2N$ individuals according to this ranking.

Care has to be taken when adapting the $(1+1)$-xNES hillclimber to this selection scheme, because the notion of success differs from the condition for survival. We say that an offspring that is ranked higher than its parent is successful, resulting in a covariance matrix update. In contrast, depending on the success of the mutation, the step size is updated *for parent and offspring.*

The resulting MO-NES algorithm is summarized in Algorithm 4. It is derived straightforwardly by removing the $(1 + 1)$-CMA-ES module from MO-CMA-ES and replicing it with $(1 + 1)$-xNES. An individual is represented by the triplet $(\mu, \sigma, A) \in \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^{d \times d}$, which is the state of the corresponding hillclimber. We denote parents by (μ_i, σ_i, A_i) and offspring by $(\mu'_i, \sigma'_i, A'_i)$ for $i \in \{1, \dots, N\}$.

6 Experimental Evaluation of MO-NES

We assess the performance of MO-NES compared to MO-CMA-ES on a collection of test problems found in [5], namely the standard benchmarks FON, ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, coming with rectangular feasible regions, and the unbounded problems ELLI$_1$, ELLI$_2$, CIGTAB$_1$, and CIGTAB$_2$. These benchmarks cover a number of typical challenges such as concave and disconnected pareto fronts, as well as highly correlated variables.

Experimental Setup. We largely follow the experimental procedure of [5]. The population size was set to $N = 100$, with individuals initialized uniformly at random in the feasible region. For the unbounded benchmarks the population

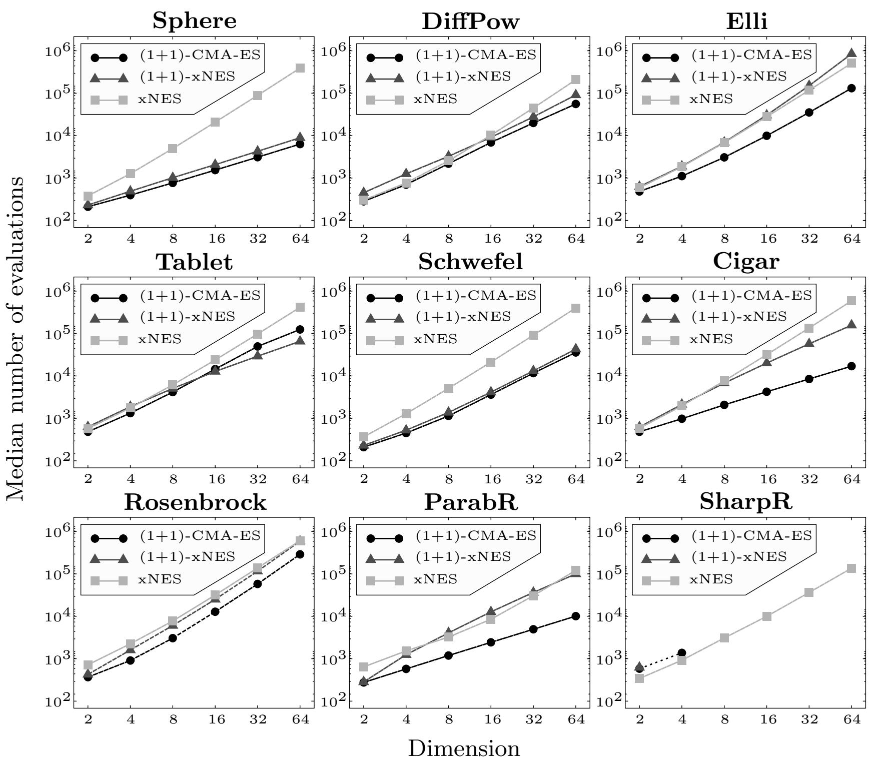


Fig. 1. Log-log plot of fitness evaluations required to reach target fitness (see text) over search space dimension for 9 different benchmark functions. The level of opacity of dashed connections indicates the fraction of successful runs. Setups for which no single run converged are not shown at all.

Table 1. Hypervolume covered by the populations of MO-NES and MO-CMA-ES after 50,000 fitness evaluations. Statistically superior results are marked bold.

	MO-NES			MO-CMA-ES		
benchmark function	25% quantile	50% quantile	75% quantile	25% quantile	50% quantile	75% quantile
FON	0.337443	0.337453	0.337479	**0.337496**	**0.337511**	**0.337539**
ZDT1	0.661945	0.661962	0.661972	0.661934	0.661958	0.661972
ZDT2	0.328698	0.328703	0.328713	0.328697	0.328707	0.328720
ZDT3	1.042180	1.042180	1.042190	1.042180	1.042180	1.042190
ZDT4	0.661836	0.661860	0.661885	0.661834	0.661857	0.661879
ZDT4	10.93040	12.80430	15.95730	9.53145	11.77960	12.46610
ZDT6	**0.3225640**	**0.3225750**	**0.3225810**	0.0863215	0.3225550	0.3225770
ELLI_1	95.5311	95.5527	95.5593	95.5466	95.5553	95.5604
ELLI_2	99.9872	99.9905	99.9922	99.9897	99.9907	99.9931
CIGTAB_1	97.2286	97.2302	97.2310	**97.2303**	**97.2312**	**97.2318**
CIGTAB_2	99.9981	99.9985	99.9987	**99.9984**	**99.9988**	**99.9990**

was sampled from $[-10, 10]^d$. We used a search space dimension of $d = 10$ for all problems except FON, where it is fixed to $d = 3$. Resembling [5], we set the component-wise standard deviation of the initial search distribution to 0.6 times the edge length of the hyper-rectangle from which the initial population is sampled. Constraints were handled by evaluating the closest feasible point and adding 10^{-6} times the squared norm of the distance to the feasible region to each fitness component.

Each algorithm was granted 50,000 fitness evaluations, and assigned a score according to the hypervolume dominated by the final population. To achieve comparability with other studies, we fix the reference point for the hypervolume computation to $(1, 1)^T$ for FON and the ZDT-benchmarks, with the exception of $(1, 20)^T$ for ZDT4 (which is not solved satisfactory by any of the two algorithms), and to $(10, 10)^T$ for the unconstrained ELLI and CIGTAB problems.[1] We performed 25 independent trials for each experiment.

Results and Discussion. The results are summarized in Table 1. There is no clear trend indicating that one algorithm would be generally preferable to the other. In most cases the differences between the algorithms are negligible, in the sense that they are below the range of the inter-trial deviations, and only in the fifth digit of the hypervolume. We found four significant differences (Wilcoxon rank sum test, $p = 0.01$): The MO-CMA-ES performs better on benchmarks with quadratic objectives, such as FON and CIGTAB, while MO-NES is superior on the ZDT6 problem. We conclude that the Pareto front approximations obtained by the two algorithms are generally of comparable quality. Taking the conceptual parallels of the two algorithms into account, this result does not come as a surprise. It shows that our novel MO-NES algorithm achieves state-of-the-art performance.

7 Conclusion

We presented two novel algorithms. The $(1 + 1)$-xNES hillclimber constituts a minimal elitist variant of the xNES algorithm which can be derived completely from the principle of natural gradient descent, despite its success-based step size adaptation rule. Like other NES algorithms, $(1+1)$-xNES is more principled than its canonical counterpart $(1 + 1)$-CMA-ES. Combining our new hillclimber with the multi-objective optimization scheme established for MO-CMA-ES results in the MO-NES algorithm. We empirically find both algorithms to exhibit state-of-the-art performance.

The impact of these contributions be seen from two perspectives: On the one hand, they make NES capable of multi-objective optimization, on the other

[1] Note that adaptively computed reference points are used in both algorithms to compute the S-measure for selection, and that we use these fixed reference points only for the evalution of the final fronts. This procedure is chosen to foster comparability with future studies. In particular, it does not expose any additional information to the search algorithms.

hand, they enrich the field of evolutionary MOO by the NES principle of descent along the natural fitness gradient.

Acknowledgments

This work was funded through the 7th framework programme of the European Union, under grant number 231576 ("STIFF" project), and SNF grant number 200021-113364/1.

References

1. Coello Coello, A.C., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary algorithms for solving multi-objective problems. Springer, New York (2007)
2. 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)
3. Glasmachers, T., Schaul, T., Sun, Y., Wierstra, D., Schmidhuber, J.: Exponential Natural Evolution Strategies. In: Genetic and Evolutionary Computation Conference, GECCO (2010)
4. Hansen, N., Ostermeier, A.: Completely Derandomized Self-Adaptation in Evolution Strategies. Evolutionary Computation 9(2), 159–195 (2001)
5. Igel, C., Hansen, N., Roth, S.: Covariance Matrix Adaptation for Multi-objective Optimization. Evolutionary Computation 15(1), 1–28 (2007)
6. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms–a comparative review. Natural Computing 3(1), 77–112 (2004)
7. Knowles, J., Watson, R., Corne, D.: EMO 2001. LNCS, vol. 1993. Springer, Heidelberg (2001)
8. Rückstieß, T., Sehnke, F., Schaul, T., Wierstra, D., Yi, S., Schmidhuber, J.: Exploring parameter space in reinforcement learning. Paladyn Journal of Behavioral Robotics 1(1), 14–24 (2010)
9. Sun, Y., Wierstra, D., Schaul, T., Schmidhuber, J.: Efficient Natural Evolution Strategies. In: Genetic and Evolutionary Computation Conference, GECCO (2009)
10. Sun, Y., Wierstra, D., Schaul, T., Schmidhuber, J.: Stochastic Search using the Natural Gradient. In: International Conference on Machine Learning, ICML (2009)
11. Wierstra, D., Schaul, T., Peters, J., Schmidhuber, J.: Natural Evolution Strategies. In: Proceedings of the Congress on Evolutionary Computation (CEC 2008, Hongkong. IEEE Press, Los Alamitos (2008)
12. Zitzler, E., Künzli, S.: Indicator-Based Selection in Multiobjective Search. 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. 832–842. Springer, Heidelberg (2004)
13. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)

Solving Multiobjective Optimization Problem by Constraint Optimization

He Jiang[1,2], Shuyan Zhang[1], and Zhilei Ren[3]

[1] School of Software, Dalian University of Technology
jianghe@dlut.edu.cn, shyzhang@mail.dlut.edu.cn
[2] Department of Computer Science, University of Vermont
[3] School of Mathematical Sciences, Dalian University of Technology
ren@mail.dlut.edu.cn

Abstract. Multiobjective optimization problems (MOPs) have attracted intensive efforts from AI community and many multiobjective evolutionary algorithms (MOEAs) were proposed to tackle MOPs. In addition, a few researchers exploited MOEAs to solve constraint optimization problems (COPs). In this paper, we investigate how to tackle a MOP by iteratively solving a series of COPs and propose the algorithm named multiobjective evolutionary algorithm based on constraint optimization (*MEACO*). In contrast to existing MOEAs, *MEACO* requires no complex selection mechanism or elitism strategy in solving MOPs. Given a MOP, *MEACO* firstly constructs a new COP by transforming all but one of objective functions into constraints. Then, the optimal solution of this COP is computed by a subroutine evolutionary algorithm so as to determine some Pareto-optimal solutions. After that, a new COP with dramatically reduced search space can be constructed using existing Pareto-optimal solutions. This new generated COP will be further solved to find more Pareto-optimal solutions. This process is repeated until the stopping criterion is met. Experimental results on 9 well-known MOP test problems show that our new algorithm outperforms existing MOEAs in terms of convergence and spacing metrics.

Keywords: Multiobjective Optimization, Constraint Optimization, Evo- lutionary Algorithm.

1 Introduction

Multiobjective optimization problems (MOPs) arising from real-world applications have attracted great efforts from AI community. Since those objective functions in MOPs may conflict with each other, researchers usually seek for a set of trade-off solutions (Pareto-optimal solution set) rather than a unique solution. Since evolutionary algorithms (EAs) are capable of handling a set of solutions for complex problems, many multiobjective evolutionary algorithms (MOEAs) have been proposed for MOPs in recent years. Those MOEAs fall into two categories [1]. Algorithms of the first category use Pareto-ranking based selection mechanisms and employ fitness sharing to retain diversity. Some representative algorithms include *VEGA* [2], *MOGA* [3], *NSGA* [4], *NPGA* [5]. Algorithms of the second category employ elitism strategy and some other mechanisms (e.g., the clustering procedure, the nearest neighbor density estimation technique, and the adaptive grid algorithm) to maintain diversity. Some representative algorithms include *SPEA*2 [6], *PAES* [7], *NSGA*-II [8], *PESA*-II [9], *RM-MEDA* [10], and *NNIA* [11].

Constraint optimization problems (COPs) aim to achieve the optimal solution of the objective function under certain constraints, including inequality and equality constraints. Due to the great success of MOEAs, many researchers have exploited MOEAs to solve COPs by transforming a fraction of those constraints in COPs into objective functions. Some representative algorithms include *IS-PAES* [12], *COMOGA* [13], and *HCOEA* [14].

In this paper, we investigate how to solve MOPs in a reverse way by transforming them into a series of COPs. Motivated by this idea, the multiobjective evolutionary algorithm based on constraint optimization (*MEACO*) is proposed. Given a MOP, *MEACO* retains only one objective function and transforms other ones into constraints. In this way, a new COP is constructed. Then, the optimal solution of this COP can be achieved to initialize the Pareto-optimal solution set. With existing Pareto-optimal solutions, another new COP with dramatically reduced search space can be constructed. This new COP can be solved to find more Pareto-optimal solutions. This process is iteratively repeated until no Pareto-optimal solution can be found. To evaluate the performance of *MEACO*, experiments are conducted on 9 widely used MOP test problems. Experiments indicate that *MEACO* can achieve better convergence and spacing metrics than *NSGA*-II, *SPEA*2, and *NNIA*.

2 Notations

Without loss of generality, we consider the minimization problem forms for both MOPs and COPs in this paper. A MOP can be defined as follows.

$$\min F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \tag{1}$$

s.t. $x = (x_1, x_2, \ldots, x_n) \in \Omega$, where x is the decision vector and Ω is the feasible region in decision space.

Let $x^1, x^2 \in \Omega$, x^1 is said to *d*ominate x^2 (denoted as $x^1 \succ x^2$) iff $\forall i \in \{1, 2, \ldots, m\}$, $f_i(x^1) \leq f_i(x^2)$ and $\exists i^* \in \{1, 2, \ldots, m\}$, $f_{i^*}(x^1) < f_{i^*}(x^2)$. A decision vector x^* is a *Pareto-optimal solution* in (1) iff there is no $x \in \Omega$ such that $x \succ x^*$. The *Pareto-optimal set* is defined as $P^* = \{x^* \in \Omega | \neg\exists x \in \Omega, x \succ x^*\}$. The *Pareto-optimal front* is the image of the Pareto-optimal set in the objective space, which is defined as $PF^* = \{F(x^*) = (f_1(x^*), f_2(x^*), \ldots, f_m(x^*))^T | x^* \in P^*\}$. Under those definitions, the goal is to achieve a set of Pareto-optimal solutions equidistantly approximating the true Pareto-optimal front.

In contrast to MOP, a COP can be defined as follows.

$$\min f(x) \tag{2}$$

s.t. $g_i(x) \leq 0$ $(i \in \{1, 2, \ldots, q\})$ and $h_j(x) = 0$ $(j \in \{1, 2, \ldots, r\})$, where $x = (x_1, x_2, \ldots, x_n) \in R^n$ is a n-dimensional decision vector and $f(x)$ is the objective function. If a decision vector satisfies all the inequality and equality constraints, we say it is a feasible solution. Otherwise, it is an infeasible solution.

There are many metrics for evaluating the performance of MOEAs in the literature. In this paper, we adopt two widely used metrics, i.e., convergence metric [15] and spacing metric [16].

Convergence metric: let $P^* = \{p^1, p^2, p^3, \ldots, p^{|P^*|}\}$ be the target set of points on the true Pareto-optimal front and $P = \{x^1, x^2, x^3, \ldots, x^{|P|}\}$ be the solution set by an algorithm. For every solution $x^i \in P$, the smallest normalized Euclidean distance to P^* is defined as

$$d_i = \min_{j=1}^{|P^*|} \sqrt{\sum_{k=1}^{m} \left(\frac{f_k\left(x^i\right) - f_k\left(p^j\right)}{f_k^{max} - f_k^{min}} \right)^2} \tag{3}$$

wheref_k^{max} and f_k^{min} are the maximum and minimum values of the k^{th} objective function in P^*, respectively. Then convergence metric is defined as the average value of those normalized distance for all solutions in P, i.e., $Con(P) = \sum_{i=1}^{|P|} d_i/|P|$. It indicates the extent to which P approximates the true Pareto-optimal front. The smaller $Con(P)$ is, the better an algorithm is.

Spacing metric: let $P = \{x^1, x^2, x^3, \ldots, x^{|P|}\}$ be the solution set by an algorithm. Let $\hat{d}_i = \min_{j=1}^{|P|} \sum_{k=1}^{m} |f_k(x^i) - f_k(x^j)|$ for every $x^i \in P$. Let $\bar{d}$ be the average value of $\hat{d}_i$. Spacing metric is defined as follows.

$$S\left(P\right) = \sqrt{\frac{1}{|P| - 1} \sum_{i=1}^{|P|} \left(\bar{d} - \hat{d}_i \right)^2} \tag{4}$$

Spacing metric presents the spread measure of the solution set obtained by an algorithm. The smaller this value is, the better distribution an algorithm provides. This value is zero when all the solutions in P are equidistantly spaced.

3 MEACO

Due to the paper length limit, we will explain our algorithm for bi-objective optimization problem in this paper.

3.1 An Example

Fig.1 presents an example for a bi-objective optimization problem. We transform the second objective function f_2 into a constraint. Given a value f^*, there may exists several solutions $x^1, x^2, x^3, \ldots, x^w$ such that $f_2(x^i) = f^*(i \in \{1, 2, \ldots, w\})$ and $f_1(x^1) \leq f_1(x^2) \leq f_1(x^3) \leq \ldots \leq f_1(x^w)$ (see Fig.1 (a)). For brevity, we only plot the pixel $(f^*, f_1(x^1))$ in Fig.1.

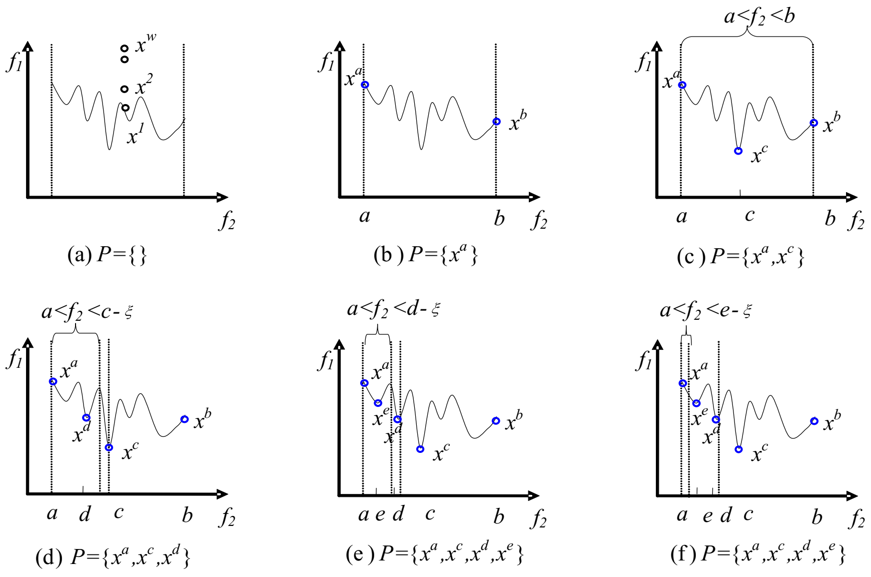


Fig. 1. Illustration of *MEACO*

For this example, our algorithm works as follows. Firstly, the minimum and maximum values (denoted as a and b, respectively) of f_2 are calculated (see Fig.1 (b)). Obviously, this computation for a can be done by solving a single-objective optimization problem (SOP) $\min f_2(x)$ s.t. $x \in \Omega$. Similarly, the computation for b is equivalent to solving a SOP $\max f_2(x)$ s.t. $x \in \Omega$. According to the definition of Pareto-optimal solution, the solution x^a must be Pareto-optimal and be added to the solution set P. A COP is then constructed as $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$. Let x^c be the optimal solution to this new COP, let $c = f_2(x^c)$ (see Fig.1 (c)). Obviously, x^b is dominated by x^c. It can be easily verified that x^c is Pareto-optimal, and no Pareto-optimal solution exists when $f_2(x) > c$. Therefore, the solution set P is updated by adding x^c to it. Then, we continue to construct a new COP as $\min f_1(x)$ s.t. $a < f_2(x) < c - \xi$, where $x \in \Omega$ and ξ is a predefined parameter aiming to keep the spacing of P. This latest constructed COP is solved to achieve the optimal solution x^d (see Fig.1 (d)). Similar to x^c, we can verify that x^d is Pareto-optimal and P is further updated. Since no other Pareto-optimal solution exists when $d < f_2(x) < c - \xi$, we further solve the COP $\min f_1(x)$ s.t. $a < f_2(x) < d - \xi$, where $x \in \Omega$(see Fig.1 (e)). This process is repeated until the stopping criterion is met. Usually, the stopping criterion is determined by spacing metric.

3.2 Framework of MEACO

Table 1 presents the framework of our *MEACO* algorithm. Firstly, the minimum value and maximum values of $f_2(x)$ are achieved by solving SOPs $\min f_2(x)$ or

$\mathbf{max} f_2(x)$ s.t. $x \in \Omega$ (see Step (1)). When multiple solutions $x^1, x^2, x^3, \ldots, x^w$ can be returned, the only solution x^i is retained such that $f_1(x^i)$ is minimal among $f_1(x^1), f_1(x^2), f_1(x^3), \ldots, f_1(x^w)$. By this way, we have two solutions x^a and x^b, where $a = f_2(x^a), b = f_2(x^b)$ are the minimum, maximum values for $f_2(x)$, respectively. Obviously, x^a is Pareto-optimal to be added to the Pareto-optimal solution set P (see Step (2)). It's still uncertain at this time whether x^b is Pareto-optimal or not. Then, a COP is constructed as $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$. After the optimal solution x^c is obtained (see Step (3)), we can decide whether x^b is Pareto-optimal or not. There're two cases as follows.

Case 1: x^b is Pareto-optimal (see Step (4.1)–(4.5))

After x^b is added to P (see Step (4.1)), we compare x^a and x^c to check whether x^c is dominated by x^a. If so, no Pareto-optimal solution exists under

Table 1. *MEACO* algorithm

```
Algorithm: MEACO
Input: $f_1, f_2$
Output: solution set $P$
Begin
(1) obtain the minimum value $a$, maximum value $b$ of $f_2$, let $x^a$, $x^b$ be the solution
    related to $a, b$, respectively
(2) let $P = \{x^a\}$ // $x^a$ must be Pareto-optimal
(3) obtain the optimal solution $x^c$ to $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$, let
    $c = f_2(x^c)$
(4) if $f_1(x^b) < f_1(x^c)$ and $f_1(x^b) < f_1(x^a)$ then //$x^b$ is Pareto-optimal
    (4.1) $P = P \cup \{x^b\}, \delta = (|f_1(x^a) - f_1(x^b)| + |b - a|)/200$
    (4.2) if $f_1(x^a) < f_1(x^c)$ then return $P$ //$x^c$ isn't Pareto-optimal
    (4.3) if $|f_1(x^a) - f_1(x^c)| + |c - a| > \delta$ and $|f_1(x^c) - f_1(x^b)| + |b - c| > \delta$ then
        (4.3.1) $P = P \cup \{x^c\}$
        (4.3.2) $P' = IntervalOpt(a, c - \xi, x^a, x^c, \delta, \xi)$
        (4.3.3) $P = P \cup P'$
        (4.3.4) return $P$
    (4.4) if $|f_1(x^a) - f_1(x^c)| + |c - a| \leq \delta$ then return $P$;
    (4.5) if $|f_1(x^c) - f_1(x^b)| + |b - c| \leq \delta$ then
        (4.5.1) $P' = IntervalOpt(a, c - \xi, x^a, x^b, \delta, \xi)$
        (4.5.2) $P = P \cup P'$
        (4.5.3) return $P$
    else // $x^b$ isn't Pareto-optimal
    (4.6) if $f_1(x^a) < f_1(x^c)$ then return $P$ //$x^c$ isn't Pareto-optimal
    (4.7) if $|f_1(x^a) - f_1(x^c)| + |c - a| \leq \delta$ then return $P$
        else
        (4.7.1) $P = P \cup \{x^c\}, \delta = (|f_1(x^a) - f_1(x^c)| + |c - a|)/200$
        (4.7.2) $P' = IntervalOpt(a, c - \xi, x^a, x^c, \delta, \xi)$
        (4.7.3) $P = P \cup P'$
        (4.7.4) return $P$
End
```

the constraint $a < f_2(x) < b$ and the current solution set P is then returned (see Step (4.2)). Otherwise, we need to further check if x^c satisfies the spacing requirement that the distance (in the objective space) between any two solutions in P must be greater than a threshold δ. If the spacing requirement is satisfied, x^c can be added to P and we continue to obtain Pareto-optimal solutions under the constraint $a < f_2(x) < c-\xi$ by calling a subroutine function *IntervalOpt* (see Step (4.3)), where ξ(ξ is set to be δ /10) is introduced to avoid endless loops when the Pareto-optimal front is smoothly continuous. If the spacing requirement is not met, there may be two possible reasons.

For the first reason that x^c is too close to x^a (see Step (4.4)), it's unnecessary to further investigate those solutions under either the constraint $c+\xi < f_2(x) < b$ or the constraint $a < f_2(x) < c - \xi$. On the one hand, every solution under the constraint $c + \xi < f_2(x) < b$ is dominated by x^c. On the other hand, every solution under the constraint $a < f_2(x) < c - \xi$ is either dominated by x^a or closer to x^a than x^c.

For the second reason that x^c is too close to x^b (see Step (4.5)), we need to further achieve those solutions under the constraint $a < f_2(x) < c - \xi$ by calling the subroutine function *IntervalOpt* (see Step (4.5.1)–(4.5.3)). It's now unnecessary to consider those solutions under the constraint $c + \xi < f_2(x) < b$, since all such solutions are dominated by x^c.

Case 2: x^b isn't Pareto-optimal (see Step (4.6)–(4.7))

In this case, x^b will be dominated by x^c or x^a. We compare x^a and x^c to check whether x^c is dominated by x^a. If so, no Pareto-optimal solution exists under the constraint $a < f_2(x) < b$ and P is then returned (see Step (4.6)). Otherwise, we continue to check whether x^c satisfies the spacing requirement or not (see Step (4.7)). The following work (Step (4.7.1)–(4.7.4)) can be explained in a similar way as Step (4.3.1)–(4.3.4).

3.3 IntervalOpt

In *MEACO*, a subroutine named *IntervalOpt* is called to achieve the Pareto-optimal solution set under certain constraints. As shown in Tab.2, *IntervalOpt* takes in several parameters, including a, b, x^1, and x^2, where a and b impose the constraint $a < f_2(x) < b$, x^1 and x^2 provide the spacing requirements. *IntervalOpt* consists of 5 steps.

In Step (1), we achieve the optimal solution x^c to the COP $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$.

Then we check whether x^c is dominated by x^1 in Step (2). If so, it can be verified that no other Pareto-optimal solution exists under the constraint $a < f_2(x) < b$. Therefore, an empty solution set will be returned. It should be noted that we needn't to compare x^c with x^2, since $f_2(x^c) < b \leq f_2(x^2)$ implies that x^c will never be dominated by x^2.

When x^c is Pareto-optimal, *InterverOpt* continues to check if x^c is too close to x^1 in Step (3). If so, no other Pareto-optimal solution can be found under the constraint $a < f_2(x) < b$, due to the same reason as Step (4.4) in *MEACO*. Otherwise, *IntervalOpt* continues to check whether it's too close to x^2 (see Step

Table 2. *IntervalOpt* algorithm

```
Algorithm: IntervalOpt
Input: $a, b, x^1, x^2, \delta, \xi$
Output: solution set $P$
Begin
(1) obtain the optimal solution $x^c$ to $\min f_1(x)$ s.t. $a < f_2(x) < b$, where $x \in \Omega$, let
    $c = f_2(x^c)$
(2) if $x^c$ is dominated by $x^1$ then return $\phi$
(3) if $|f_1(x^c) - f_1(x^1)| + |c - f_2(x^1)| \leq \delta$ then return $\phi$
(4) if $|f_1(x^c) - f_1(x^2)| + |c - f_2(x^2)| \leq \delta$ then
    (4.1) $P = IntervalOpt(a, c - \xi, x^1, x^2, \delta, \xi)$
    (4.2) return $P$
(5) if $|f_1(x^c) - f_1(x^1)| + |c - f_2(x^1)| > \delta$ and $|f_1(x^c) - f_1(x^2)| + |c - f_2(x^2)| > \delta$ then
    (5.1) $P = \{x^c\}$
    (5.2) $P' = IntervalOpt(a, c - \xi, x^1, x^c, \delta, \xi)$
    (5.3) $P = P \cup P'$
    (5.4) return $P$
End
```

(4)). If so, the Pareto-optimal solutions will be further achieved by recursively calling *IntervalOpt* with more restrictive constraints (see Step (4.1)–(4.2)).

Finally, when x^c satisfies all spacing requirements, x^c will be added to P and *IntervalOpt* recursively obtains new Pareto-optimal solutions with more restrictive constraints (see Step (5)).

3.4 Solving COPs with EAs

As shown in both *MEACO* and *IntervalOpt*, we need to solve some COPs transformed from the original MOPs. (1) In Step (1) of *MEACO*, we need to compute the minimum and maximum values of f_2. (2) In Step (3) of *MEACO*, we need to compute the optimal solution x^c to $\min f_1(x)$, s.t. $a < f_2(x) < b$. (3) In Step (1) of *IntervalOpt*, we also need to compute the the optimal solution x^c to $\min f_1(x)$, s.t. $a < f_2(x) < b$.

Although any existing algorithm for COPs can be incorporated into *MEACO*, we employ EAs to tackle those COPs in this paper. In contrast to other algorithms, EAs are more widespread with no restriction on COPs. Due to the paper length limit, all the details of those EAs are presented in our technical report http://www.cems.uvm.edu/∼hejiang/MEACO-tech.pdf.

4 Experimental Results

To evaluate our algorithm, experiments are conducted on 9 well-known bi-objective optimization problems, including SCH, FON, POL, KUR, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. In the experiment, *MEACO* is coded in Microsoft

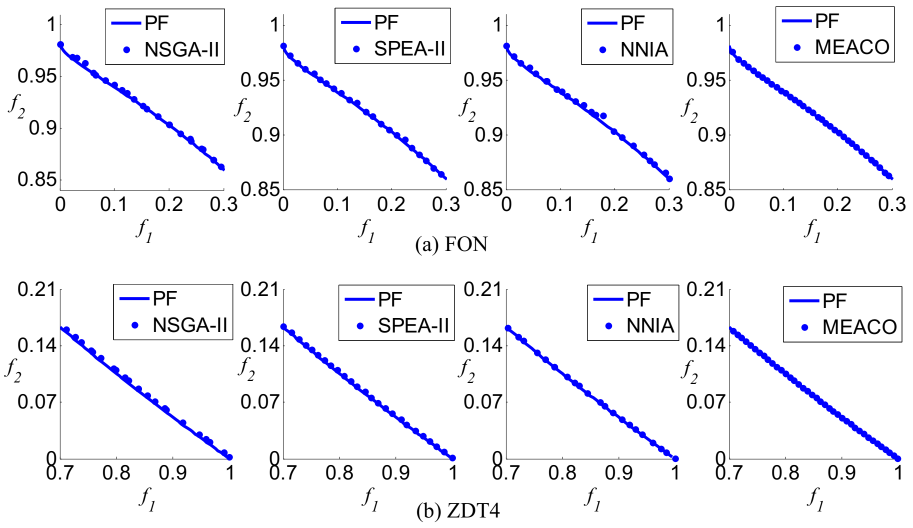


Fig. 2. Pareto-optimal front of algorithms

VC++ 6.0 and run on a Pentium Dual Core 2.8G with 4G memory running Win XP. For comparison, we also run *NSGA*-II, *SPEA*2, and *NNIA* on the same test problems. Those source codes of *NSGA*-II and *SPEA*2 are downloaded from (http://www.lania.mx/∼ccoello/EMOO/). The source code of *NNIA* is obtained from (http://see.xidian.edu.cn/iiip/mggong/Projects/NNIA.htm). All the parameters for *NSGA*-II, *SPEA*2, and *NNIA* are also from [11].

In this conference paper, we only plot part of the Pareto-optimal front of algorithms on FON (see Fig.2(a)) and ZDT4 (see Fig.2(b)). Similar results can be concluded for other MOPs. It can be observed that all the algorithms can well approximate the true Pareto-optimal fronts on both FON and ZDT4. Out of all 4 algorithms, our new algorithm achieves solutions which are the most equidistantly distributed on the true Pareto-optimal fronts on both FON and ZDT4.

Table 3 present the numerical results of convergence, spacing and time metrics on all MOPs. For every MOP, the averaged values over 30 runs are given and the best ones are marked in bold font. It can be observed from Table 3 that *MEACO* outperforms other algorithms on all MOPs except POL and ZDT6 ni terms of convergence metric, while SPEA2 achieves the best convergence metric on POL and *NSGA*-II achieves the best on ZDT6. Table 3 illustrates that *MEACO* could achieve better spacing metrics on 6 MOPs out of all MOPs.From Table 3, we can observe that NSGA-II uses least time for all test problems among those 4 algorithms.The running of our algorithm is similar to that of *SPEA*2.

Table 3. Convergence spacing and time results of algorithms on MOPs

	NSGA			*SPEA2*			*NNIA*			*MEACO*		
	Conv.	Spac.	Time	Conv.	Spac.	Time	Conv.	Spac.	Time	Conv.	Spac.	Time
SCH	0.0020	0.0193	**0.0373**	0.0020	**0.0147**	0.1564	0.0021	0.0272	0.0646	**0.0019**	0.0209	0.0380
FON	0.0034	0.0068	**0.0414**	0.0025	0.0029	0.1826	0.0031	0.0060	0.1033	**0.0020**	**0.0019**	0.1568
POL	0.0017	0.1002	**0.0418**	**0.0015**	**0.0394**	0.1715	0.0016	0.0916	0.0929	0.0022	0.0719	0.2017
KUR	0.0022	0.1016	**0.0416**	**0.0018**	0.0823	0.1744	0.0020	0.0878	0.0978	**0.0018**	**0.0814**	0.1347
ZDT1	0.0024	0.0068	**0.0401**	0.0022	0.0034	0.1240	0.0020	0.0072	0.0648	**0.0019**	**0.0018**	0.1475
ZDT2	0.0020	0.0072	**0.0382**	0.0019	0.0032	0.1445	**0.0018**	0.0073	0.0648	**0.0018**	**0.0016**	0.1600
ZDT3	0.0019	0.0077	**0.0360**	0.0019	**0.0039**	0.1589	0.0018	0.0079	0.0634	**0.0016**	0.0101	0.0878
ZDT4	0.0031	0.0075	**0.0361**	0.0032	0.0028	0.2729	0.0027	0.0069	0.0643	**0.0020**	**0.0021**	0.5304
ZDT6	**0.0019**	0.0075	**0.0450**	0.0020	0.0023	0.0805	0.0020	0.0056	0.0686	0.0030	**0.0013**	0.1015

5 Conclusion and Future Work

In this paper, a new algorithm *MEACO* is proposed to solve MOPs by transforming them into a series of COPs. Several EAs are also presented to tackle those transformed COPs. Experimental results indicate that our algorithm is a promising way to tackle MOPs. In contrast to solving COPs with MOEAs, work of this paper throws a light on how to solve MOPs by methods for COPs. In future work, we will extend *MEACO* for MOPs with more objective functions. It's also interesting to incorporate some other efficient algorithms (for COPs) into *MEACO*.

Acknowledgement

Our work is partially supported by the Natural Science Foundation of China under Grant No. 60805024, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20070141020.

References

1. Coello Coello, C.A.: Evolutionary Multi-Objective Optimization: A Historical View of the Field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006)
2. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proc. of the 1st Int'l Conf. on Genetic Algorithms, pp. 93–100. L. Erlbaum Associates, Inc., Hillsdale (1985)
3. Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Proc. of the 5th Int'l Conf. on Genetic Algorithms, pp. 416–423. Morgan Kauffman, San Mateo (1993)
4. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
5. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proc. of the 1st IEEE Conference on Evolutionary Computation, pp. 82–87. IEEE, Piscataway (1994)

6. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, pp. 95–100. Springer, Berlin (2001)
7. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6(2), 182–197 (2002)
9. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization. In: Proc. of the Genetic and Evolutionary Computation Conf., pp. 283–290. Morgan Kaufmann Publishers, San Francisco (2001)
10. Zhang, Q.F., Zhou, A.M., Jin, Y.: RM-MEDA: A Regularity Model-Based Multiobjective Estimation of Distribution Algorithm. IEEE Trans. on Evolutionary Computation 11(1), 41–63 (2007)
11. Gong, M.G., Jiao, L.C., Du, H.F., Bo, L.F.: Multiobjective Immune Algorithm with Nondominated Neighbor-Based Selection. Evolutionary Computation 16(2), 225–255 (2008)
12. Aguirre, A.H., Rionda, S.B., Coello Coello, C.A., Lizrraga, G.L., Montes, E.M.: Handling Constraints Using Multiobjective Optimization Concepts. Int'l Journal for Numerical Methods in Engineering 59(15), 1989–2017 (2004)
13. Surry, P.D., Radcliffe, N.J.: The COMOGA Method: Constrained Optimization by Multi-Objective Genetic Algorithms. Control and Cybernetics 26(3), 391–412 (1997)
14. Wang, Y., Cai, Z.X., Guo, G.Q., Zhou, Y.R.: Multiobjective Optimization and Hybrid Evolutionary Algorithm to Solve Constrained Optimization Problems. IEEE Trans. on System, Man And Cybernetics 37(3), 560–575 (2007)
15. Deb, K., Jain, S.: Running Performance Metrics for Evolutionary Multi-Objective Optimization, Technical Report. Kanpur: Indian Institute of Technology Kanpur. NO. 2002004 (2002)
16. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA (1995)

Enhancing Diversity for Average Ranking Method in Evolutionary Many-Objective Optimization

Miqing Li, Jinhua Zheng, Ke Li, Qizhao Yuan, and Ruimin Shen

Institute of Information Engineering, Xiangtan University,
411105 Hunan, China
limit1008@126.com, jhzheng@xtu.edu.cn, JerryI00@yahoo.com.cn,
yuanyuan_xtu@163.com, srmxp@126.com

Abstract. The average ranking (AR) method has been shown highly effective to provide sufficient selection pressure searching towards Pareto optimal set in many-objective optimization. However, as lack of diversity maintenance mechanism, the obtained final set may only concentrate in a subregion of Pareto front. In this paper, we propose a diversity maintenance strategy for AR to balance convergence and diversity during evolution process. We employ grid to define an adaptive neighborhood for each individual, whose size varies with the number of objectives. Moreover, a layering selection scheme integrates it and AR to pick out well-converged individuals and prohibit or postpone the archive of adjacent individuals. From an extensive comparative study with original AR and two other diversity maintenance methods, the proposed method shows a good balance among convergence, uniformity and spread.

Keywords: Multiobjective optimization, Many-objective optimization, Average ranking, Diversity maintenance.

1 Introduction

During the recent past, evolutionary multiobjective optimization (EMO) algorithms have been receiving an extensive interest, mainly because of their potential to find a well-distributed approximation of Pareto optimal set. Nonetheless, most of them merely focus on the problems with two or three objectives, in spite of the fact that the problems with more than three objectives widely exists in real-world application [1], which is generally termed many-objective problems. One of the main reasons for this occurrence is that the proportion of nondominated solutions in a population rises rapidly with the increasing of the number of objectives [2]. The Pareto dominance relation based algorithms, such as NSGA-II [3] and SPEA2 [4], would fail to provide enough selection pressure to distinguish these solutions for searching towards the Pareto front.

Very recently, some non-Pareto-based techniques have been proposed specially for solving many-objective problems; such as, k-optimality, preference order ranking, favour relation, contraction-expansion, and so forth [2,5,6]. These methods commonly employ some other optimality relations replacing or enhancing Pareto dominance relation to increase the selection pressure, and they seem to perform well in terms of converging close to the optimum. However, as lack of effective diversity maintenance mechanism, the final sets obtained by these relations are usually just a subset of the Pareto optimal set [6]. In fact, as for many-objective problems, it is not a trivial job to provide sufficient selection pressure towards the Pareto front and at the same time maintain a good distribution of solutions. The conflict between the requirements of convergence and diversity is gradually aggravated with the growing of the number of objectives, due to the fact that the size of feasible objective space for a certain problem increasing with the dimensionality of the optimization problem [7].

Average ranking (AR) proposed by Bentley and Wakefield [8] is regarded as an alternative to rank individuals in multiobjective population, though the authors were not particularly concerned with many objective problems. In recent years, the AR method has been found to perform successfully in searching towards the Pareto front in many-objective optimization [5,6]. However, similar to the aforementioned non-Pareto-based methods, it often converges into a subset of Pareto front because of the lack of diversity maintenance mechanism [6]. In this paper, we incorporate a diversity maintenance strategy into AR to cover this shortage. We define a grid-based adaptive neighborhood for each individual in the population to preserve the suitable spacing among them. Moreover, a layering selection method utilizing it and AR is designed to pick out well-converged individuals and prohibit or postpone the archive of neighboring individuals.

The remainder is structured as follows. Section 2 describes the AR method and shows its properties. Section 3 is devoted to detail our diversity maintenance strategy. Section 4 provides a comparison of the proposed method versus the original AR and other high dimension optimization techniques. Finally, in Section 5 the results are summarized and directions for future line are pointed out.

2 Average Ranking

The average ranking method compares all individuals on each objective and ranks them independently. For a specific solution, a rank for each objective is assigned based on the level of its objective value among all solutions in the population. Thus each solution has M ranks (where M is the number of objectives), and the final rank is obtained by summing them. Table 1 illustrates the AR method with a simple example considering 4-objective solutions.

Clearly, AR is capable of distinguishing the nondominated solutions according to their ranks on different objectives. Additionally, it is also computationally simple and range-independent since all objective values are compared separately. Corne and Knowles have reported that AR outperforms some more complicated ranking strategies in terms of *cover metric* [5]. However, as lack of diversity maintenance mechanism, the population may converge into a subregion of the Pareto front. For instance, in Table 1 if the size of the population is 3, the winner

Table 1. An example of the AR method

solution	(f_1, f_2, f_3, f_4)	rank1	rank2	rank3	rank4	AR
A	(1, 1, 6, 5)	1	1	4	2	8
B	(1, 3, 5, 6)	1	3	2	3	9
C	(2, 2, 5, 6)	3	2	2	3	10
D	(6, 5, 1, 7)	4	4	1	5	14
E	(7, 5, 7, 1)	5	4	5	1	15

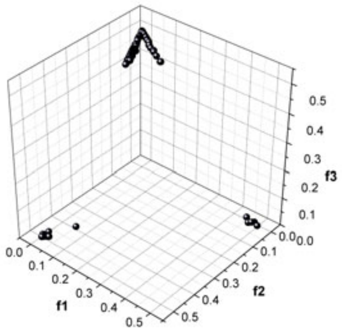


Fig. 1. Final solutions obtained by AR on DTLZ1

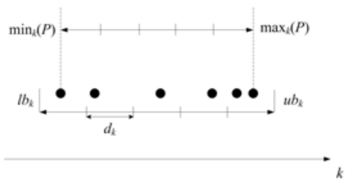


Fig. 2. Setting of grid in the kth objective

will be **A**, **B** and **C** according to the AR values of them. Unfortunately, though they seem to perform better in terms of convergence, they concentrate in a tiny region against the whole objective space, and have a high likelihood of evolving towards a local region of the optimal front. Figure 1 gives the final solutions set obtained by AR on 3-objective DTLZ1 problem.

3 The Proposed Diversity Maintenance Method

Grid technique has been widely used in the field of evolutionary multiobjective optimization. Many grid-based EMO algorithms have been proven to perform well in maintaining diversity when problems have two or three objectives. Here, we expand its potential to many-objective problems. First we fix a grid environment, where the population dwells.

3.1 Grid Setting

Borrowing from AGA [9], the grid is determined by the distribution of the current population. Fig. 2 illustrates the setting of grid in the kth objective.

First the minimum and maximum values of the objective k among the individuals in a population P are found and thus denoted as $\min_k(P)$ and $\max_k(P)$, respectively. Afterward, the lower and upper boundaries of grid in the kth objective are determined by them:

$$lb_k = \min_k(P) - (\max_k(P) - \min_k(P))/(2\times div) \quad (1)$$

$$ub_k = \max_k(P) + (\max_k(P) - \min_k(P))/(2\times div) \tag{2}$$

where *div* is a constant parameter, the number of divisions of the objective space in each dimension, set by the user (e.g., in Fig. 2 $div = 5$). Accordingly, the original M-dimensional objective space will be divided into div^M hyperboxes. Thus, the hyperbox width in the kth objective, d_k, could be formed as

$$d_k = (ub_k - lb_k)/div \tag{3}$$

Therefore, according to lb_k and d_k, the grid coordinate of any individual in the kth objective is determined as

$$G_k(\mathbf{A}) = \lfloor (F_k(\mathbf{A}) - lb_k)/d_k \rfloor \tag{4}$$

where $G_k(\mathbf{A})$ is the grid coordinate of individual $\mathbf{A}$ in the kth objective, $F_k(\mathbf{A})$ is the actual objective value in the kth objective. For instance, in Fig. 2, the grid coordinates of all individuals (from left to right) in the kth objective are 0, 1, 2, 3, 4 and 4, respectively.

3.2 Adaptive Neighborhood

Many existing grid-based EMO algorithms encounter difficulties in their scalability to many-objective optimization. One of the main reasons is that their density estimation mechanisms, which only consider the crowding of unit hyperbox in grid, may be invalid in high dimensional space. As the increase of objectives, the number of hyperboxes in grid will grow exponentially [5] (the number of hyperboxes in a k-objective problem is r^k, where r is the divisions in each dimension).

In this paper, we present an adaptive neighborhood based density estimation strategy to address this issue. The neighborhood of individuals here is composed of several hyperboxes around it, and the size of it will vary with the number of objectives. Specifically, for individual A, the neighborhood of it is defined as:

$$N(\mathbf{A}) = \left| \{X : \sum_{k=1}^{M} |G_k(\mathbf{A}) - G_k(X)| < M\} \right| \tag{5}$$

where $|\cdot|$ denotes the cardinality of a set, $G_k(\mathbf{A})$ implies the grid coordinate of individual $\mathbf{A}$ in the kth objective, $G_k(\boldsymbol{X})$ stands for the coordinate of hyperbox $\boldsymbol{X}$ in the kth objective, and M is the number of objectives. It is clear to note that the range of neighborhood of individual is determined by variable M. As M becomes larger, the number of hyperboxes in the neighborhood of individuals will increase steadily. This seems to be consistent with the total number of hyperboxes in grid environment. In the following, we describe the diversity maintenance method using the neighborhood.

3.3 Layering Selection

In this section, we introduce a layering selection approach integrating AR and adaptive neighborhood to determine the survival of individuals. The individual

Algorithm 1. *Layering Selection (P)*

```
Require: $P$(candidate set), $Q$(archive set), $N$(archive size), $CP$(current layer solutions
    set), $NP$(next layer solution set)
1: $Q \longleftarrow null, NP \longleftarrow null, CP \longleftarrow P$    /* Initialize sets Q, NP, and CP*/
2: while $|Q| < N$ do
3:   if $CP = null$ then
4:     $CP \leftarrow NP$
5:     $NP \leftarrow null$
6:   end if
7:   $q \leftarrow FindoutBest(CP)$ /* Find out the individual with the best AR value in CP*/
8:   $Q \leftarrow Q \bigcup \{q\}$    /* Put the best individual into archive set*/
9:   $CP \leftarrow CP \backslash \{q\}$    /* Remove the best individual from CP*/
10:  for all $p \in CP$ do
11:    if $G(p) \in N(q)$ then
12:      $NP \leftarrow NP \bigcup \{p\}$
         /* Add the individual who is the neighbor of q into NP and delete it from CP*/
13:      $CP \leftarrow CP \setminus \{p\}$
14:    end if
15:  end for
16: end while
17: return $Q$
```

with best AR value in current layer is selected to be archived firstly, and the neighbors of it (i.e., the individuals located in its neighborhood) will be demoted to next layer, no matter how good their ranks are. Algorithm 1 gives a detailed procedure of this approach.

The essential purpose of the algorithm is to prohibit or postpone the entry of adjacent individuals. Function *FindoutBest* (line 7) is designed to find out the best individual according to AR in current layer. The lines 10-15 of the algorithm is implemented a punishment to the neighbors of the best individual by relegating them to next layer. If the current layer is null, the next layer is activated to continue the above selection procedure (lines 3-6). Fig. 3 illustrates the algorithm with a simple example on 2-objective optimization problem. Initially, the current layer set contains individuals **A-H**. **G** is picked out firstly into the archive since it has the best AR value (7). Correspondingly, the neighbors (**B**, **C**, **D**, **E**, **F**) of **G** are degraded into the next layer (shown in Fig. 3(b)). Repeat this procedure until the current layer is null (shown in Fig. 3(e)). At this time, the next layer will be activated and turn into the current layer. So the best individual (**D**) in new current layer is selected, and the neighbors (**B**, **C**, **E**, **F**) of it enter the new next layer correspondingly (shown in Fig. 3(f)). Finally, the individuals in the archive are **A**, **D**, **G**, **H**, and **J**, when the vacancies are filled up.

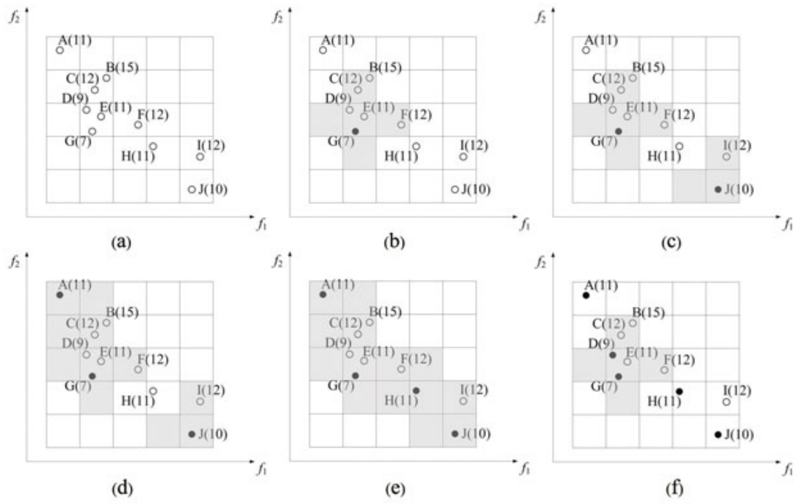


Fig. 3. An illustration of layering selection algorithm. Where archive size is set to 5. The value in the brackets corresponds to AR of individuals. Hollow points stand for the candidate individuals for archive and black points stand for the individuals that have selected into the archive set. Shadow area indicates the neighborhood of the selected individuals in current layer (i.e., the candidate individuals located in this area have been demoted to next layer)

4 Experimental Setup and Results

In this section, two diversity maintenance methods, improved Crowding Distance [10] and DMO [7], as well as Original AR algorithm are introduced to validate the proposed method. The improved crowding distance method assigns a zero distance (instead of an infinity distance) to extreme solutions in order to advance the convergence of algorithm [10]. DMO employs a diversity management operator to control or promote the diversity requirement. If the diversity indicator is smaller than 1 according to normal *maximum spread* test, the diversity promotion mechanism (i.e., crowding distance) is activated, conversely deactivated. For a fair comparison,the two methods are incorporated into AR. We note them as AR+CD$'$ and AR+DMO, respectively. The original AR algorithm, similar to [5], is implemented to select individuals according to AR for variation yet renew the archive in a random way. Anyway, the four algorithms (AR, AR+CD$'$, AR+DMO and the proposed method) adopt identical fitness strategy (AR) in selection for variation stage and yet adopt distinct diversity maintenance schemes (random selection, improved Crowding Distance, DMO and grid-based technique) in selection for survival stage. In addition, these algorithms are embeded into NSGA-II template for a fair comparison. Parent population combines with current population for generating the best half offspring. The last allowed nondominated front is considered by the above schemes instead of crowding distance. In the following, several performance metrics and test problem used in comparison are introduced in brief.

4.1 Performance Metrics and Test Problem

Usually, there are three goals that EMO algorithms can be identified and measured in performance [11]: (i) the distance of the resulting solutions to Pareto front (PF) should be minimized; (ii) a uniform distribution of the solutions found is desirable and (iii) the extent of the solutions should be maximized. In this paper, three performance metrics (CM [12], DM [12] and MS [13]), which directly evaluate each of the above goals, are considered.

The convergence metric CM calculates the average distance of the obtained solutions set away from the Pareto front. Similar to the studies in [10], the distance to the Pareto front is determined analytically without using a reference set. The uniformity metric DM measures the homogenization for a set of points. In DM, the obtained nondominated points are projected on a hyperplane, which is divided into a number of boxes. Depending on each box contains a point or not, the DM value is defined. DM takes the value between zero and one (one is the ideal result), and the larger value it achieves, the better is the uniformity. The detailed description of DM can be referred in [12]. The spread metric MS is an improved version of *Maximum Spread* considering the distribution of the Pareto front [13]. The original MS, which measures the length of the diagonal of the hypercube formed by the extreme objective values in a given set, may be influenced heavily by convergence of algorithm. The improved MS is devised to introduce the extreme values of the Pareto front for ease this effect. It also takes the value between zero and one and a higher value will tell about a larger extent of the obtained nondominated set.

To benchmark the performance of the four algorithms, the scalable function DTLZ2 [14] is invoked.The number of objectives used in this experiment is 3, 4, 6, 8, 10, 12, and 15. The total number of decision variables of the function is l=M+n−1. Where M is the number of objectives and n can be set by user to specify the distance to PF. According to [14], n=10 is used in DTLZ2.

4.2 Comparative Experiment

All compared algorithms are given real-valued decision variables. A crossover probability p_c=1.0 and a mutation probability p_m=1/l (where l is the number of decision variables) are used. The operators for crossover and mutation are simulated binary crossover (SBX) and polynomial mutation with the both distribution indexes 20. We run each algorithm independently 100 times. In each run a population of 100 individuals during 300 generations is predefined. For the proposed method, the parameter div setting for different number of objectives is shown in Table 2.

Tables 3, 4 and 5 give the convergence, uniformity and spread comparison respectively for all four algorithms over 3, 4, 6, 8, 10, 12, and 15 objectives. The values in the tables correspond to mean and standard deviation. In order to give a visual comparison, Figure 4 plots the distribution of the final solution set for

Table 2. The *div* setting of the proposed method

Objective number	3	4	6	8	10	12	15
Division	20	18	15	14	13	12	11

Table 3. CM comparison of the four EMOAs

Obj.	AR	AR + CD′	AR+DMO	P roposed Method
3	$0.002630_{(0.001219)}$	$0.007697_{(0.000969)}$	$0.003127_{(0.001131)}$	$0.002122_{(0.000512)}$
4	$0.000265_{(0.000368)}$	$0.019888_{(0.002790)}$	$0.009224_{(0.003546)}$	$0.005162_{(0.001205)}$
6	$0.000197_{(0.000161)}$	$0.061420_{(0.010694)}$	$0.076881_{(0.017319)}$	$0.011763_{(0.002084)}$
8	$0.000314_{(0.000427)}$	$0.464933_{(0.093250)}$	$0.166829_{(0.024957)}$	$0.166829_{(0.024957)}$
10	$0.000414_{(0.000522)}$	$1.175100_{(0.145026)}$	$0.258695_{(0.034931)}$	$0.031825_{(0.008503)}$
12	$0.000634_{(0.000454)}$	$1.489080_{(0.120641)}$	$0.335911_{(0.049553)}$	$0.065004_{(0.028429)}$
15	$0.001182_{(0.000610)}$	$1.686800_{(0.101894)}$	$0.507604_{(0.062442)}$	$0.144395_{(0.039438)}$

Table 4. DM comparison of the four EMOAs

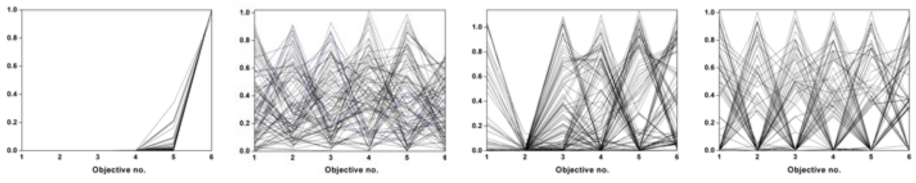
Obj.	AR	AR + CD′	AR+DMO	P roposed Method
3	$0.285620_{(0.061040)}$	$0.765828_{(0.033490)}$	$0.307042_{(0.092739)}$	$0.875861_{(0.038620)}$
4	$0.024635_{(0.006037)}$	$0.750573_{(0.056049)}$	$0.272139_{(0.073947)}$	$0.830936_{(0.043014)}$
6	$0.013498_{(0.004680)}$	$0.728653_{(0.062698)}$	$0.291955_{(0.053626)}$	$0.769867_{(0.036861)}$
8	$0.007968_{(0.002830)}$	$0.577113_{(0.095491)}$	$0.232654_{(0.035723)}$	$0.687197_{(0.015683)}$
10	$0.003084_{(0.001091)}$	$0.143857_{(0.032180)}$	$0.179114_{(0.015738)}$	$0.554510_{(0.018500)}$
12	$0.003356_{(0.000000)}$	$0.097137_{(0.019159)}$	$0.107763_{(0.011902)}$	$0.430962_{(0.031788)}$
15	$0.000625_{(0.000000)}$	$0.045647_{(0.023319)}$	$0.075254_{(0.024951)}$	$0.344784_{(0.040216)}$

Table 5. MS comparison of the four EMOAs

Obj.	AR	AR + CD′	AR+DMO	P roposed Method
3	$0.954843_{(0.101269)}$	$0.976454_{(0.015941)}$	$0.999668_{(0.001187)}$	$0.999982_{(0.000050)}$
4	$0.136769_{(0.045679)}$	$0.945741_{(0.018671)}$	$0.982307_{(0.031182)}$	$1.000000_{(0.000000)}$
6	$0.111958_{(0.035708)}$	$0.944685_{(0.019716)}$	$0.857206_{(0.048480)}$	$0.999716_{(0.000808)}$
8	$0.098869_{(0.032621)}$	$0.997528_{(0.005891)}$	$0.726130_{(0.024045)}$	$0.998269_{(0.005425)}$
10	$0.078272_{(0.034608)}$	$0.999999_{(0.000004)}$	$0.664174_{(0.030018)}$	$0.991779_{(0.017571)}$
12	$0.063128_{(0.022658)}$	$0.999996_{(1.177e-06)}$	$0.614995_{(0.033558)}$	$0.978742_{(0.031660)}$
15	$0.064900_{(0.019471)}$	$1.000000_{(1.583e-07)}$	$0.545415_{(0.029611)}$	$0.965983_{(0.039171)}$

four algorithms by parallel coordinates on the problem with 6 objectives. Inferred from the CM value in Table 3, the proposed method could in general reach the Pareto front of the problem with all considered number of objectives. The other two diversity maintenance methods perform well for 3, 4, and 6 objectives, but encounter difficulty in case of more objectives. Note that original AR obtains the better CM values than the proposed method in all case except the number of objectives is equal to 3. However, from Tables 4 and 5, this result is achieved at the cost of the loss of diversity. In most case, the final solution set obtained by AR locates in a microscopic part of the Pareto front.

Concerning uniformity assessment metric DM in Table 4, the proposed method achieves the best values on the problem with all considered number of objectives. AR+CD′ performs better than AR+DMO on the problem with 3, 4, 6, and 8 objectives, but slightly worse in case of higher dimension. The original AR algorithm obtains the worst results for all objectives. This is due to the final solutions



(a) AR (b) AR + CD′ (c) AR+DMO (d) Proposed method

Fig. 4. Distribution of final solution set by parallel coordinates on six-objective DTLZ2

set of it concentrated practically into a point, rather than distributed over the Pareto front.

Table 5 shows the spread comparison results from the MS metric. Clearly, the proposed method reaches the boundary of the whole Pareto front for all considered number of objectives. AR achieves a fairly good value on 3-objective problem, but fail in case of more objectives. Similarly, AR+DMO only performs well for the problem with 3 and 4 objectives. AR+CD′ can obtain a passable value for 3, 4 and 6 objectives problem. The proximity of the four algorithms to the boundary of Pareto front on 6-objective DTLZ2 could be seen in Figure 4. Additionally, it is interesting to note that the MS values obtained by AR+CD′ have a sudden raise when the number of objective reaches 8. This occurrence could be attributed to the reason that AR+CD′ fails to approximate to the Pareto front on 8 or more objectives problem. The maximum value in each objective obtained by it would exceed one, thereby producing a misleading result with respect to MS.

In summation, from the comparative studied above, we can conclude that the proposed method produces a good balance with regard to convergence, uniformity, and spread in the specific settings of grid parameter. Due to space limitations, we do not show results of it with different settings. Actually, in some preliminary trials, we found that grid division has more effect upon uniformity than convergence and spread, especially in lower dimension space (e.g., objective = 3, 4, or 6).

5 Conclusions

This paper has presented a diversity maintenance strategy for original AR algorithm to balance its convergence and diversity in evolutionary many-objective optimization. The proposed method has defined a grid-based adaptive neighborhood varied with the number of objectives to preserve a suitable spacing among individuals. Moreover, a layering approach integrating it and AR has been introduced to select the promising individuals for archive store. Simulation experiments have been studied by providing a detailed comparison with other three algorithms (AR, AR + CD′, and AR+DMO). The results reveal that the proposed algorithm has been successful in finding a near-optimal, uniformly distributed and well-extended solution set. Possible avenues of future work include the investigation of grid parameter, more many-objective test problems, and the incorporation of layer information to automatically tune the division setting according to the number of objectives.

References

1. Purshouse, R.C., Fleming, P.J.: Conflict, Harmony and Independence: Relationships in Evolutionary Multi-criterion Optimisation. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 16–30. Springer, Heidelberg (2003)
2. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: 2008 Congress on Evolutionary Computation (CEC 2008), pp. 2424–2431. IEEE Service Center, Hong Kong (2008)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002)
4. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design, Optimisation and Control, CIMNE, Barcelona, Spain, pp. 95–100 (2002)
5. Corne, D., Knowles, J.: Techniques for highly multiobjective optimization: Some non-dominated points are better than others. In: Proc. of 2007 Genetic and Evolutionary Computation Conference (GECCO 2007), London, pp. 773–780 (July 2007)
6. Jaimes, A.L., Quintero, L.V.S., Coello Coello, A.C.: Ranking Methods in Many-objective Evolutionary Algorithms. In: Chiong, R. (ed.) Nature-Inspired Algorithms for Optimisation, pp. 413–434. Springer, Berlin (2009)
7. Adra, S.F., Fleming, P.J.: A Diversity Management Operator for Evolutionary Many-Objective Optimisation. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 81–94. Springer, Heidelberg (2009)
8. Bentley, P.J., Wakefield, J.P.: Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In: Soft Computing in Engineering Design and Manufacturing, Part 5, London, pp. 231–240 (June 1997)
9. Knowles, J., Corne, D.: Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. IEEE Trans. Evol. Comput. 7(2), 100–116 (2003)
10. Wagner, T., Beume, N., Naujoks, B.: Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 742–756. Springer, Heidelberg (2007)
11. Li, M., Zheng, J.: Spread Assessment for Evolutionary Multi-Objective Optimization. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 216–230. Springer, Heidelberg (2009)
12. Deb, K., Jain, S.: Running Performance Metrics for Evolutionary Multi-Objective Optimization. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), pp. 13–20 (2002)
13. Goh, C.K., Tan, K.C.: An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization. IEEE Trans. Evol. Comput. 11(3), 354–381 (2007)
14. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. in Evolutionary Multiobjective Optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Theoretical Advances and Applications, pp. 105–145. Springer, Berlin (2005)

Objective Space Partitioning Using Conflict Information for Many-Objective Optimization

Antonio López Jaimes[1], Hernán Aguirre[2], Kiyoshi Tanaka[2], and Carlos A. Coello Coello[1]

[1] CINVESTAV-IPN, Computer Science Department, Mexico City 07360, Mexico
tonio.jaimes@gmail.com, ccoello@cs.cinvestav.mx
[2] Shinshu University, Faculty of Engineering, Nagano 380-8553, Japan
{ahernan,ktanaka}@shinshu-u.ac.jp

Abstract. Here, we present a partition strategy to generate objective subspaces based on the analysis of the conflict information obtained from the Pareto front approximation found by an underlying multi-objective evolutionary algorithm. By grouping objectives in terms of the conflict among them, we aim to separate the multi-objective optimization into several subproblems in such a way that each of them contains the information to preserve as much as possible the structure of the original problem. The ranking and parent selection is independently performed in each subspace. Our experimental results show that the proposed conflict-based partition strategy outperforms NSGA-II in all the test problems considered in this study. In problems in which the degree of conflict among the objectives is significantly different, the conflict-based strategy achieves its best performance.

1 Introduction

Since the first implementation of a Multi-objective Evolutionary Algorithm (MOEA) in the mid 1980s, a wide variety of approaches have been proposed, gradually improving in both their effectiveness and their efficiency to solve multi-objective problems (MOPs) [1]. However, recent experimental and analytical studies have shown that MOEAs based on Pareto optimality scale poorly when the number of objectives is increased (this is called a *many-objective problem*) [2]. Approaches to deal with such problem have mainly focused on the use of alternative optimality relations [3,4], reduction of the number of objectives of the problem, either during the search process [5,6] or, at the decision making process [7,8,9], and the incorporation of preference information [2].

A general scheme for partitioning the objective space in several subspaces in order to deal with many-objective problems was introduced in [10]. In this approach the solution ranking and parent selection are independently performed in each subspace to emphasize the search within smaller regions of objective function space. Here, we propose a new partition strategy that creates objective subspaces based on the analysis of the conflict information obtained from the Pareto front approximation found by the underlying MOEA. By grouping objectives in terms of the conflict among them, we aim to separate the MOP into several subproblems in such a way that each subproblem contains the information to preserve as much as possible the structure of the original problem.

Our approach is more closely related to the objective reduction approaches, specially those adopted during the search. However, its main difference with respect to them is the incorporation of all the objectives in order to cover the entire Pareto front. Deb and Saxena [7] proposed a method for reducing the number of objectives based on principal component analysis. Although some modifications can be made to this method in order to use it during the search, this method was designed as an *a posteriori* method. Brockhoff and Zitzler [5], and López Jaimes et al. [6] used similar objective reductions algorithms incorporated into a MOEA. However, in both cases, the non-conflicting objectives were discarded or aggregated to form a single objective.

2 Basic Concepts and Notation

Definition 1 (Objective space Φ). *The objective space of a MOP is the set $\Phi = \{f_1, f_2, \ldots, f_M\}$ of the M objective functions to be optimized.*

Definition 2 (Subspace ψ). *A subspace ψ of Φ is a lower dimensional space that includes some of the objective functions in Φ, i.e. $\psi \subset \Phi$.*

Definition 3 (Space partition Ψ). *A space Φ is said to be partitioned into N_S subspaces, denoted as Ψ, if $\Psi = \{\psi_1, \psi_2, \ldots, \psi_{N_S} \mid \cup_{i=1}^{N_S} \psi_i = \Phi \wedge \cap_{i=1}^{N_S} \psi_i = \emptyset\}$.*

Definition 4 (Pareto dominance relation). *A solution $\mathbf{x}^1$ is said to Pareto dominate solution $\mathbf{x}^2$ in the objective space Φ, denoted by $\mathbf{x}^1 \prec \mathbf{x}^2$, if and only if (assuming minimization): $\forall f_i \in \Phi : f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2) \wedge \exists f_i \in \Phi : f_i(\mathbf{x}^1) < f_i(\mathbf{x}^2)$.*

Definition 5 (Pareto optimal set). *The Pareto optimal set, P_{opt}, is defined as: $P_{\text{opt}} = \{\mathbf{x} \in \mathcal{X} \mid \nexists\, \mathbf{y} \in \mathcal{X} : \mathbf{y} \prec \mathbf{x}\}$, where $\mathcal{X} \in \mathbb{R}^n$ is the variable space.*

Definition 6 (Pareto front). *For a Pareto optimal set P_{opt}, the Pareto front, PF_{opt}, is defined as: $PF_{\text{opt}} = \{\mathbf{z} = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \mid \mathbf{x} \in P_{\text{opt}}\}$. We will denote by PF_{approx} the Pareto front approximation achieved by a MOEA.*

Definition 7 (Sample Correlation coefficient). *The sample correlation coefficient, r_{XY}, is defined by $r_{XY} = \sum_{i=1}^{m}(X_i - \bar{X})(Y_i - \bar{Y})/(m-1)s_X s_Y$, where $s_X > 0$ and $s_Y > 0$ denote the sample standard deviations for the data sets X and Y, respectively, and m is the number of elements of each data set.*

3 The Conflict-Based Partitioning Framework

3.1 General Idea of the Partitioning Framework

The basic idea of the partitioning framework is to divide the objective space into several subspaces so that a different portion of the population focuses the search in a different subspace. By partitioning the objective space into subspaces, we aim to emphasize the search within smaller regions of objective space. Instead of dividing the population into independent subpopulations, a fraction of the pool of parents for the next generation is selected based on a different subspace. This way, the pool of parents will be composed with individuals having a good performance in each subspace. In our approach, we partition the M-dimensional space $\varPhi = \{f_1, f_2, \ldots, f_M\}$ into N_S non-overlapping subspaces $\varPsi = \{\psi_1, \psi_2, \ldots, \psi_{N_S}\}$. We selected NSGA-II to implement our proposed partitioning framework. Thus, the nondominated sorting and truncation procedures of NSGA-II are modified in the following way. The union of the parents and offspring, $\mathcal{P} \cup \mathcal{Q}$, is sorted N_S times using a different subspace each time. Then, from each mixed sorted population, the best $|\mathcal{P}|/N_S$ solutions are selected to form a new parent population of size $|\mathcal{P}|$. After this, the new population is generated by means of recombination and mutation using binary tournaments. Algorithm 1 shows this procedure.

3.2 A New Partition Strategy

The number of all possible ways to partition $\varPhi$ into N_S subspaces is very large. Therefore, it is not feasible to search in all the possible subspaces. Instead, we can define a schedule of subspace sampling by using a partition strategy. In [10] three strategies to partition $\varPhi$ were investigated: random, fixed, and shift partition. Here, we investigate a new strategy using the conflict information among objectives. Namely, the first partition would contain the least conflicting objectives, the second one the next least conflicting objectives, and so on. Therefore, instead of removing the least conflicting objectives, we integrate those objectives to form subspaces in such a way that all the objectives are optimized. By grouping objectives in terms of the conflict among them, we are trying to separate the MOP into subproblems in such a way that each subspace contains information to preserve most of the structure of the original problem. We propose using the correlation among solutions in PF_{approx} to estimate the conflict among objectives. A negative correlation between a pair of objectives means that one objective increases while the other decreases and vice versa. Thus, a negative correlation estimates the conflict between a pair of objectives. On the other hand, if the correlation is positive, then both objectives increase or decrease at the same time. That is, the objectives support each other.

In order to implement the new partition strategy we should take into account that the conflict relation among the objectives changes during the search. To deal with this situation we suggest a new partitioning framework in which the search is divided in several stages. Each of these stages is divided in two phases,

Algorithm 1. Procedure of non-dominated sort and truncation.

```
procedure SORT&TRUNCATION($\mathcal{R}, \mathcal{P}, \varPsi$)
    $\mathcal{P}^* \leftarrow \emptyset$
    for $i \leftarrow 1$ until $|\varPsi|$ do
        $\mathcal{F}^{\psi_i} \leftarrow$ NONDOMINATEDSORT($\mathcal{R}, \psi_i$)
        CROWDING($\mathcal{F}^{\psi_i}, \psi_i$)
        $\mathcal{P}^{\psi_i} \leftarrow$ TRUNCATION($\mathcal{F}^{\psi_i}, |\mathcal{P}|/|\varPsi|$)    $\triangleright$ $|\mathcal{P}^{\psi_i}| = |\mathcal{P}|/|\varPsi|$
        $\mathcal{P}^* \leftarrow \mathcal{P}^* \cup \mathcal{P}^{\psi_i}$
    return $\mathcal{P}^*$    $\triangleright$ $|\mathcal{P}^*| = |\mathcal{P}|$
```

Algorithm 2. Pseudocode of our proposed partitioning MOEA.

Input: Evolutionary operators values, N_S (Num. of subspaces).
Output: Pareto front approximation.

```
$\mathcal{P}_0 \leftarrow$ RANDOMPOPULATION()
EVALUATE($\mathcal{P}_0$)
CROWDING($\mathcal{P}_0$)
$integrationPhase = TRUE$
for $t \leftarrow 1$ until $G_{\max}$ do
    $\mathcal{Q}_t \leftarrow$ NEWPOP($\mathcal{P}_t$)    ▷ selection, crossover and mutation.
    EVALUATE($Q_t$)
    $\mathcal{R}_t \leftarrow \mathcal{P}_t \cup \mathcal{Q}_t$
    if $integrationPhase = TRUE$ then
        $\mathcal{P}_{t+1} \leftarrow$ SORT&TRUNCATION($\mathcal{R}_t, \mathcal{P}_t, \{\Phi\}$)
        if $g \geq G_\Phi$ then
            $integrationPhase = FALSE$
            $g \leftarrow 0$
    else
        if $g = 1$ then
            $\Psi \leftarrow$ CONFLICTPARTITION($\mathcal{P}, \Phi, N_S$)
        $\mathcal{P}_{t+1} \leftarrow$ SORT&TRUNCATION($\mathcal{R}_t, \mathcal{P}_t, \Psi$)
        if $g \geq G_\Psi$ then
            $integrationPhase = TRUE$
            $g \leftarrow 0$
    $g \leftarrow g + 1$
```

Algorithm 3. Partitioning Using Conflict Information.

```
procedure CONFLICTPARTITION($\mathcal{P}, \Phi, N_S$)
    cMatrix $\leftarrow$ COMPUTECONFLICTMATRIX($\mathcal{P}$)
    $k \leftarrow (|\Phi|/N_S) - 1$
    $\Phi' \leftarrow \Phi = \{f_1, \ldots, f_M\}$
    for 1 until $N_S - 1$ do
        for each objective $f_i$ in $\Phi'$ do
            $V_{f_i} \leftarrow$ Ascending ordered list of $k$-nearest neighbors of $f_i$ wrt conflict.
        $V^\star \leftarrow V_{f_i} \cup \{f_i\} : \forall f_j \in \Phi',\ V_{f_i}[k] \leq V_{f_j}[k]$
        $\Psi_{N_S} \leftarrow \Psi_{N_S} \cup V^\star$
        $\Phi' \leftarrow \Phi' - V^\star$.
    $\Psi_{N_S} \leftarrow \Psi_{N_S} \cup \Phi'$.
```

namely, an approximation phase followed by a partitioning phase. In the approximation phase all the objectives are used to select the new parent population. The goal of this phase is finding a good approximation of the current PF_{opt}. The proposed procedure is described in Algorithm 2.

3.3 Partitioning Using Conflict Information

Since we are interested in measuring the negative correlation between objectives, the correlation matrix was modified so that each entry, r_{f_i,f_j}, contains the value $1 - r_{f_i,f_j}$. Thus, each value of this new "conflict matrix" is in the range $[0, 2]$. A value of zero indicates that objectives f_i and f_j are not in conflict at all, and a value of 2 indicates that they are completely in conflict. The procedure to create the subspaces is presented in Algorithm 3.

4 Experimental Results

4.1 Algorithms, Metrics and Parameter Settings

Since we wish to investigate the advantages and disadvantages of the conflict-based strategy with respect to a random strategy which creates the partitions at random, we compare the original NSGA-II with the NSGA-II using the conflict-based strategy and the random strategy. In all the algorithms we use a population of 200 individuals running during 200 generations. The results presented are the average over 30 runs of each MOEA. In the conflict-based strategy, the search is divided in 10 stages, and the values for G_{Φ} and G_{Ψ} represent the 30% and 70% of the generations of each stage, respectively.

In order to show how the conflict-based strategy works, we will use a test problem in which the conflicting objectives can be defined *a priori* by the user. Namely, the problem DTLZ5(I, M) [7], where M is the total number of objectives, and I is the number of objectives in conflict. Additionally, we employ the 0/1 Knapsack with 300 items since the conflict relation among its objectives is not known *a priori*. Unless specified otherwise, in our experiments we use from 4 to 15 objectives in each test problem. For 4-9 objectives we use 2 subspaces, and for 10-15 objectives, we use 3 subspaces. In order to assess convergence we adopt generational distance (GD). In the case of DTZL5(I, M) we use the exact generational distance, namely $GD = \frac{1}{m}\sum_{\mathbf{z}\in PF_{\text{approx}}}\sum_{j=1}^{M}(z_j)^2 - 1$, where $m = |PF_{\text{approx}}|$. In the case of the Knapsack problem, the generational distance is computed using as our reference Pareto front, the non-dominated set resulting of the union of the PF_{approx} sets obtained by the three algorithms in all the runs. Additionally, to directly compare the convergence of the MOEAs, we utilize the additive ϵ-indicator [11]. In order to evaluate diversity, we adopt the inverted generational distance (IGD). Finally, to assess both convergence and diversity, we adopt the hypervolume indicator. For DTLZ5(I, M) the reference point was $\mathbf{z}^{\text{ref}} = 1.5^M$. For the Knapsack problem, the reference point was formed using the worst value in each objective of all the PF_{approx} generated by all the algorithms.

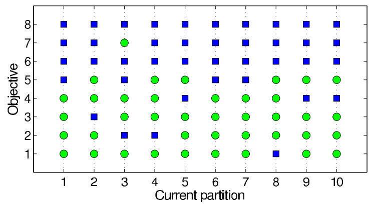
4.2 DTLZ5(I, M): Conflict Known *a Priori*

In these experiments we use $I = 4$ conflicting objectives from a total of $M = 4, \ldots, 15$ objectives. For 4-9 objectives, 2 subspaces are used, whereas for 10-15 objectives, we employ 3 subspaces. First, we show that the conflict-based strategy is able to identify the conflicting objectives in most of the partitions generated during the search process. Fig. 1 shows the subspaces generated by the conflict-based and the random partition strategies during the search process. In this example, there is a total of $M = 8$ objectives. The conflicting objectives are objectives 6-8 and any other objective. The objectives in the most conflicting subspace are denoted by squares, and the other subspace is denoted by circles.

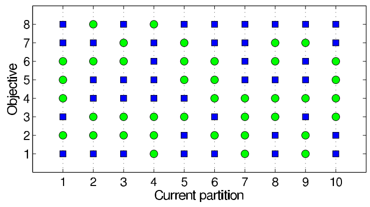
As the search progresses, the input PF_{approx} used to estimate the conflict approaches the true Pareto front. Therefore, as can be seen in Fig. 1(a), in the last stages of the search, the conflict-based strategy was able to create the correct partition. On the other hand, by using the random strategy (Fig. 1(b)), only one of the generated partitions contains the correct subspaces. Consequently, in most of the generations of the search, the selected parents emphasize objective subspaces that do not maximize the contribution to form the true Pareto front. By inspecting the parallel coordinate plot presented in Fig. 2 we realize that NSGA-II with the random strategy converges to the extremes of the Pareto front. That is, most of the solutions are close to 0 or 1 in one objective, but very few solutions are in the middle. In contrast, the conflict-based strategy covers all the trade-offs among the objectives. In order to quantify this situation, we compute the IGD. Fig. 3 shows that the conflict-based partition strategy achieves better values in terms of IGD.

This indicates a better distribution using the conflict-based partition strategy. In addition, the convergence of NSGA-II degrades dramatically when the number of objectives is more than 6. A possible reason of this behavior is the generation of dominance resistant solutions in DTLZ5(I, M). In contrast, the IGD values using any of the partition strategies, are not affected by the number of objectives. In particular, we can see that the convergence obtained by using the conflict-based partition strategy is better than the one achieved by the random strategy.

The results of the ϵ-indicator are presented in the matrices of subplots of Fig. 4. $I_{\epsilon+}(A, B)$ is the subplot located in row A and column B of the matrix.



(a) Conflict-based partition strategy.



(b) Random partition strategy.

Fig. 1. Subspaces generated using the conflict and random partition strategies in DTLZ5$(I = 4, M = 8)$. Objectives 6-8 and any other are the conflicting objectives.

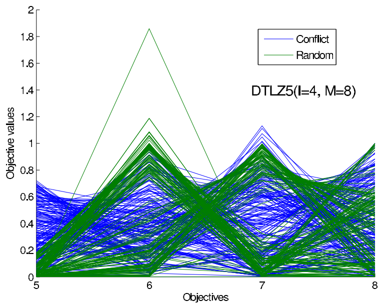


Fig. 2. Parallel coordinate plot of the PF_{approx} obtained with the random and the conflict partition strategies

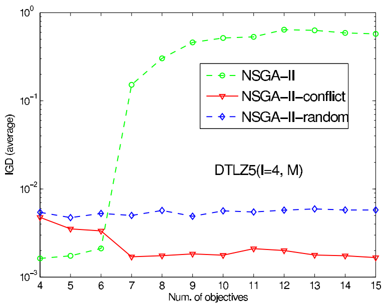


Fig. 3. IGD for DTLZ5(I=4,M). For 4-9 objs. we used a partition with 2 subspaces, and for 10-15 objs., one with 3.

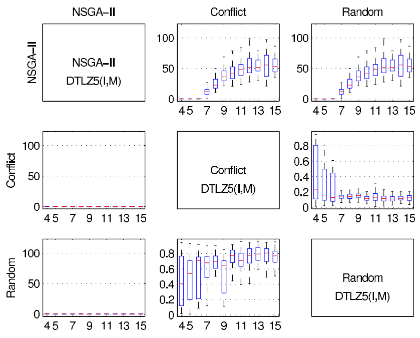


Fig. 4. ϵ-indicator results. The horizontal axis denotes the number of objectives.

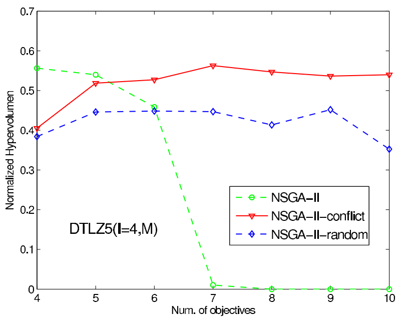


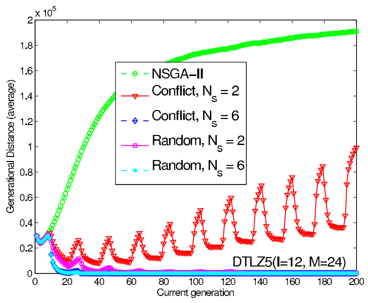
Fig. 5. Normalized hypervolume results on DTLZ5(I, M)

As we can see, NSGA-II is clearly outperformed by the NSGA-II using any of the partition strategies. In turn, we can observe that the conflict-based strategy is better than the random strategy, specially for 6 or more objectives. Since the hypervolume considers both convergence and distribution, as we can see in Fig. 5, the conflict-based partition strategy outperforms the random strategy. For less than 5 or 6 objectives, NSGA-II presents a better or similar performance than that achieved by using a partition strategy. There are two causes for this behavior. Firstly, that the NSGA-II is still able to deal with that lower number of objectives. Second, since there are 4 conflicting objectives for 4-6 objectives, using 2 subspaces is not possible that all the conflicting objectives are grouped in one subspace. This suggests that is convenient to assign all the highly correlated objectives to a single subspace. However, a large subspace might surpass the capacities of the underlying MOEA.

4.3 Effect of the Size of the Subspaces

In this section we analyze if it is better to have all the conflicting objectives together in a large subspace, or small subspaces in which the conflicting objectives are in different subspaces. To this end, we used DTLZ5$(I = 12, M = 24)$ to compare two partitions, namely, one with two subspaces with 12 objectives each, and another one with 6 subspaces with 4 objectives each. Fig. 6 shows the progress of GD during all the search process. We want to emphasize the fact that each partition strategy achieved a better convergence using 6 subspaces with 4 objectives. This suggests that is preferable to have subspaces of moderate size, even if highly conflicting objectives have to be assigned to different subspaces. The optimal size of the subspaces depends on the capacities of the underlying MOEA. For example, based on the experimental results, an appropriate size of the subspaces for NSGA-II would be between 4 and 6 objectives.

As in previous experiments, using parallel coordinate plots we realized that the solutions using the random strategy converge to the extremes of some objectives. To quantitatively assess the distribution, we compare the algorithms using IGD (Table 1). Although the obtained GD of the conflict and random strategies are



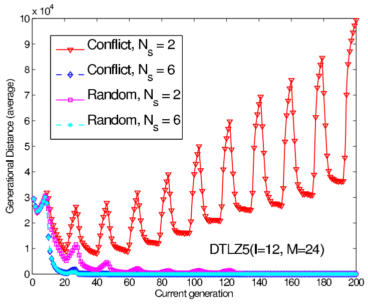


Fig. 6. Online GD using a partition with 2 subspaces and another one with 6 subspaces

Table 1. IGD values for using 2 and 6 subspaces in each of the partitioning strategies, namely, random- and conflict-based partitions

	NSGA-II	**Conflict**		**Random**	
		$N_S = 2$	$N_S = 6$	$N_S = 2$	$N_S = 6$
Average	0.18005	0.00838	**0.00570**	0.00682	0.00769
Std. Dev.	0.04695	0.00010	0.00047	0.00092	0.00029

similar using 6 subspaces (Fig. 6), the results of IGD suggest that the conflict strategy with 6 subspaces achieved a better distribution of the solutions than the random strategy with 6 subspaces.

4.4 Knapsack Problem: Unknown Conflict *a Priori*

In the Knapsack problem there is an interesting conflict relation among the objectives that allows the conflict-based strategy performing better than the random strategy. Fig. 7 shows the subspaces generated by the conflict strategy in the Knapsack problem. As we can see, as the search progresses, a particular partition is formed repeatedly, namely $\Psi_3 = \{\{4,5,8\},\{1,3,9\},\{2,6,7\}\}$. This suggests that the conflict among certain objectives is considerably larger than the conflict among others. In order to measure the contribution of each subspace to the total conflict in the problem, we compute for each subspace its "conflict degree", i.e., the sum of the conflict between each pair of objectives.

The ratio of the conflict degree of each subspace and the total conflict is called the *conflict contribution*. In Fig. 8, we can clearly see that subspace 3 has a larger conflict contribution with respect to the other subspaces. From the results obtained in GD and IGD (see Fig. 9) we can say that the conflict-based partition strategy achieved better Pareto front approximations than the random-based strategy both in terms of convergence and distribution. The results obtained with the hypervolume indicator (Fig. 10) confirm that the conflict-based strategy outperformed the random strategy. We can conclude that the differences in the degrees of conflict between each pair objectives was used by the conflict-based strategy to obtain better results than those obtained using a random partition.

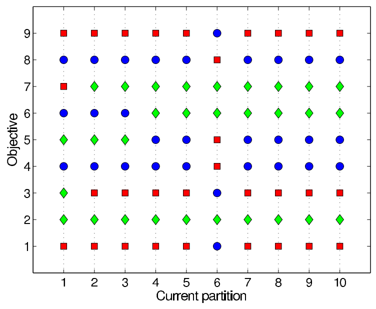


Fig. 7. Generated subspaces by the conflict-based partition strategy on the Knapsack problem

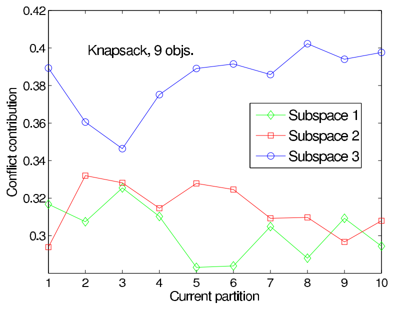


Fig. 8. Conflict contribution of each of the three subspaces generated using the conflict partition strategy

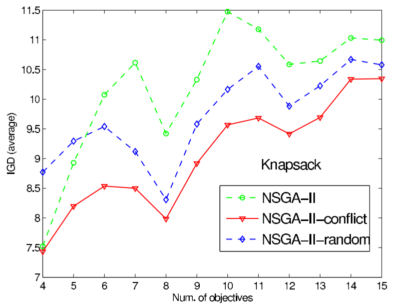


Fig. 9. Inverted generational distance in the Knapsack problem

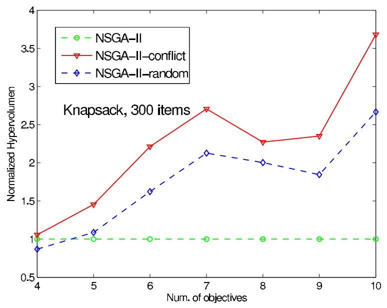


Fig. 10. Normalized Hypervolume wrt the one achieved by the NSGA-II

5 Conclusions and Future Work

The experimental results showed that both the conflict-based and random partition strategies outperformed NSGA-II in all the test problems considered in this study. While NSGA-II diverges in some test problems, the NSGA-II using any of the partition strategies maintains a good convergence despite the number of objectives. Regarding the two partition strategies, the conflict-based partition strategy achieved a better distribution of the solutions than the random strategy. In some problems, by using the random strategy, the solutions converged to the extremes of the Pareto front. In problems in which the degree of conflict among the objectives was different, the conflict-based strategy presented a better performance. It is important to note that in the Knapsack problem, where the conflict relation among the objectives is not known *a priori*, the conflict-based strategy was able to detect important dependencies among the objectives in terms of the conflict. Another finding is that the best size of the subspaces considerably depends on the scalability of the underlying MOEA. As part of our future work, we plan to exploit the conflict information to automatically adapt the proportion of resources granted to each subspace.

Acknowledgements. The first author acknowledges support from CONACyT to pursue graduate studies in Computer Science at CINVESTAV-IPN. The fourth author acknowledges support from CONACyT project no. 103570.

References

1. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007), ISBN 978-0-387-33254-3
2. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: CEC 2008, Hong Kong, pp. 2424–2431. IEEE Service Center, Los Alamitos (2008)
3. Farina, M., Amato, P.: On the Optimal Solution Definition for Many-criteria Optimization Problems. In: Proceedings of the NAFIPS-FLINT International Conference 2002, Piscataway, New Jersey, pp. 233–238. IEEE Service Center, Los Alamitos (June 2002)
4. Sato, H., Aguirre, H.E., Tanaka, K.: Controlling Dominance Area of Solutions and Its Impact on the Performance of MOEAs. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 5–20. Springer, Heidelberg (2007)
5. Brockhoff, D., Zitzler, E.: Improving Hypervolume-based Multiobjective Evolutionary Algorithms by Using Objective Reduction Methods. In: CEC 2007, Singapore, pp. 2086–2093. IEEE Press, Los Alamitos (September 2007)
6. López Jaimes, A., Coello Coello, C.A., Urías Barrientos, J.E.: Online Objective Reduction to Deal with Many-Objective Problems. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 423–437. Springer, Heidelberg (2009)
7. Deb, K., Saxena, D.K.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: CEC 2006, Vancouver, BC, Canada, pp. 3353–3360. IEEE Press, Los Alamitos (2006)
8. Brockhoff, D., Zitzler, E.: Are All Objectives Necessary? On Dimensionality Reduction in Evolutionary Multiobjective Optimization. In: Parallel Problem Solving from Nature IX, pp. 533–542. Springer, Heidelberg (2006)
9. López Jaimes, A., Coello Coello, C.A., Chakraborty, D.: Objective Reduction Using a Feature Selection Technique. In: 2008 Genetic and Evolutionary Computation Conference (GECCO 2008), Atlanta, USA, pp. 674–680. ACM Press, New York (2008)
10. Aguirre, H.E., Tanaka, K.: Many-Objective Optimization by Space Partitioning and Adaptive e-Ranking on MNK-Landscapes. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 407–422. Springer, Heidelberg (2009)
11. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)

How Crossover Speeds Up Evolutionary Algorithms for the Multi-criteria All-Pairs-Shortest-Path Problem

Frank Neumann[1] and Madeleine Theile[2]

[1] Max-Planck-Institut für Informatik, Campus E 1 4, 66123 Saarbrücken, Germany
[2] Technische Universität Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany

Abstract. Understanding the impact of crossover in evolutionary algorithms is one of the major challenges in the theoretical analysis of these stochastic search algorithms. Recently, it has been shown that crossover provably helps to speed up evolutionary algorithms for the classical all-pairs-shortest path (APSP) problem. In this paper, we extend this approach to the NP-hard multi-criteria APSP problem. Based on rigorous runtime analyses, we point out that crossover leads to better worst case bounds than previous known results. This is the first time that rigorous runtime analyses have shown the usefulness of crossover for an NP-hard multi-criteria optimization problem.

1 Introduction

Stochastic search algorithms such as evolutionary algorithms [7] and ant colony optimization [4] have found many application for complex combinatorial optimization problems. In contrast to the variety of application domains and many successful approaches for different kind of problems, the theoretical understanding lacks far behind the practical success. Analyzing stochastic search algorithms with respect to their runtime behavior has become a major branch in the theoretical analysis of these algorithms. Starting with results on simple pseudo-Boolean functions (see e. g. [12,5,9]), different results have been obtained for classical combinatorial optimization problems such as shortest paths, minimum spanning trees, or maximum matchings (see [11] for an overview). The goal of all these studies is to increase the understanding of stochastic search algorithms in a rigorous way and provide guidelines for the application of these methods.

Recently, evolutionary algorithms have been analyzed for the basic all-pairs-shortest-path (APSP) problem. It has been shown that the use of crossover leads to a better bound on the expected optimization time compared to an algorithm that is just based on mutation [2,3]. The question arises whether these results can be generalized to NP-hard variants of the APSP problem ([6]). We answer this question in a positive way and show that the approach introduced in [2] leads to a good approximation for the multi-criteria APSP problem. For the multi-criteria APSP problem, the task is to approximate for each pair of distinct vertices the set of Pareto optimal paths.

Coping with multi-objective problems one is often faced with the fact that the number of Pareto optimal objective vectors might be exponential with respect to the problem size. Due to this it is not possible to compute the whole Pareto front in polynomial time. As we are only interested in good approximations for the multi-criteria APSP problem, we investigate evolutionary algorithms that make use of the ε-dominance approach [10]. This has already been shown to be successful for the multi-criteria single-source shortest path (SSSP) problem [8]. Using the ε-dominance approach, the population size can be controlled by a parameter ε that determines the quality of the approximation to be achieved. Note, that the multi-criteria APSP problem can be solved by computing n times the multi-criteria SSSP problem where n denotes the number of vertices of the given input graph.

We generalize the model for the APSP problem given in [2] to the multi-criteria APSP problem and analyze the corresponding multi-objective evolutionary algorithm with respect to the runtime behavior. Our results are runtime bounds that generalize the ones given in [3] to the multi-objective case. Comparing these results for 2 and 3 objectives to the ones that can be obtained by applying n times the approach for the multi-criteria single-source-shortest path (SSSP) problem [8], we show that our approach yields better runtime guarantees. The reason for this is the use of a crossover operator which allows the algorithm to work with a smaller population size.

The outline of the paper is as follows. In Section 2, we introduce the problem and the algorithm under consideration. We carry out a rigorous runtime analysis which shows the impact of crossover for the computation of multi-criteria all-pairs-shortest-paths in Section 3. Finally, we finish with some concluding remarks.

2 Problem and Algorithm

Computing shortest paths in a given graph is one of the fundamental problems in computer science. We consider the multi-criteria all-pairs-shortest-path (APSP) problem. The input is a connected directed graph $G = (V, E)$ with n vertices and m edges, and a weight function $w \colon E \rightarrow (\mathbb{R}^+)^d$ which assigns to each edge a vector of $d \geq 2$ weights. We denote by $w_{\max} = \max_{i=1}^{d}(\max_{e \in E} w_i(e))$ the maximum weight of the given input. In addition, we assume that there are no negative cycles with respect to each dimension of the weight function.

Let $P_{u,v}$ be a path from u to v in G and

$$w(P_{u,v}) = (w_1(P_{u,v}), \ldots, w_d(P_{u,v}))$$

the corresponding objective vector which gives the different paths weights with respect to the d objective functions. Comparing two paths $P_{u,v}$ and $P'_{u,v}$, we write $w(P_{u,v}) \leq w(P'_{u,v})$ iff $w_i(P_{u,v}) \leq w_i(P'_{u,v})$, $1 \leq i \leq d$. Similarly, we write $w(P_{u,v}) < w(P'_{u,v})$ iff $w(P_{u,v}) \leq w(P'_{u,v})$ and $w_i(P_{u,v}) < w_i(P'_{u,v})$ for at least one i.

Considering a multi-objective problem, one is interested in computing or approximating the Pareto front. $P_{u,v}$ is called Pareto optimal if there is no other path $P'_{u,v} \neq P_{u,v}$ from u to v for which $w(P'_{u,v}) < w(P_{u,v})$ holds. The Pareto front $F_{u,v}$ for a given pair of vertices u, v is the set of all objective vectors for which a Pareto optimal path from u to v exists. The Pareto front for the multi-criteria APSP problem is constituted by the Pareto fronts $F_{u,v}$ for each distinct pair of vertices u and v. Computing the Pareto front for the multi-criteria APSP problem is NP-hard iff $d \geq 2$ (see [6]). On the other hand, the number of Pareto optimal objective vectors might grow exponentially with respect to the given input.

Due to this, we are interested in good approximations for the problem. In this case, it is only necessary to compute a smaller set of solutions. Our goal is to compute for each pair of distinct vertices u and v and each Pareto optimal path $P_{u,v}$ an $(1+\varepsilon)$-approximation $P'_{u,v}$ of $P_{u,v}$, i. e. a path $P'_{u,v}$ for which $w(P'_{u,v}) \leq (1+\varepsilon) \cdot w(P_{u,v})$ holds. Here $\varepsilon > 0$ is a parameter that determines the quality of the approximation.

This notion of a multiplicative $(1+\varepsilon)$-approximation is very common when considering approximations and relates well to the concept of ε-dominance used in evolutionary multi-objective optimization. The concept of ε-dominance is implemented in the algorithm in terms of hyperboxing (dividing the objective space into hyperboxes) and box-domination. Let $r > 1$ be a parameter determining the size of the hyperboxes. The box value $b_r(P_{u,v})$ of path $P_{u,v}$ is given by

$$b_r(P_{u,v}) := (\lfloor \log_r(w_1(P_{u,v})) \rfloor, \lfloor \log_r(w_2(P_{u,v})) \rfloor, \ldots, \lfloor \log_r(w_d(P_{u,v})) \rfloor).$$

For the multi-criteria APSP problem we investigate an algorithm called Diversity Evolutionary Multi-objective Optimizer (DEMO) which is an extension of the population-based approach for the single-criteria APSP problem introduced in [2]. Each individual $P_{u,v}$ in the population is a path from u to v, where u and v are two distinct vertices of the given input graph. The fitness of $P_{u,v}$ is given by $w(P_{u,v})$. Our algorithm starts with a population $\mathcal{P} := \{P_{u,v} = (u,v) | (u,v) \in E\}$ of size $|E|$ which contains all paths corresponding to the edges of the given graph G. In each iteration a single new individual is produced either by crossover or mutation depending on the pre-defined probability p_c. For all investigations in this paper, we assume that p_c is chosen as an arbitrary constant, i. e. $p_c \in]0,1[$ and $p_c = \Omega(1)$.

The mutation operator takes an individual $P_{x,y}$ from the population and applies sequentially $S+1$ local operations. Here, S is a parameter that is chosen according to the Poisson distribution (cf. [13]) with parameter $\lambda = 1$. In a local operation, the current path is either lengthened or shortened by a single edge. Assume that the current individual represents a path $P_{x,y} = (x = v_0, v_1, \ldots v_{\ell-1}, y = v_\ell)$ from x to y of length ℓ, i. e. consisting of ℓ edges. We denote by $E^-(v)$ and $E^+(v)$ the set of incoming and outgoing edges of a vertex v in G and choose an edge $e = (u,v) \in E^-(x) \cup E^+(y) \cup \{(x,v_1),(v_{\ell-1},y)\}$ uniformly at random from union of the sets. If $e \in \{(x,v_1),(v_{\ell-1},y)\}$, the edge is removed, otherwise the edge $e \in (E^-(x) \cup E^+(y)) \setminus \{(x,v_1),(v_{\ell-1},y)\}$ is added.

The crossover operator combines two individuals P_1, P_2 consisting of ℓ_1 and ℓ_2 edges by appending P_2 to P_1 which results in an individual of length $\ell_1 +$

```
1  $\mathcal{P} = \{P_{u,v} = (u,v) \mid (u,v) \in E\}$;
2  while true do
3      Choose $q \in [0,1]$ uniformly at random;
4      if $q \le p_c$ then
5          choose two individuals $P_{x,y}$ and $P_{x',y'}$ from $\mathcal{P}$;
6          perform crossover on $P_{x,y}$ and $P_{x',y'}$ to obtain an individual $P'_{s,t}$ ;
7      else
8          choose one individual $P_{x,y}$ uniformly at random from $\mathcal{P}$ and mutate
           $P_{x,y}$ to obtain an individual $P'_{s,t}$;
9      if ($P'_{s,t}$ is a path from $s$ to $t$) and (there is no $P''_{s,t} \in \mathcal{P}$ with
       $w(P''_{s,t}) \le w(P'_{s,t})$ and $w(P''_{s,t}) \ne w(P'_{s,t})$) then
10         exclude all $P''_{s,t}$ where $b_r(P'_{s,t}) \le b_r(P''_{s,t})$ and add $P'_{s,t}$ to $\mathcal{P}$
```

Algorithm 1. DEMO(r) (Diversity Evolutionary Multi-objective Optimizer)

ℓ_2. Contrary to the mutation operator a new individual created by a crossover operation may no longer represent a path and thus constitute an invalid solution which is rejected by the algorithm.

Finally the environmental selection of Line 9 and 10 includes such a new individual in the population iff it is not dominated by some other s-t-path. Inserting the new individual in the population all other s-t-paths residing in the same hyperbox are removed.

In this paper, we study the runtime of DEMO until it has produced an $(1+\varepsilon)$-approximation of the multi-criteria APSP problem. The runtime is measured by the number of iterations (of the while loop) until the algorithm has achieved the desired approximation. To analyze the runtime behavior of our algorithm, we make use of the following upper bound on the population size of DEMO which is a consequence of Theorem 2 given in [10].

Lemma 1. *Let $r > 1$ be the input of DEMO(r), $w_{\max}$ be the maximum weight of the weight function w, and $\mathcal{P}_{\max}$ be the maximal population size. Then*

$$\mathcal{P}_{\max} \le n(n-1) \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(r)}\right)^{d-1} .$$

We will show improved runtime bounds for $d \in \{2,3\}$. In principle our analysis can be extended to larger dimensions. However, for $d > 3$ they would not reveal an asymptotic speed up in comparison to using n times the multi-criteria SSSP approach of [8].

3 Analysis

In this section we show improved worst-case bounds on the runtime of our algorithm compared to the SSSP problem approach of [8]. The results for the single-criteria APSP problem [3] are included in our generalized bounds as a special case.

We adapt the concept of Pareto-optimality and box-domination with respect to the length k of a path, whereas the length of a path is measured by the number of its edges. With Lemma 3 we show how much time is needed to evolve a population approximating the Pareto-front of paths of length at most k into a population approximating the Pareto-front of paths of length at most $k + c$. This result is based on mutation only and is needed in Lemma 4 to prove how crossover and mutation work together. To show the interaction between crossover and mutation we make use of the the gap concept introduced in [3]. The final proof of our main theorem uses this lemma to show how a stage-wise increase of the length of the paths in the population constructs an approximation of the Pareto-front with a well controlled approximation factor.

In the following, we assume $2 \leq d \leq 3$. Our results can be extended to larger dimensions but for larger dimensions we would only get results that are comparable to running n times the approach of [8]. Our results are summarized in the following main theorem.

Theorem 1. *Let $r > 1$ be the input parameter of DEMO, and let $g := \left(\frac{\mathcal{P}_{\max}\log(n)}{n}\right)^{1/4}$. Then w. h. p. DEMO($r$) computes an $r^{3\cdot g\cdot\log(n)}$-approximation for the multi-criteria APSP problem in time*

$$O(n \cdot \mathcal{P}_{\max} \cdot g).$$

We remark that by the term w. h. p. (with high probability) we understand a result that holds with probability at least $(1-O(n^{-c}))$ for some $c > 0$ independent of n. The bound given in Theorem 1 depends on the maximal population size $\mathcal{P}_{\max}$ that DEMO might encounter. Furthermore, the bound on $\mathcal{P}_{\max}$ given in Lemma 1 depends on the number of given objectives as well as on the maximal weight $w_{\max}$ and the approximation guarantee $\log(r)$ both of which are required to be polynomially bounded in n.

We discuss the interesting cases for the value of d. Setting $d := 1$ we get the well-known tight bounds of $\Theta\big(n^{3.25}\big)$ on the optimization time for the single-criteria case as has been proven in [3]. Note, that no approximation is needed in this case.

To achieve an $(1+\varepsilon)$-approximation for the multi-criteria APSP problem, the parameter r needs to be tuned accordingly. Aiming at an $(1+\varepsilon)$-approximation, we may choose $r = (1+\varepsilon)^{\frac{1}{3\cdot g\cdot\log(n)}}$. Then for $d = 2$ the corresponding worst-case bound is

$$O\left(n^{11/3} \cdot (\log(n))^{\frac{7}{3}} \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)^{\frac{5}{3}}\right),$$

and for $d = 3$ it becomes

$$O\left(n^{4.5} \cdot (\log(n))^{6.5} \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)^{5}\right).$$

The multi-criteria APSP problem can be tackled by solving for each vertex of the given input graph the multi-criteria single-source-shortest path (SSSP) problem.

We compare our bound for dimension $d = 2$ and dimension $d = 3$ to n times the mutation-only worst-case bound of $O\big(n^2 \cdot \mathcal{P}_{\max}\big)$ for the multi-criteria SSSP of [8] with a maximal population of size

$$\mathcal{P}_{\max} \leq (n-1)\left(\frac{\log(n \cdot w_{\max})}{\log(r)}\right)^{d-1} \text{ and } r = (1+\varepsilon)^{1/(n-1)}.$$

Due to [8], the worst-case bound for $d = 2$ is

$$O\left(n^5 \cdot \frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)$$

Hence, our approach yields an asymptotic improvement of a factor of

$$n^{4/3} \cdot \log(n)^{-7/3} \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)^{-5/3}.$$

For dimension $d = 3$, the bound given in [8] is

$$O\left(n^6 \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)^2\right)$$

and our approach gives an asymptotic improvement of a factor of

$$n^{3/2} \cdot \log(n)^{-13/2} \cdot \left(\frac{\log(n \cdot w_{\max})}{\log(1+\varepsilon)}\right)^{-3}.$$

For the analysis of DEMO we need to consider a slightly adapted notion of Pareto-optimality which also takes into account the length of the path. Among all paths of length at most k we define an s-t-path $\hat{P}^k_{s,t}$ to be Pareto-optimal with respect to length k iff there is no other path $\tilde{P}^k_{s,t}$ of length at most k for which $w(\tilde{P}^k_{s,t}) < w(\hat{P}^k_{s,t})$ holds.

We denote by $\mathbb{P}_k$ the set of Pareto-optimal paths of length at most k, i. e. $\mathbb{P}_k = \{P_{u,v} \mid (u,v) \in V^2 \wedge |P_{u,v}| \leq k \wedge \nexists \tilde{P}_{u,v} \colon |\tilde{P}_{u,v}| \leq k \wedge w(\tilde{P}_{u,v}) < w(P_{u,v})\}$. Mapping the objective vectors of the set $\mathbb{P}_k$ into the set of hyperboxes $\mathbb{B}_k := b_r(w(\mathbb{P}_k))$, we are able to relate the notion of Pareto-optimality to the individuals kept in the population. We call a hyperbox $B \in \mathbb{B}_k$ dominated with respect to a pair of vertices $(u,v) \in V^2$ iff there is an individual $P_{u,v} \in \mathcal{P}$ such that $b_r(w(P_{u,v})) < B$. With the concept of box-domination we keep the size of the population as small as claimed in Lemma 1 because we only keep individuals from non-dominated hyperboxes in the population. Furthermore, for each non-dominated hyperbox and every s-t-path, $s, t \in V$, we have at most one individual. At the same time we are able to control the approximation-factor of the individuals in the population.

We need to make our arguments concerning the approximation precise. Assume that for every Pareto-optimal path

$$\hat{P}_{s,t} = (s = v_0, v_1, \ldots, v_k = v, v_{k+1} = t) \in \mathbb{P}_{k+1}$$

there is an individual $P_{s,v}$ in our population that r^i-approximates $\hat{P}_{s,v} \in \mathbb{P}_k$. Then it is clear that there is a mutation operation which appends the edge (v,t) to $P_{s,v}$. This increases the length of $P_{s,v}$ by one and results in an individual $P_{s,t}$ that r^i-approximates $\hat{P}_{s,t}$. During the remaining process of the optimization it might happen that we replace the path $P_{s,t}$ by some path $P'_{s,t}$ that is not dominated by $P_{s,t}$ and resides in the same hyperbox. In such a situation the approximation factor may increase from r^i to at most r^{i+1} while at the same time $P'_{s,t}$ also r^{i+1}-approximates all Pareto-optimal paths from s to t mapped to the hyperbox $b_r(w(\hat{P}_{s,t}))$ (see Lemma 2).

We extend the notion of approximation to sets in the following way: By $w(\mathcal{P}) \leq r^i \cdot w(\hat{P}_{s,t})$ we say that there exists a path $P_{s,t} \in \mathcal{P}$ such that $w(P_{s,t}) \leq r^i \cdot w(\hat{P}_{s,t})$ holds. And in the same fashion $w(\mathcal{P}) \leq r^i \cdot w(\mathbb{P}_k)$ denotes that for every $\hat{P}_{s,t} \in \mathbb{P}_k$ we have $w(\mathcal{P}) \leq r^i \cdot w(\hat{P}_{s.t})$.

Lemma 2 (Due to Lemma 1 in [8]). *Let $\hat{P}_{s,t}$ be an arbitrary path of the search space $\mathcal{S}$. Assume that there is a path $P_{s,t} \in \mathcal{P}$ in the current population with $w(P_{s,t}) \leq r^i \cdot w(\hat{P}_{s,t})$. Then all subsequent populations $\mathcal{P}'$ of DEMO(r) with $r > 1$ fulfill*

$$w(\mathcal{P}') \leq r^{i+1} \cdot w(P'_{s,t})$$

for all $P'_{s,t} \in \mathcal{S}$ with $b_r(w(P'_{s,t})) \leq b_r(w(\hat{P}_{s,t}))$.

With the result from [3, Lemma 2] we are able to analyze the time and probability of how to evolve a population $\mathcal{P}$ based on individuals that r^i-approximate all paths from $\mathbb{P}_k$, i. e. $w(\mathcal{P}) \leq r^i \cdot w(\mathbb{P}_k)$, into a population $\mathcal{P}'$ that r^{i+c}-approximates all paths from $\mathbb{P}_{k+c}$, i. e. $w(\mathcal{P}') \leq (r^{i+c}) \cdot w(\mathbb{P}_{k+c})$.

Lemma 3 (Analysis of Mutation). *Let $w(\mathcal{P}) \leq r^i \cdot w(\mathbb{P}_k)$ and $c \cdot \lambda \geq 12 \cdot \ln(n)$. Then, DEMO(r) computes w. h. p. in time $O(c \cdot \lambda \cdot n \cdot \mathcal{P}_{\max})$ an r^{i+c}-approximating individual for every Pareto-optimal path $\mathbb{P}_{k+c}$.*

One of the obvious problems when theoretically analyzing crossover algorithms is the proof of a good success probability. Often there are not that many individuals that can be combined to obtain a good offspring which may decrease the usefulness of crossover. To analyze the crossover operator, we use the gap concept introduced in [3]. This leads to an improved worst-case bound on the optimization time of evolutionary algorithms for the multi-criteria APSP problem. For a path $P_{u,v} = (u = u_0, u_1, \ldots, u_\ell = v)$ and an arbitrary sub-path $P_{u_i,u_j} = (u_i, u_{i+1}, \ldots, u_j)$ with $0 \leq i \leq j \leq \ell$, we call the integer $g := i + \ell - j$ the *gap* of the path P_{u_i,u_j} (w.r.t. $P_{u,v}$). We also call P_{u_i,u_j} a gap-path of $P_{u,v}$.

The key observation is that it suffices that crossover finds a gap-path that is sufficiently close to a sought-after path, because mutation is fast enough to fill in the edges of the gap. Utilizing the ideas introduced in [3] to our framework we can prove a sufficiently good success probability for a single crossover step that produces a gap-path. Assume that our current population $\mathcal{P}$ r^i-approximates all Pareto-optimal paths from the set $\mathbb{P}_k$, i. e. $w(\mathcal{P}) \leq r^i \cdot w(\mathbb{P}_k)$. Consider a Pareto-optimal path $\hat{P}_{u,v} \in \mathbb{P}_{\frac{3k}{2}}$ and a gap-path $\hat{P}_{u',v'}$ of $\hat{P}_{u,v}$ having gap

$g \leq n$. Then DEMO constructs a path $P_{u',v'}$ that r^i-approximates the gap-path $\hat{P}_{u',v'}$ with probability $\Omega(\frac{g^2 \cdot k}{\mathcal{P}_{\max}{}^2})$. In subsequent populations $\mathcal{P}'$ we might again loose an additional approximation factor r due to the selection in the algorithm such that we have $w(\mathcal{P}') \leq r^{i+1} \cdot w(\hat{P}_{u',v'})$. And $\mathcal{P}'$ also r^{i+1}-approximates all Pareto-optimal paths from the hyperbox $b_r(w(\hat{P}_{u',v'}))$.

Combining the analysis of the mutation operator (cf. Lemma 3) and the crossover operator employing the gap concept, we obtain the following key lemma. It shows the interplay between the two operators from the point on when we have an r^i-approximating individual in $\mathcal{P}$ for every Pareto-optimal $\mathbb{P}_{k_0}$ with

$$k_0 = \left(\frac{\mathcal{P}_{\max} \log(n)}{n}\right)^{1/4}.$$

The Lemma also shows, why it is not possible to have presumably better upper bounds for dimensions beyond $d > 3$ with crossover. The problem here is, that k_0 as well as the gap g are naturally upper bounded by $n-1$, the maximum length of a path.

Lemma 4. *Let* $k_0 := g := \left(\frac{\mathcal{P}_{\max} \log(n)}{n}\right)^{1/4}$, $k := (1.5)^i \cdot k_0$, *and* $w(\mathcal{P}) \leq r^j \cdot w(\mathbb{P}_k)$. *Then DEMO produces a population* $\mathcal{P}'$ *with* $w(\mathcal{P}') \leq r^{j+1+g} \cdot w(\mathbb{P}_{1.5k})$ *in time* $O\big((1.5)^{-i} \cdot n \cdot \mathcal{P}_{\max} \cdot g\big)$ *w. h. p.*

For the proof we first of all compute the time needed to get at least a path in our population which is an approximating path of a gap-path (towards a Pareto-optimal path). Interestingly this time is also sufficient to produce such a path for every path of the Pareto-set with respect to length $1.5k$. In the final step we show that again this time suffices to fill the remaining gap with mutation.

Proof. Fix an arbitrary Pareto-optimal path

$$\hat{P}_{u,v} = (u, u_1, \ldots, v) \in \mathbb{P}_{1.5k} \setminus \mathbb{P}_k.$$

The probability to get an arbitrary but fixed r^j-approximating gap-path of $\hat{P}_{u,v}$ with gap at most g in a crossover step is $\Omega\Big(\frac{g^2 k^2}{\mathcal{P}_{\max}{}^2}\Big)$. Note, that our success probabilities for crossover and mutation steps hold regardless of the steps before. Hence, we are allowed to treat the events considered here independently.

Consider a phase of length $t = O(n \cdot \mathcal{P}_{\max} \cdot g \cdot \Delta)$ with $\Delta := \frac{k_0}{k}$. Then the probability not to get any of the possible gap-paths of $\hat{P}_{u,v}$ within t iterations is

$$\begin{aligned}\left(1 - \Omega\left(\frac{g^2 \cdot k}{\mathcal{P}_{\max}{}^2}\right)\right)^t &\leq \exp\left(-\frac{g^2 \cdot k}{\mathcal{P}_{\max}{}^2} \cdot t\right) \\ &\leq \exp(-\Omega(\ln(n)))\end{aligned}$$

Time t also suffices to produce an r^j-approximating path for at least one of the possible gap-paths for every $\hat{P} \in \mathbb{P}_{1.5k}$. Consider the set of hyperboxes $\mathbb{B}_k := b_r(w(\mathbb{P}_k))$ as defined above with a size of at most $|\mathbb{B}_k| \leq \mathcal{P}_{\max}$. And fix for each $B \in \mathbb{B}_k$ a path $\hat{P}_{s,t}$ with $w(\hat{P}_{s,t}) = B$. Then we have already computed the failure probability that the algorithms does not derive at least one path $P'_{s',t'}$ which is a r^j-approximation of an arbitrary gap-path $\hat{P}_{s',t'}$ of $\hat{P}_{s,t}$ in time t. With the Union Bound Theorem cf. [1] we can thus bound the probability to get at least one r^j-approximating gap-path $P'_{s',t'}$ for every $\hat{P}_{s,t} \in \mathbb{P}_{1.5k}$ to

$$1 - \mathcal{P}_{\max} \cdot \exp(-\Omega(\log n)) \leq 1 - O\left(n^{-c'}\right),$$

for a constant c'. For subsequent populations $\mathcal{P}'$ and all paths from s' to t' mapped to $b_r(w(P'_{s',t'}))$ we have have again to account for a loss in the approximation factor, going over from r^j to r^{j+1} due to Lemma 2.

Assuming now that we have at least an r^{j+1}-approximating gap-path of an arbitrary but fixed (Pareto-optimal) $\hat{P}_{s,t}$ path with gap at most g in our population. Then DEMO can fill the gaps to gain an r^{j+1+g}-approximating path $P_{s,t}$ of path $\hat{P}_{s,t}$ (and all Pareto-optimal st-paths from the same hyperbox $b_r(w(\hat{P}_{s,t}))$) with mutation only. Assume, that the subpath $P_{s',t'}$ of $P_{s,t}$ has to be extended at both ends to get $P_{s,t}$. Then two phases of overall length $t = O(n \cdot \mathcal{P}_{\max} \cdot g \cdot \Delta)$ suffice to close the gap of $P_{s,t}$ with mutation. Without loss of generality in the first phase $P_{s',t'}$ is extended to $P_{s,t'}$, and in the second phase to $P_{s,t}$. Using Lemma 3 we set c to the value of the gap g and choosing λ in such a way, that $c \cdot \lambda \geq 12 \ln(n)$ holds. For every reasonable choice of λ (depending on c) we get a runtime bound of $O(n \cdot \mathcal{P}_{\max} \cdot g \cdot \Delta)$ which holds w. h. p. Time t does not suffice if our assumption to have a r^{j+1}-approximating gap-path in the population for every $\hat{P}_{s,t} \in \mathbb{P}_{1.5k}$ fails. However, the failure probability of $O\left(n^{-c'}\right)$ is very small. The claim now follows directly because the overall runtime does not exceed $O(n \cdot \mathcal{P}_{\max} \cdot g \cdot \Delta)$ with high probability. □

Based on our previous investigations which control the runtime and approximation error in the different steps, we are able to prove our main theorem.

Proof (of Theorem 1). The probability to apply the crossover or the mutation operator in an iteration is constant, and neither decreases the fitness of an individual. Hence, it is no problem to only regard the effect of one of the two in a starting phase. Let $k_0 := g$. Applying Lemma 3 with $c \cdot \lambda := k_0$, we see that after $O(n \cdot \mathcal{P}_{\max} \cdot k_0)$ iterations, w. h. p. we have a population $\mathcal{P}$ with the property $w(\mathcal{P}) \leq r^{k_0} \cdot \mathbb{P}_{k_0}$.

We now repeatedly apply Lemma 4. In time $O\left(n \cdot \mathcal{P}_{\max} \cdot g \cdot (1.5)^{-i}\right)$, with high probability we get a population $\mathcal{P}'$ with $w(\mathcal{P}') \leq r^{k_0 + i \cdot (g+1)} \cdot \mathbb{P}_{(1.5)^i k_0}$ out of a population $\mathcal{P}$ with $w(\mathcal{P}) \leq r^{k_0 + (i-1) \cdot (g+1)} \cdot \mathbb{P}_{(1.5)^{i-1} k_0}$. Hence the run-times form a geometric series and less than $\log_{1.5}(n)$ such stages suffice to find a population $\mathcal{P}$ with an approximation factor

$$r^{k_0 + \log_{1.5}(n)(g+1)} \leq r^{3 \cdot g \cdot \log(n)} =: \alpha$$

such that $w(\mathcal{P}) \leq \alpha \mathbb{P}_n$ in time $O(n \cdot \mathcal{P}_{\max} \cdot g)$. Since each stage works fine with high probability, our algorithm finds the desired approximation for the multi-criteria APSP problem with high probability as well. □

4 Conclusions

Understanding the usefulness of crossover is one of the major challenges in the theoretical analysis of evolutionary computation. In this paper, we have examined how crossover can speed up the computation for the multi-criteria all-pairs-shortest path problem. Due to the use of crossover, we were able to show runtime bounds that are significantly better than solving the multi-criteria single-source shortest path problem n times. Future research should further explore the usefulness of crossover for other combinatorial and multi-criteria problems.

References

1. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley, Chichester (1992)
2. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. In: Ryan, C., Keijzer, M. (eds.) GECCO, pp. 539–546. ACM, New York (2008)
3. Doerr, B., Theile, M.: Improved analysis methods for crossover-based algorithms. In: Rothlauf, F. (ed.) GECCO, pp. 247–254. ACM, New York (2009)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
5. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. 276, 51–81 (2002)
6. Ehrgott, M.: Multicriteria optimization, 2nd edn. Springer, Berlin (2005)
7. Eiben, A., Smith, J.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2007)
8. Horoba, C.: Analysis of a simple evolutionary algorithm for the multiobjective shortest path problem. In: 10th International Workshop on Foundations of Genetic Algorithms (FOGA 2009), pp. 113–120. ACM Press, New York (2009)
9. Jansen, T., Wegener, I.: Evolutionary algorithms - how to cope with plateaus of constant fitness and when to reject strings of the same fitness. IEEE Trans. Evolutionary Computation 5(6), 589–599 (2001)
10. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multiobjective optimization. Evolutionary Computation 10(3), 263–282 (2003)
11. Oliveto, P.S., He, J., Yao, X.: Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. International Journal of Automation and Computing 4(3), 281–293 (2007)
12. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Kovač (1997)
13. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. Journal of Mathematical Modelling and Algorithms, 349–366 (2004)

Path Relinking on Many-Objective NK-Landscapes

Joseph M. Pasia[1,2], Hernán Aguirre[1], and Kiyoshi Tanaka[1]

[1] Faculty of Engineering, Shinshu University
4-17-1 Wakasato, Nagano 380-8553, Japan
{ahernan,ktanaka}@shinshu-u.ac.jp

[2] Institute of Mathematics, University of the Philippines-Diliman
1101 Quezon City, Philippines
jmpasia@up.edu.ph

Abstract. Path relinking is a population-based heuristic that explores the trajectories in decision space between two elite solutions. It has been successfully used as a key component of several multi-objective optimizers, especially for solving bi-objective problems. In this paper, we focus on the behavior of pure path relinking, propose several variants of the path relinking that vary on their selection strategies, and analyze its performance using several many-objective NK-landscapes as instances. The study shows that the path relinking becomes more effective in improving the convergence of the algorithm as the number of objectives increases. It also shows that the selection strategy associated to path relinking plays an important role to emphasize either convergence or spread of the algorithm.

1 Introduction

Multi-objective optimization (MO) is the process of simultaneously finding solutions to two or more objectives. It is often called as *many*-objective optimization (MaO) if there are at least four objectives. MaO has attracted the interest of many researchers because of the poor performance of multi-objective evolutionary algorithms (MOEAs) that are known to be efficient in solving MO problems. Their poor performance is due to the large number of solutions in every Pareto front levels when the number of objectives is high [1], making their Pareto dominance ranking coarser, thus weakening their convergence property [2,3].

Most of the recent approaches that improve the performance of MOEAs in solving MaO problems introduce modifications that are based on either ranking improvement, dimensionality reduction, use of preference information, or use of indicator or scalarizing functions [3]. Recently, a strategy that searches on the subspaces obtained by partitioning the objective space has been studied [4]. However, all these approaches have focused mainly on the management of the objective space taking no or just slight consideration of the decision space.

In this paper, we study the behavior of path relinking (PR), a procedure that provides a unifying principle for combining elite solutions to create new ones based on generalized path constructions in *both* objective and decision spaces [5]. Although the efficacy of path relinking in solving MO problems has been demonstrated, it has not been used as a stand-alone algorithm but only as key component of different optimizers. Moreover, except for [6] where it considered four-objective knapsack problems, it has not been applied to solve complex MaO problems. Thus, we propose several approaches for implementing path relinking for complex combinatorial optimization problems having many objectives. We also investigate how the different selection strategies associated to PR can emphasize either convergence or spread of the algorithm

It is important to note that we do not aim to propose a pure PR as an alternative search procedure to MaO problems. Rather, we study the performance of PR using MNK-landscape models [2] to provide useful insights for practitioners on how to exploit the desirable properties of PR to enhance existing MOEAs.

2 Multi-objective Optimization and MNK-Landscapes

Multi-objective optimization involves simultaneously optimizing a set of two or more, and often conflicting, objective functions. In general, there are several efficient or nondominated solutions to MO problems. A solution $\boldsymbol{x}$ is nondominated if there exists no other feasible solution $\boldsymbol{y}$ such that $f_i(\boldsymbol{y}) \geq f_i(\boldsymbol{x})$[1], for $i = 1, 2, \ldots, M$ and $f_i(\boldsymbol{y}) > f_i(\boldsymbol{x})$ for some i, where f_i denote the individual objective functions and M is the number of objectives.

The MNK-landscape is an extension of Kauffman's NK-landscape models of epistatic interaction [7] to multi-objective *combinatorial* optimization problems. It is defined as a vector function mapping binary strings of length N into M real numbers $\boldsymbol{f} : \mathbb{Z}^N \rightarrow \mathbb{R}^M$, where $\mathbb{Z} = \{0, 1\}$. $\boldsymbol{K} = \{K_1, K_2, \ldots, K_M\}$ is a set of integers where each K_i gives the number of bits in the string that interact with each bit in the ith landscape. Each $f_i(\cdot)$ is expressed as

$$f_i(\boldsymbol{x}) = \frac{1}{N} \sum_{j=1}^{N} f_{i,j}(x_j, z_1^{(i,j)}, z_2^{(i,j)}, \ldots, z_{K_i}^{(i,j)}) \tag{1}$$

where $f_{i,j} : \mathbb{Z}^{K_i+1} \rightarrow \mathbb{R}$ gives the fitness contribution of x_j to $f_i(\cdot)$, and $z_1^{(i,j)}, z_2^{(i,j)}, \ldots, z_{K_i}^{(i,j)}$ are the K_i bits interacting with x_j in string $\boldsymbol{x}$.

3 General Concepts of Path Relinking

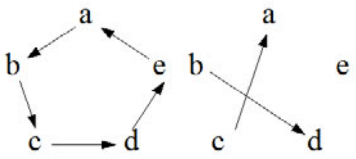
Path relinking generates a sequence of solutions in the decision space by exploring the trajectories that connect elite solutions. Starting from an initiating solution ($\boldsymbol{i_s}$), it creates new solutions that form a *path* by performing moves in the decision space that progressively incorporate the attributes (e.g edges) of the guiding solution ($\boldsymbol{g_s}$) [5]. In general, PR requires the following: (a) neighborhood structure for the moves, (b) solution attribute and its measure, (c) selection criteria

[1] Throughout the paper, we assume maximization of the objective functions.

```
P ←Generate();
repeat {/*iteration loop*/}
   I ←Define(P);
   S ← {};
   for all (i_s, g_s) ∈ I do
      repeat {/*Path Generation*/}
         F ←PathRelink(i_s, g_s, N, γ);
         x ←PathSelect(F,ω);
         S ← S ∪ {x};
         i_s ← x;
      until γ(i_s, g_s) < d_0
   end for
   P ← Nondominated(P ∪ S);
   P ←Select(P, l);
until termination condition is satisfied

return  nondominated set P
```

Fig. 1. Path relinking algorithm



(a) Cycle (b) Pair

Fig. 2. Concept diagram of selecting initiating and guiding solution via Cycle and Pair. The head of the arrow points to the guiding solution and the tail corresponds to the initiating solution

for $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$, and (d) selection criteria for the path. The neighborhood structure, solution attribute and its measure are usually problem dependent while $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ are characterized as high quality solutions. For the multi-objective case, these solutions are usually drawn from the set $\mathcal{P}$ of potentially efficient solutions (e.g. [8]). Since there are several paths that can be formed between any two solutions, the path selection strategy chooses the moves to realize. For example, one may use scalarizing function [8,9].

4 Path Relinking for Many-Objective Optimization

Like any other implementation of PR in multi-objective case, we perform PR between two solutions that belong to set $\mathcal{P}$. However, since we deal primarily with many-objective problems, we consider several ways of defining the initiating and guiding solutions only from the set of extreme solutions $\mathcal{P}' \subset \mathcal{P}$. Moreover, we use several forms of scalarizing functions to select new solutions to form the path. Whereas all applications perform local search procedures within PR to intensify the search towards the optimal Pareto front (e.g. [8]), we do not implement any such procedure in order to clearly reveal the behavior of the pure PR algorithm. Figure 1 provides the algorithmic framework for the PR used in this study.

4.1 Initial, Initiating and Guiding Solutions

In Fig. 1, the procedure `Generate` creates a randomly generated distinct extreme solutions that initially forms the set $\mathcal{P}$. The procedure `Define` initializes the set $\mathcal{I}$ of $\boldsymbol{i_s}$–$\boldsymbol{g_s}$ pairs in every iteration by first determining the set $\mathcal{P}'$. Then, it forms the $\boldsymbol{i_s}$–$\boldsymbol{g_s}$ pairs via two proposed methods, called *Cycle* and *Pair*, that differ in the way they sample the solutions. In every iteration, *Cycle* arranges the solutions of $\mathcal{P}'$ in random order. Then, the first and second solutions are labeled as the $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$, respectively. For the succeeding pairs of solutions, the initiating solution is the guiding solution of the previous pair and the guiding solution is the next solution in $\mathcal{P}'$. The final $\boldsymbol{i_s}$-$\boldsymbol{g_s}$ pair are the last and first solutions in $\mathcal{P}'$, respectively. The total number of pairs formed is $|\mathcal{P}'|$.

Pair forms a set of distinct pairs of $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ by iteratively drawing two distinct solutions from $\mathcal{P}'$, until it is no longer possible to obtain different pair of solutions from $\mathcal{P}'$. It forms a total of $\lfloor|\mathcal{P}'|/2\rfloor$ $\boldsymbol{i_s}$–$\boldsymbol{g_s}$ pairs and leaves one solution unmatched if $|\mathcal{P}'|$ is odd. Figure 2 illustrates the two methods when $M = 5$.

4.2 Path Generation and Selection

The actual generation of the sequence of solutions or *path* from solution $\boldsymbol{i_s}$ to $\boldsymbol{g_s}$ consists of two procedures. The first procedure `PathRelink`$(\boldsymbol{i_s}, \boldsymbol{g_s}, \mathcal{N}, \gamma)$ returns at each step the set F of 1-bit neighbor solutions of $\boldsymbol{i_s}$ that reduces the Hamming distance γ from $\boldsymbol{g_s}$, i.e. $F = \{\boldsymbol{x} \in \mathcal{N}(\boldsymbol{i_s}) : \gamma(\boldsymbol{x}, \boldsymbol{g_s}) < \gamma(\boldsymbol{i_s}, \boldsymbol{g_s})\}$. Since each call of `PathRelink` may return many solutions i.e. $|F| \geq 1$, then the second procedure `PathSelect` is used to choose the preferred path. `PathSelect` immediately chooses a single solution from F having the best value of the real-valued function ω expressed as the weighted sum fitness function by

$$\omega(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}) \qquad (2)$$

where $\boldsymbol{w}=[w_1, w_2, \ldots, w_M]$ is a weight vector such that $\sum_{i=1}^{M} w_i = 1$ and $w_i \geq 0\ \forall i$. The interaction between these procedures is illustrated in Fig. 3.

Initially, ω is defined as the objective function where $\boldsymbol{g_s}$ is best. Thus, this strategy clearly prefers moves that are attractive relative to f_i. Moreover, it limits the search from the many objective standpoint to single objective optimization. This is our natural way of extending the implementation of path selection in single-objective optimization problems [5] to many objective optimization. The solution selected by `PathSelect` then becomes an *intermediate* solution of the path. It is important to note that the two procedures have been expressed as a local search that optimizes a lexicographic objective function $\Phi = (\gamma, \omega)$ [9].

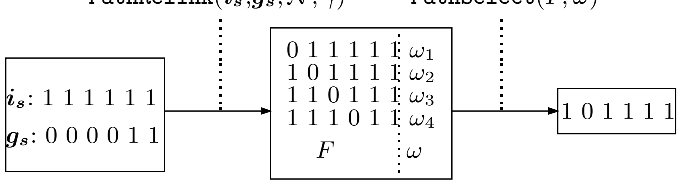


Fig. 3. Path relinking and selection. `PathRelink` generates several solutions and `PathSelects` chooses one solution to be a solution of the path

4.3 Link Direction and Archive

We consider re-initiating the process of path generation in the opposite direction by interchanging the roles of $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ to generate better extreme solutions. This has been implemented only in the bi-objective case in [6]. Combining the two strategies for setting the sets of $\boldsymbol{i_s}$–$\boldsymbol{g_s}$ pairs, and whether to interchange the roles of $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ initially give us four variants of the PR algorithm (see Table 1).

Since the number of nondominated solutions in every Pareto front increases dramatically with M [2], it is important to have an archiving strategy that controls the size (say, less than l) of the set $\mathcal{P}$. The method `Select` performs the archiving by selecting the M extreme solutions of $\mathcal{P}$ and randomly selecting solutions from the remaining $l - M$ solutions.

Table 1. Four variants of the path relinking algorithm. The symbol $\boldsymbol{i_s} \leftrightarrow \boldsymbol{g_s}$ ($\boldsymbol{i_s} \rightarrow \boldsymbol{g_s}$) indicates that there is (no) reversal of roles between initiating and guiding solutions.

Cycle	$\boldsymbol{i_s} \rightarrow \boldsymbol{g_s}$	PRCycle1
	$\boldsymbol{i_s} \leftrightarrow \boldsymbol{g_s}$	PRCycle2
Pair	$\boldsymbol{i_s} \rightarrow \boldsymbol{g_s}$	PRPair1
	$\boldsymbol{i_s} \leftrightarrow \boldsymbol{g_s}$	PRPair2

5 Experimental Results and Analysis

5.1 Metrics, Test Problems, and Parameters

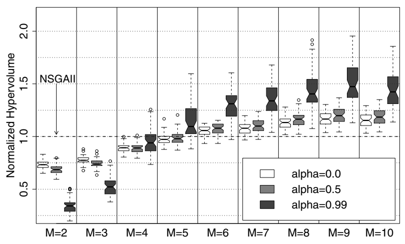
We evaluate the performance of PR algorithms using the hypervolume $\mathcal{H}$ and coverage $\mathcal{C}$ metrics [10], the sum of maximum objectives $\mathcal{S}_{\max}$ [3], and using the performance of conventional NSGA-II [11] as benchmark. The $\mathcal{H}$ metric uses several reference points $\mathcal{O}$ defined by the parameter α. If $\alpha = 0$ then $\mathcal{O}$ is the origin O and as α approaches 1, $\mathcal{O}$ approaches the point W having as coordinates the worst objective values of the solutions found. If $\alpha = 0.5$ then $\mathcal{O}$ is the midpoint of the segment $\overline{OW}$. $\mathcal{C}(A, B)$ gives the proportion of set B that is weakly dominated by set A. $\mathcal{S}_{\max}$ measures the convergence at the extremes and around the M edges of the Pareto front. It is equal to $\sum_{i=1}^{M} \max_{x \in \mathcal{P}} f_i(\boldsymbol{x})$.

We test the PR algorithms using MNK-landscapes with $2 \leq M \leq 10$ objectives, $N = 100$ bits, and $K_i = K = \{0, 1, 3, 5, 7, 10, 15, 25, 35, 50\}, i = 1, 2, \ldots, M$ epistatic interactions. Each M, N, K combination has 50 different landscapes. NSGA-II uses 100 individuals, 2-pt crossover with 0.6 recombination rate, and $1/N$ per bit mutation rate. All algorithms run once per landscape, use an archive size l of 100, and terminate after performing 3×10^5 function evaluations.

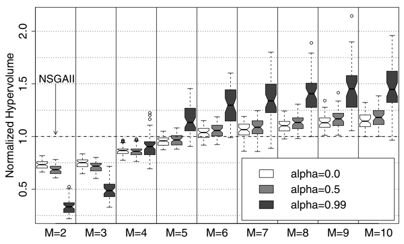
5.2 Performance Varying the Number of Objectives and Selection

We analyze the performances of the PR variants for $M = 2, 3, \ldots, 10$ and $K = 7$. It is known that high values of M translates to fewer but denser Pareto fronts [2]. We can see from Fig. 4(a)–(d), which gives the normalized $\mathcal{H}$ values $\mathcal{H}(\mathrm{PR})/\mathcal{H}(\mathrm{NSGA\text{-}II})$, that NSGA-II outperforms the PR variants when $2 \leq M \leq 4$, and there is a decrease in $\mathcal{H}$ values when α increases to 0.99 and $2 \leq M \leq 3$. Likewise, the $\mathcal{C}$ metric in Fig. 4(e) shows that NSGA-II weakly dominates all solutions of PR when $M = 2$ and covers most solutions when $M = 3$.

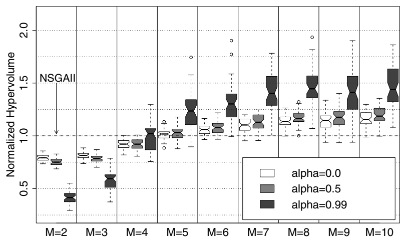
As M increases from 4 to 10, the convergence of PR variants improves. For example, at least 75% of the runs of PR show improvement in $\mathcal{H}$ (i.e. normalized $\mathcal{H} > 1$) for all values of α when $M \geq 6$. Also, although they just weakly dominate



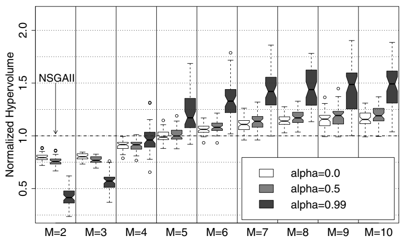
(a) $\mathcal{H}$: PRCycle1



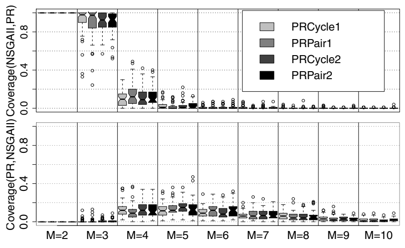
(b) $\mathcal{H}$: PRPair1



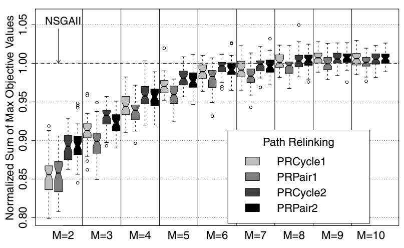
(c) $\mathcal{H}$: PRCycle2



(d) $\mathcal{H}$: PRPair2



(e) $\mathcal{C}$ metric



(f) $\mathcal{S}_{\max}$ metric

Fig. 4. (a)–(d)Normalized $\mathcal{H}$ metric (e) $\mathcal{C}$ metric (f) normalized $\mathcal{S}_{\max}$ metric between PR and NSGA-II for different M and $K = 7$

few of the solutions of NSGA-II, almost none of their solutions are covered by NSGA-II. Likewise, the big difference in the performance in $\mathcal{H}$ between $\alpha = 0.5$ and $\alpha = 0.99$ can be attributed to the better convergence in the central region.

Among the different PR variants, it can be seen that reversing the roles of $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ is beneficial in terms of improving the extreme solutions for $M \leq 5$. Figure 4(f) shows that for $M \leq 5$, the median of the normalized $\mathcal{S}_{\max}$ values of PRCycle2 and PRPair2 are significantly better than that of PRCycle1 and PRPair1. Likewise, *Cycle* shows significant edge over *Pair* only in the $\mathcal{S}_{\max}$ metric and only between PRCycle1 and PRPair1. This edge is insignificant when the re-initiating strategy is implemented. This suggests that reversing the roles of $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ is more effective in improving the quality of the extreme solutions.

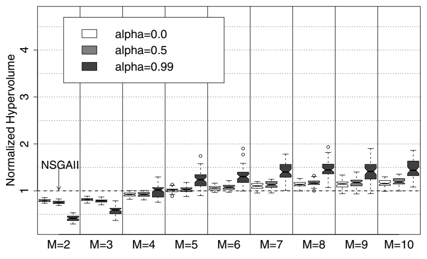
In terms of convergence, there is no strong indication that one variant is better than the others. This suggests that the manner of defining $\boldsymbol{i_s}$ and $\boldsymbol{g_s}$ from the set of extreme solutions and the re-initiating strategy do not have strong influence in the convergence property of the path relinking.

We also study three other ways of defining the components w_i of $\boldsymbol{w}$ (see Table 2). First, we let $w_i = 1$ if and only if $\boldsymbol{i_s}$ is best in f_i. Next, w_i is set to 0.5 if and only if $\boldsymbol{i_s}$ or $\boldsymbol{g_s}$ is best in f_i, and 0 otherwise. This method performs the search by using two objective functions each time. It is biased towards the central portion between the two functions. The final method targets the central region of the Pareto front by using Eq. 2 that aggregates (aggr) all the objective functions. The values of the weights are changed for every call of `PathSelect`.

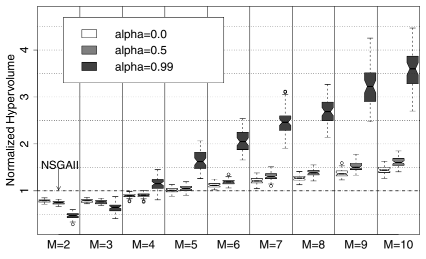
Table 2. Selection strategies for the path relinking algorithm using PRCycle2 variant

Fitness Function	Path relinking
$\omega(\boldsymbol{x}) = f_i(\boldsymbol{x})$	PRCycle2_w1.0
$\omega(\boldsymbol{x}) = 0.5 f_i(\boldsymbol{x}) + 0.5 f_j(\boldsymbol{x})$	PRCycle2_w0.5
$\omega(\boldsymbol{x}) = \sum_{i=1}^{M} w_i f_i(\boldsymbol{x})$	PRCycle2_aggr

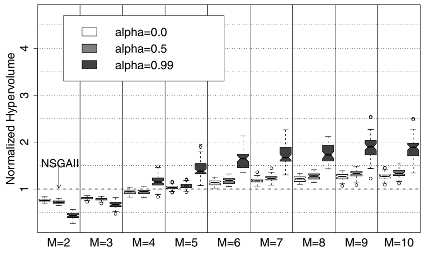
It can be seen in Fig. 5(a)-(d) that the normalized $\mathcal{H}$ for the different selection strategies improves as M increases. Remarkably, PRCycle2_aggr posted the biggest improvement. For example when $\alpha = 0.99$, the median $\mathcal{H}$ value of PRCycle2_aggr is almost 100% greater than that of the NSGA-II if $M = 6$. Also, roughly between 40% to 60% of the solutions of NSGA-II are covered by PRCycle2_aggr when $M \geq 6$, while NSGA-II covers nothing of PRCycle2_aggr (see Fig. 5(e)). The good convergence of PRCycle2_aggr expectedly sacrifices the quality of its extreme solutions since the $\mathcal{S}_{\max}$ metric (see Fig. 5(f)) shows that PRCycle2_aggr is totally outperformed by NSGA-II. It is PRCycle2 and PRCycle2_w1.0 that perform well in terms of $\mathcal{S}_{\max}$ with the latter obtaining the best extreme values. PRCycle_w1.0 even outperforms NSGA-II when $M \geq 6$. All these results suggest that the manner of selecting the intermediate solutions or creating the path is a valuable factor when implementing PR. The different selection strategies exhibit a trade-off between convergence and spread. Thus, it will



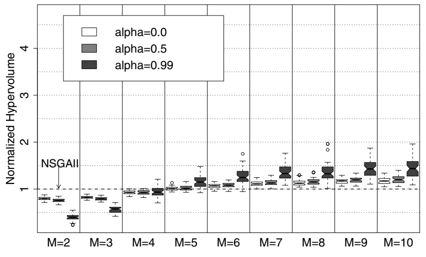
(a) $\mathcal{H}$: PRCycle2



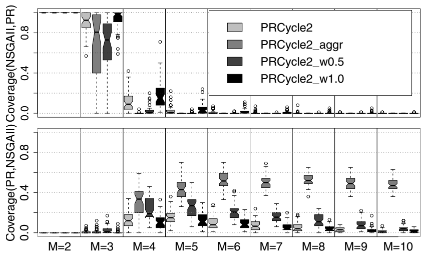
(b) $\mathcal{H}$: PRCycle2_aggr



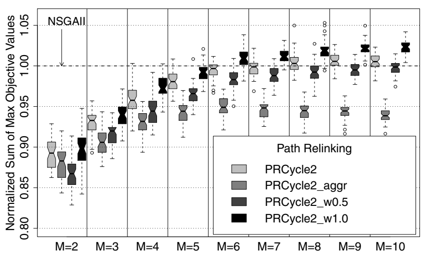
(c) $\mathcal{H}$: PRCycle2_w0.5



(d) $\mathcal{H}$: PRCycle2_w1.0



(e) $\mathcal{C}$ metric



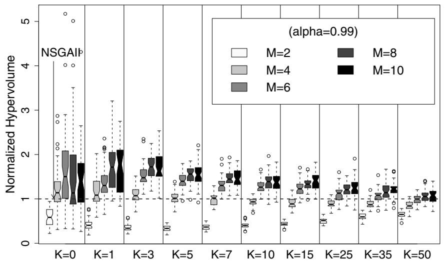
(f) $\mathcal{S}_{\max}$ metric

Fig. 5. (a)–(d) Normalized $\mathcal{H}$ metric (e) $\mathcal{C}$ metric (f) normalized $\mathcal{S}_{\max}$ metric between PR and NSGA-II for different M and $K = 7$

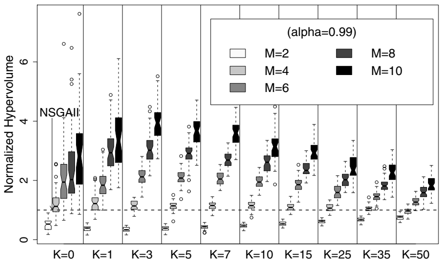
be interesting in the future to investigate adaptive strategies that simultaneously improve both convergence and spread.

5.3 Performance Varying the Levels of Epistastic Interactions

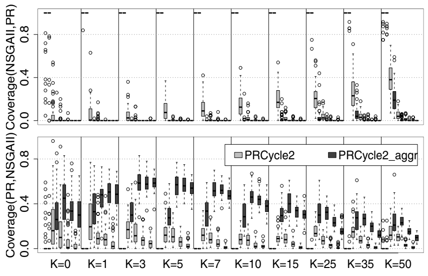
It was demonstrated in [2] that the complexity of the MNK-landscape models increases with K since the optimal solutions and their true Pareto fronts become



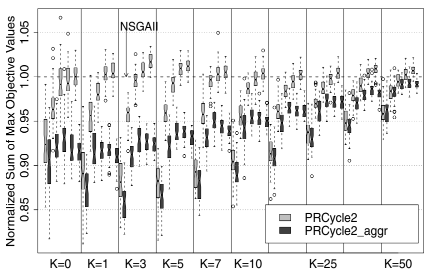
(a) $\mathcal{H}$: PRCycle2



(b) $\mathcal{H}$: PRCycle2_aggr



(c) $\mathcal{C}$ metric



(d) $\mathcal{S}_{\max}$ metric

Fig. 6. (a)–(b) Normalized $\mathcal{H}$ metric (c) $\mathcal{C}$ metric (d) normalized $\mathcal{S}_{\max}$ metric between PR and NSGA-II for different values of K and for $M = 2, 4, 6, 8, 10$

more "discontiguous", and more non-convex regions appear in the fronts. To study the effects of varying K, we analyze the performances of PRCycle2 and PRCycle2_aggr when K ranges from 0 to 50 and under different M values. It can be observed in Fig. 6(a)–(b) that for all values of K, the $\mathcal{H}$ values of PR are better than NSGA-II only when M is high. However, the edge of PR over NSGA-II diminishes as K increases.

Figure 6(c) shows that NSGA-II covers almost all the solutions of PR for all K and $M = 2$. But, PRCycle2 and PRCycle2_aggr weakly dominated more solutions of NSGA-II than NSGA-II can cover them when $M > 4$. PRCycle2_aggr also has higher coverage of NSGA-II compared to PRCycle2. However, the $\mathcal{C}$ values of NSGA-II by PRCycle2_aggr decreases as K increases from 3 to 50. For example when $K = 3$, the average coverages of NSGA-II by PRCycle2_aggr are 32.16% and 58.16% for $M = 4$ and 8, respectively. These values drop to 9.78% and 24.1%, respectively, when K increases to 50. Figure 6(d) suggests that PRCycle2 obtains better extreme solutions than PRCycle2_aggr but both don't find extreme solutions that are as good as NSGA-II. However, as K increases, there is an improving trend for the normalized $\mathcal{S}_{\max}$ of PRCycle2_aggr.

6 Conclusions

In this paper, we study the performance of PR algorithm on different MNK-landscape models. We design several variants of path relinking that differ on the way the initiating and guiding solutions are defined, and on whether to interchange their roles or not. We also study how the selection of the path using several fitness functions affects the performance of PR. Experiments show that the choice of the initiating and guiding solutions have a slight effect on the convergence around the edges of the Pareto front while the selection of path has strong influence on the convergence in the central region. In fact, PR has a stronger convergence property around the central region compared to NSGA-II when $M \geq 4$. This good convergence can be seen in a broad range of levels of epistatic interaction K, with its peak improvement around $1 \leq K \leq 10$.

References

1. Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Evolutionary Multi-Criterion Optimization. LNCS, pp. 367–390. Springer, Heidelberg (2003)
2. Aguirre, H.E., Tanaka, K.: Working principles, behavior, and performance of MOEAs on MNK-landscapes. European Journal of Operational Research 181(3), 1670–1690 (2007)
3. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: Proc. of 2008 IEEE Congress on Evolutionary Computation, Hong Kong, pp. 2424–2431 (June 2008)
4. Aguirre, H., Tanaka, K.: Many-objective optimization by space partitioning and adaptive ε-ranking on MNK-Landscapes. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 407–422. Springer, Heidelberg (2009)
5. Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics 29(3), 653–684 (2000)
6. Beausoleil, R.P., Baldoquin, G., Montejo, R.: Multi-start and path relinking methods to deal with multiobjective knapsack problems. Annals of Operations Research 157(1), 105–133 (2008)
7. Kauffman, S.A.: The Origins of Order: Self-Organization and Selection in Evolution. Oxford University Press, Oxford (1993)
8. Pasia, J., Gandibleux, X., Doerner, K., Hartl, R.: Local search guided by path relinking and heuristic bounds. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 501–515. Springer, Heidelberg (2007)
9. Jaszkiewicz, A., Zielniewicz, P.: Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. European Journal of Operational Research 193(3), 885–890 (2009)
10. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Trans. Evolutionary Computation 7, 117–132 (2003)
11. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation:NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)

In Search of Equitable Solutions Using Multi-objective Evolutionary Algorithms

Pradyumn Kumar Shukla, Christian Hirsch, Hartmut Schmeck

Institute AIFB
Karlsruhe Institute of Technology
Karlsruhe, D-76128, Germany
{pradyumn.shukla,c.hirsch,hartmut.schmeck}@kit.edu

Abstract. Over the last two decades, evolutionary algorithms have been applied in solving multi-objective optimization problems. Most of these algorithms use the concept of Pareto-optimality to drive their search. However, many real-world multi-objective applications, in particular from location theory and general resource allocation models, require finding so-called equitably efficient points. These solutions form a subset of the Pareto-optimal set. In equitable efficiency, objective functions are considered impartial which makes the distribution of outcomes more important rather than assignment of several outcomes to an objective. In literature, we found two classical approaches to compute an equitably efficient point. These approaches rely on either solving a problem which is always non-differentiable or on solving a more difficult problem. In this paper, for the first time, a multi-objective evolutionary approach to this problem is proposed. The approach finds a diverse set of equitably optimal solutions and, in addition, tackles the non-differentiability which is inherently present in the classical approach. It is shown that even for simple differentiable problems, which belong to the realm of classical techniques, the evolutionary approach is a better choice than the classical ones. Computational studies on a number of test problems of varying complexity demonstrate the efficiency of the evolutionary approach in solving a large class of both simple and complex equitable multi-objective optimization problems.

1 Introduction

Over the last two decades, evolutionary algorithms have been applied in solving multi-objective optimization problems. Most of these algorithms use the concept of Pareto-optimality to drive their search [1]. However, in many real-world applications the objective functions express ideas of allocation of resources and the corresponding multi-objective optimization problem is to achieve some equitable allocation of resources. These problems arise in general resource allocation models and location theory among others [2, 3]. Equitable optimality is a stronger notion than that of Pareto-optimality and equitably optimal points to a multi-objective optimization problem form a subset of the Pareto-optimal set. In equitable efficiency, objective functions are considered impartial/ anonymous and this emphasizes the distribution of outcomes rather than the assignment of several outcomes to an objective. This issue is detailed later in Section 2.

In literature, we found two classical approaches towards finding an equitably optimal point [2]. These approaches rely on either solving a problem which is *always non-differentiable* (even if the original problem is differentiable) or on solving a *more difficult* problem having a large number of variables and constraints. In this paper, for the first time, we propose a multi-objective evolutionary approach to find a diverse set of equitably optimal solutions. In addition to this, the evolutionary approach tackles the non-differentiability which is inherently present in the classical approach. It is shown that even for simple differentiable problems the evolutionary approach is a better choice than the classical ones. Computational studies on a number of test problems of varying complexity demonstrate the efficacy of the evolutionary approach in solving a large class of both simple and complex equitable multi-objective optimization problems.

This paper is divided into four sections of which this is the first. The next section presents concepts of equitable efficiency and an overview of solution approaches using classical generating methods. The same section also introduces a multi-objective evolutionary approach for finding equitably efficient points. Simulation results using the evolutionary approach and a comparison with some existing classical methods is presented in Section 3. Conclusions as well as extensions which emanated from this study are presented at the end of this contribution.

2 Equitable Efficiency and Solution Approaches

In this section, we present the notion of equitable efficiency and discuss possible ways to solve equitable multi-objective optimization problems.

2.1 Equitable Efficiency

For given objective functions $f_1, \dots, f_m : \mathbb{R}^n \to \mathbb{R}$ and a given $X \subseteq \mathbb{R}^n$, let us consider the following multi-objective optimization problem (MOP):

$$\min \mathbf{f}(\mathbf{x}) := (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})) \qquad \text{s.t. } \mathbf{x} \in X. \tag{1}$$

Let $\mathcal{I} := \{1, 2, \dots, m\}$ denote the index set and let $\mathcal{Y} := \mathbf{f}(X)$ denote the set of all feasible points in the objective space. A central optimality notion for the above problem is that of Pareto-optimality.

Definition 1 (Pareto-optimality). *A point* $\mathbf{x}^* \in X$ *is called* Pareto-optimal *if no* $\mathbf{x} \in X$ *exists so that* $f_i(\mathbf{x}) \le f_i(\mathbf{x}^*)$ *for all* $i \in \mathcal{I}$ *with strict inequality for at least one index* i. *If* $\mathbf{x}^*$ *is Pareto-optimal, then* $\mathbf{f}(\mathbf{x}^*)$ *is called an efficient point.*

The above notion of Pareto-optimality is related to a preference model. Corresponding to a preference model, denoted as $\preceq$, the corresponding relations of strict preference $\prec$ and indifference $\cong$ are defined as:

$$\begin{aligned} \mathbf{y}^1 \prec \mathbf{y}^2 &\iff (\mathbf{y}^1 \preceq \mathbf{y}^2 \text{ and not } \mathbf{y}^2 \preceq \mathbf{y}^1) \text{ and} \\ \mathbf{y}^1 \cong \mathbf{y}^2 &\iff (\mathbf{y}^1 \preceq \mathbf{y}^2 \text{ and } \mathbf{y}^2 \preceq \mathbf{y}^1), \end{aligned}$$

for all $\mathbf{y}^1, \mathbf{y}^2 \in \mathcal{Y}$. In order to study general preference relations, we introduce the following axioms.

(A1) $\preceq$ is *reflexive*, i.e., $\mathbf{y} \preceq \mathbf{y}$, $\forall \mathbf{y} \in \mathcal{Y}$.
(A2) $\preceq$ is *transitive*, i.e., $\mathbf{y}^1 \preceq \mathbf{y}^2$ and $\mathbf{y}^2 \preceq \mathbf{y}^3$ implies $\mathbf{y}^1 \preceq \mathbf{y}^3$, $\forall \mathbf{y} \in \mathcal{Y}$.
(A3) $\preceq$ is *strictly monotonic*, i.e., $\mathbf{y} - \varepsilon \mathbf{e}_i \prec \mathbf{y}$, $\forall \varepsilon > 0$, $\mathbf{y} \in \mathcal{Y}$, where $\mathbf{e}_i$ denotes the i-th unit vector in $\mathbb{R}^m$.

Preference models satisfying axioms (A1-A3) are known as rational preference models. The next two definitions from [2] characterize Pareto-domination and Pareto-optimality.

Definition 2 (Pareto-domination). *A point* $\mathbf{y}^* \in \mathcal{Y}$ *is said to (Pareto) dominate a point* $\mathbf{y} \in \mathcal{Y}$*, or* $\mathbf{y}$ *is (Pareto) dominated by* $\mathbf{y}^*$*, if and only if* $\mathbf{y}^* \prec \mathbf{y}$ *for* all rational preference models $\preceq$*.*

Definition 3 (Pareto-optimality). *A point* $\mathbf{x}^* \in X$ *is Pareto-optimal if and only if there does not exist an* $\mathbf{x} \in X$ *such that* $\mathbf{f}(\mathbf{x})$ *(Pareto) dominates* $\mathbf{f}(\mathbf{x}^*)$*. If* $\mathbf{x}^*$ *is Pareto-optimal, then* $\mathbf{f}(\mathbf{x}^*)$ *is called an efficient point.*

The characterization of Pareto-optimality in Definition 3 is an axiomatic characterization and is equivalent to Definition 1 (see details in [2]).

The notion of equitably optimal points comes from axioms (A1-A3) and two additional axioms, which are defined next.

(A4) $\preceq$ is *impartial*, i.e., $\forall \mathbf{y} := (y_1, \ldots, y_m) \in \mathcal{Y}$ and any permutation τ of $\mathcal{I}$, it holds that

$$\left(y_{\tau(1)}, y_{\tau(2)}, \ldots, y_{\tau(m)}\right) \cong (y_1, y_2, \ldots, y_m)\,.$$

(A5) $\preceq$ satisfies the *Pigou-Dalton principle of transfers*, i.e.,

$$y_i < y_j \Rightarrow \mathbf{y} - \varepsilon \mathbf{e}_i + \varepsilon \mathbf{e}_j \prec \mathbf{y} \quad \text{for } 0 < \varepsilon < y_j - y_i,\ i, j \in \mathcal{I}.$$

The Pigou-Dalton principle of transfers [4] states that a transfer of small amount from an objective to a relatively worse objective results in a more preferred vector (in the objective space). Preference models satisfying axioms (A1-A5) are known as equitable preference models.

Definition 4 (Equitable-domination). *A point* $\mathbf{y}^* \in \mathcal{Y}$ *is said to equitably dominate a point* $\mathbf{y} \in \mathcal{Y}$*, or* $\mathbf{y}$ *is equitably dominated by* $\mathbf{y}^*$*, if and only if* $\mathbf{y}^* \prec \mathbf{y}$ *for* all equitable preference models $\preceq$*.*

Definition 5 (Equitable-optimality). *A point* $\mathbf{x}^* \in X$ *is equitably optimal if and only if there does not exist an* $\mathbf{x} \in X$ *such that* $\mathbf{f}(\mathbf{x})$ *equitably dominates* $\mathbf{f}(\mathbf{x}^*)$*. If* $\mathbf{x}^*$ *is equitably optimal, then* $\mathbf{f}(\mathbf{x}^*)$ *is called an equitably efficient point.*

From Definitions 3 and 5 it is clear that an equitably optimal point is also Pareto-optimal. The goal of solving an equitable multi-objective problem (and the topic of this paper) is to find equitably optimal points.

2.2 Problem Formulation

Here, we provide further characterizations of equitably optimal points and discuss some solution approaches. Many of the results are from [2].

In order to mathematically formalize the preference model of equitable domination, we proceed as follows. Let the map $\Theta : \mathbb{R}^m \to \mathbb{R}^m$ be so that $\Theta(\mathbf{y}) = (\theta_1(\mathbf{y}), \theta_2(\mathbf{y}), \ldots, \theta_m(\mathbf{y}))$, where $\theta_1(\mathbf{y}) \geq \theta_2(\mathbf{y}) \geq \ldots \geq \theta_m(\mathbf{y})$ and there exists a permutation τ of the set $\mathcal{I}$ such that $\theta_i(\mathbf{y}) = y_{\tau(i)}$ for all $i \in \mathcal{I}$. Moreover, let $\Gamma : \mathbb{R}^m \to \mathbb{R}^m$ and $\mathbf{q} := (q_1, \ldots, q_m)$ be so that $\Gamma(\mathbf{q}) = (\gamma_1(\mathbf{q}), \gamma_2(\mathbf{q}), \ldots, \gamma_m(\mathbf{q}))$, where

$$\gamma_i(\mathbf{q}) = \sum_{j=1}^{i} q_j \quad \text{for all } i \in \mathcal{I}.$$

Composition of Θ and Γ gives us the *cumulative ordering map* $\bar{\Theta} = \left(\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_m\right)$ defined as $\bar{\Theta}(\mathbf{y}) = \Gamma(\Theta(\mathbf{y}))$, i.e.,

$$\bar{\theta}_i(\mathbf{y}) = \sum_{j=1}^{i} \theta_j(\mathbf{y}) \quad \text{for all } i \in \mathcal{I}.$$

It can be easily seen that if $\mathbf{y} = \mathbf{f}(\mathbf{x})$, the coefficients of the vector $\bar{\Theta}(\mathbf{f}(\mathbf{x}))$ represent the largest, the sum of the two largest, the sum of the three largest etc., objective functions values at $\mathbf{x}$, respectively. From [5, Theorem 1], we obtain that equitable domination is equivalent to Pareto-domination on the modified set of objectives $\bar{\Theta}$. This is an important result and is used to transform the goal of finding equitably optimal points of the original problem to that of finding Pareto-optimal points of the following multi-objective problem:

$$\min \bar{\Theta}(\mathbf{x}) := \left(\bar{\theta}_1(\mathbf{x}), \bar{\theta}_2(\mathbf{x}), \ldots, \bar{\theta}_m(\mathbf{x})\right) \qquad \text{s.t. } \mathbf{x} \in X. \tag{P1}$$

If we look carefully at (P1), we see that now all the first $m-1$ objective functions are *non-differentiable*. This non-differentiability is inherent in the Θ mapping and is there even if the original multi-objective problem (1) is differentiable. Hence, although we have now reduced the problem of finding equitably optimal points to the well-studied problem of finding Pareto-optimal points, this comes at the cost of non-differentiability. In order to have a smooth formulation, the authors of the original study reformulated the multi-objective problem (P1) as the following equivalent problem:

$$\begin{aligned} \min \quad & (z_1, z_2, \ldots, z_m) \\ \text{s.t.} \quad & \mathbf{x} \in X, \\ & z_k = kt_k + \sum_{i=1}^{m} d_{ik}^{+} \quad \text{for all } k \in \mathcal{I}, \\ & t_k + d_{ik}^{+} \geq f_i(\mathbf{x}),\ d_{ik}^{+} \geq 0 \quad \text{for all } i, k \in \mathcal{I}. \end{aligned} \tag{P2}$$

Although (P2) removes the problem of non-differentiability, this is done at the extra cost of inserting a large number of additional variables and additional constrains. Assuming that the original problem (1) has k constraints, Table 2 shows a comparison of the two formulations.

Table 1. Comparison between two formulations for finding equitably optimal points

	Formulation (P1)	Formulation (P2)
# objectives	m	m
# variables	n	$n+m+m^2$
# constraints	k	$k+2m^2$

2.3 Classical and Evolutionary Solution Approaches

Formulations (P1) and (P2) are multi-objective problems and could, in principle, be solved by any multi-objective algorithm. The main aim of this paper is to show that evolutionary algorithms are better suited for finding equitably optimal points, not only for difficult but also for simple problems. This is done by applying the NSGA-II algorithm [6] with the equitable definition of domination. We call this algorithm as eNSGA-II. We will see that the non-differentiability that is inherent in formulation (P1) or the additional large number of variables and constraints that are present in formulation (P2) make it very difficult for classical techniques. It will turn out that even for simple multi-objective problems, having all differentiable objectives, eNSGA-II gives better results than classical ones. An earlier study showed that for finding efficient points, classical methods perform better than NSGA-II for simple differentiable problems [7].

In addition to the eNSGA-II algorithm, we use the Normal Boundary Intersection (NBI) method [8] and a modified NBI (mNBI) method [9], on both the formulations (P1) and (P2) (we call these as NBI-P1, NBI-P2, mNBI-P1 and mNBI-P2, respectively). Both NBI and mNBI methods are used for finding uniformly spread Pareto-optimal solutions for a general nonlinear multi-objective optimization problem. These approaches use a scalarization scheme with the

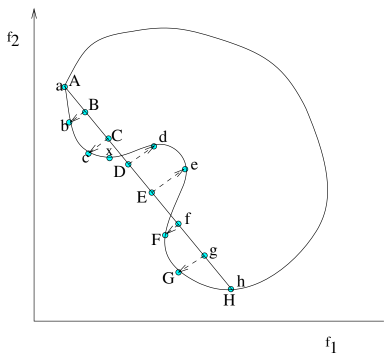


Fig. 1. Schematic showing the NBI method. Points a, b, c, d, e, f, g, h are found by solving the subproblems corresponding to points A, B, C, D, E, F, G, H

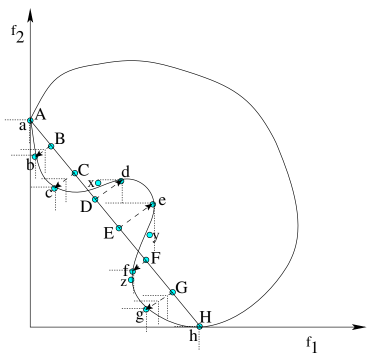


Fig. 2. Schematic showing the mNBI method. In contrast to NBI, the dominated points d, e, f are not found as they are dominated by feasible points x, y, z, respectively

property that a uniform spread in parameters will give rise to a near uniform spread in points on the efficient frontier. Figures 1 and 2 show a schematic of the NBI and mNBI methods, respectively. These methods work with the convex hull of the minima of the individual objective functions. An advantage of the mNBI method over the NBI method is that it guarantees Pareto-optimality. For further details, we refer the reader to the original studies [8, 9] or to the book [10].

3 Simulation Results

In this section, we present simulation results using eNSGA-II, NBI-P1, NBI-P2, mNBI-P1 and mNBI-P2 on a number of test problems of varying complexity. These include five from the ZDT suite (ZDT1, ZDT2, ZDT3, ZDT4, ZDT6) [6], four problems from the CEC-2007 competition (SZDT1, SZDT2, SZDT4, SZDT6), one from the DTLZ family (DTLZ5-3D) [1], and four from the WFG suite (WFG1, WFG8, with both 2 and 3 objectives) [1]. Note that the DTLZ5 and the ZDT problems are differentiable and uni-modal, except ZDT4 which is differentiable and multi-modal. The other problems are differentiable *almost everywhere* (the set of non-differentiable points is countable). In their equitable form (P1), all the problems are non-differentiable. For all problems solved using eNSGA-II, we use a population of size 100 and set the maximum number of function evaluations as 20,000 (200 generations). We use a standard real-parameter SBX and polynomial mutation operator with $\eta_c = 15$ and $\eta_m = 20$, respectively [6]. For the classical methods, we solve 100 sub-problems (analogous to the population size of 100 in eNSGA-II) and note the number of function evaluations.

As noted earlier, the equitable front is a part of the efficient front. For all problems we compute a well-distributed approximation of the equitable front (reference set) as follows. Corresponding to the problem, first we generate 10,000 well-diverse points on the efficient front. Then we find the equitable points, i.e., the points that are non-dominated in the $\bar{\Theta}$ space. In order to evaluate the results, we use the Inverted generational distance (IGD) and Generational distance (GD) metrics (wrt. the obtained reference set). For statistical evaluation we use the attainment surface based statistical metric [6]. We run each algorithm for 51 times and the median (50%) attainment surface (26^{st}) for eNSGA-II is plotted. In the classical methods, we use the SQP method for solving and we present the number of function evaluations needed for 100 sub-problems and the number of *failed* optimization runs. The source code of eNSGA-II, NBI and mNBI is made available[1]. The data files for all the 51 runs of all the problems are available on request.

Figure 3 illustrates the effect of the equitable transformation $\bar{\Theta}$ on the test problems ZDT1, ZDT2, ZDT3 and WFG1-2D. It can be seen that equitable front is only a small portion of the efficient front, except for ZDT2. From Figure 4 we see that all the algorithms are able to converge to the equitable front for ZDT1. However, mNBI-P2 and NBI-P2 find very few ($\sim$5) points on the equitable front. This has to do with the additional large number of variables that are introduced

[1] http://www.aifb.kit.edu/web/eNSGA-II/en

Table 2. The number of function evaluations needed for solving 100 sub-problems and the number of failed optimization runs using the NBI and the mNBI method

NBI Algorithm							
	ZDT1	**ZDT2**	**ZDT3**	**ZDT4**	**ZDT6**	**SZDT1**	**SZDT2**
# evals. P1	12893	12775	36423	11029	8390	25959	113268
# failed P1	0	0	27	0	80	99	96
# evals. P2	11769	12643	14010	30160	7848	287020	526100
# failed P2	0	0	0	1	38	81	60
	SZDT4	**SZDT6**	**DTLZ5**	**WFG1,2d**	**WFG1,3d**	**WFG8,2d**	**WFG8,3d**
# evals. P1	27089	15237	24335	76171	204096	65172	71207
# failed P1	100	100	100	97	1	2	53
# evals. P2	298046	202273	5848	30208	80881	255266	341487
# failed P2	0	14	0	98	91	40	18

mNBI Algorithm							
	ZDT1	**ZDT2**	**ZDT3**	**ZDT4**	**ZDT6**	**SZDT1**	**SZDT2**
# evals. P1	12893	12775	36423	11029	8390	25959	113268
# failed P1	0	0	27	0	80	99	96
# evals. P2	11769	12643	14010	30160	7848	287020	526100
# failed P2	0	0	0	1	38	81	60
	SZDT4	**SZDT6**	**DTLZ5**	**WFG1,2d**	**WFG1,3d**	**WFG8,2d**	**WFG8,3d**
# evals. P1	27089	15237	24335	76171	204096	65172	71207
# failed P1	100	100	100	97	1	2	53
# evals. P2	298046	202273	5848	30208	80881	255266	341487
# failed P2	0	14	0	98	91	40	18

Table 3. Generational distance (GD) and Inverted generational distance (IGD) metric values for the test problems, using the eNSGA-II algorithm

eNSGA-II							
GD	**ZDT1**	**ZDT2**	**ZDT3**	**ZDT4**	**ZDT6**	**SZDT1**	**SZDT2**
best	0.000036	0.000061	0.000229	0.000099	0.000334	0.000513	0.004345
worst	0.000055	0.000097	0.000291	0.003960	0.000606	0.001613	0.013401
mean	0.000040	0.000078	0.000265	0.000614	0.000459	0.000912	0.006865
std.	0.000003	0.000007	0.000012	0.000733	0.000057	0.000265	0.001650
GD	**SZDT4**	**SZDT6**	**DTLZ5**	**WFG1,2d**	**WFG1,3d**	**WFG8,2d**	**WFG8,3d**
best	0.436254	0.292219	0.000181	0.079634	0.117718	0.043129	0.056362
worst	3.369149	0.698726	0.000261	0.399920	0.710253	0.139375	0.153480
mean	1.303690	0.483888	0.000226	0.127480	0.390463	0.067919	0.087787
std.	0.644303	0.086988	0.000020	0.089739	0.188274	0.018488	0.020908

IGD	**ZDT1**	**ZDT2**	**ZDT3**	**ZDT4**	**ZDT6**	**SZDT1**	**SZDT2**
best	0.000055	0.000136	0.000171	0.000113	0.000068	0.000509	0.029495
worst	0.000134	0.002676	0.000244	0.001859	0.000126	0.001984	0.030233
mean	0.000072	0.001681	0.000215	0.000534	0.000096	0.000924	0.029917
std.	0.000017	0.001246	0.000016	0.000395	0.000013	0.000292	0.000156
IGD	**SZDT4**	**SZDT6**	**DTLZ5**	**WFG1,2d**	**WFG1,3d**	**WFG8,2d**	**WFG8,3d**
best	0.137492	0.084707	0.000299	0.125045	0.161601	0.027181	0.017899
worst	0.606653	0.104575	0.000402	0.175836	0.185154	0.117519	0.061577
mean	0.340396	0.096066	0.000340	0.142345	0.390462	0.044174	0.031567
std.	0.108192	0.004333	0.000017	0.011033	0.004141	0.019187	0.010425

in formulation (P2). Each objective function of (P2) depends on only few of these variables. Hence, there are many points (weakly efficient) which give the individual function optima. This causes a very large unnecessary region to be explored by the NBI and the mNBI method and among this region, only few points ($\sim$5%) are equitably efficient. On the ZDT2 problem, in Figure 5 we see

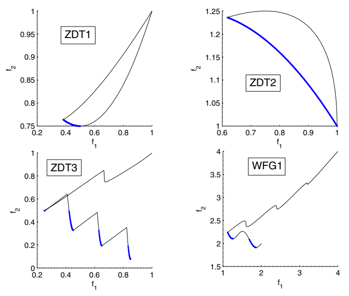


Fig. 3. Plot of the efficient front (black) in the equitable space. The blue points are equitably efficient.

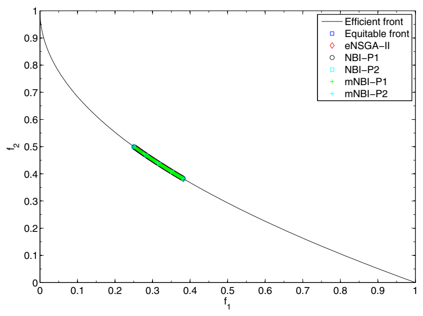


Fig. 4. Performance of all the five algorithms on ZDT1

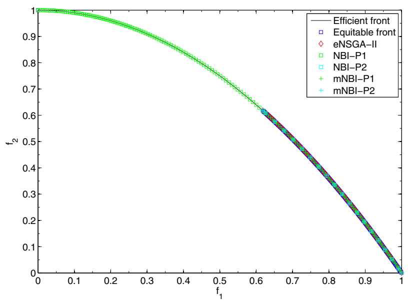


Fig. 5. Performance of all the five algorithms on ZDT2

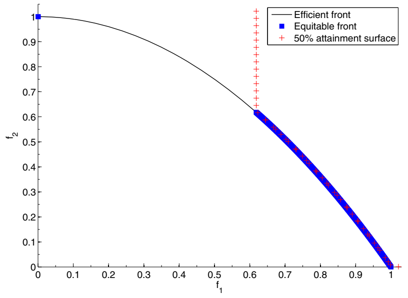


Fig. 6. Median attainment surface plot of eNSGA-II on ZDT2

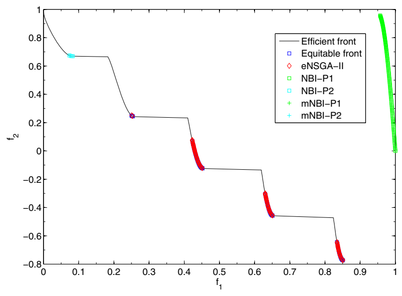


Fig. 7. Performance of all the five algorithms on ZDT3

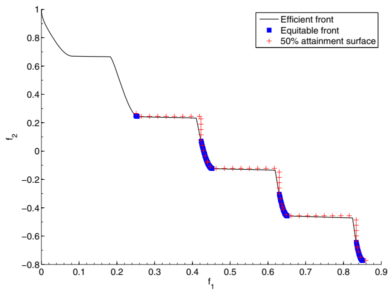


Fig. 8. Median attainment surface plot of eNSGA-II on ZDT3

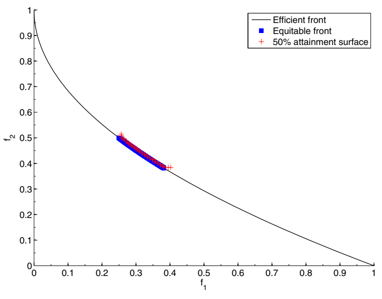


Fig. 9. Median attainment plot of eNSGA-II on ZDT4

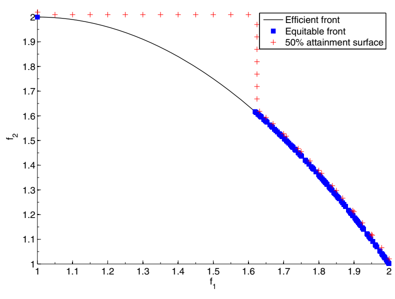


Fig. 10. Median attainment surface plot of eNSGA-II on SZDT2

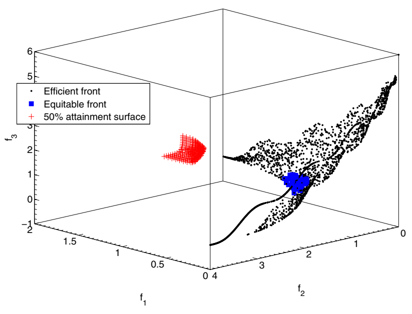


Fig. 11. Median attainment surface plot of eNSGA-II on WFG1 3D

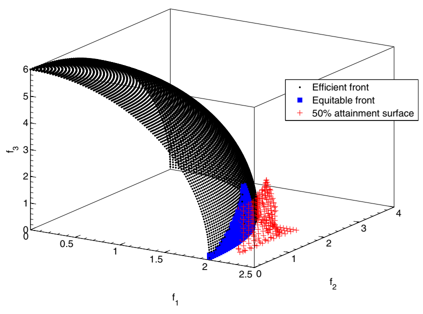


Fig. 12. Median attainment surface plot of eNSGA-II on WFG8 3D

an interesting behavior of the (P1) formulation: non-equitably efficient points are found. Looking at the second subplot in Figure 3 we see that instead of the blue equitably efficient points, the SQP based NBI/ mNBI procedure finds the black points. The methods start from the individual function minima (in equitable space, second subplot in Figure 3) and then use this optimal point as the starting point of the next sub-problem. This causes the behavior in Figure 5. Our experiments showed that if we do not use the individual function minima as the starting points (*warm start* strategy), we need considerably more function evaluations. Figure 7 shows the performance of all the five algorithms on ZDT3. Here also, we see that with the exception of eNSGA-II, the other algorithms either do not find the equitable front or find just few points on it. Table 2 shows the performance of the classical algorithm in terms of the number of function evaluations and the number of failed runs (out of 100). On all the test problems except ZDT1, ZDT2 and ZDT3, the classical algorithms are not able to find more than 2–3 points, if at all they find any. Table 3 and Figures 6, 8, 9, 10 11 and 12 show that the evolutionary based approach eNSGA-II finds a reasonably well-diverse set of equitably efficient points. On the three dimensional WFG problems, we see that eNSGA-II approaches the equitable efficient front in a way so as not to explore the unnecessary regions of the efficient front.

4 Conclusions

This study brings into light that evolutionary algorithms are better suited for finding equitably optimal points than their classical counterparts. The inherent difficulties that are present in various problem formulations, like non-differentiability or a large number of additional variables, prohibit the use of classical techniques, even for simple differentiable problems. The non-differentiability on the other hand, poses no problem for the eNSGA-II method over a wide gamut of problems, ranging from very simple to very difficult. Hence, this paper adequately demonstrates the niche of population based algorithms in finding equitably optimal points. This is the first attempt towards finding a well-diverse representation of equitably optimal solutions and we hope that the study motivates others. It would be interesting to see how other algorithms, like SPEA-2, perform on problems with equitable objectives.

References

[1] Coello Coello, A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Springer, New York (2006)

[2] Ogryczak, W.: Inequality measures and equitable approaches to location problems. European Journal of Operational Research 122, 374–391 (2000)

[3] Ogryczak, W.: Inequality measures and equitable locations. Annals of Operations Research 167, 61–86 (2009)

[4] Sen, A.K., Foster, J.E.: On economic inequality. Oxford University Press, New York (1997)

[5] Ogryczak, W., Wierzbicki, A.: On multi-criteria approaches to bandwidth allocation. Control Cybernet. 33, 427–448 (2004)

[6] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester (2001)

[7] Shukla, P.K., Deb, K.: On finding multiple pareto-optimal solutions using classical and evolutionary generating methods. European Journal of Operational Research 181, 1630–1652 (2007)

[8] Das, I., Dennis, J.: Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM Journal of Optimization 8, 631–657 (1998)

[9] Shukla, P.K.: On the Normal boundary intersection method for generation of efficient front. In: Shi, Y., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2007. LNCS, vol. 4487, pp. 310–317. Springer, Heidelberg (2007)

[10] Eichfelder, G.: Adaptive scalarization methods in multiobjective optimization. Springer, Berlin (2008)

Stopping Criteria for Genetic Algorithms with Application to Multiobjective Optimization

Marcin Studniarski

Faculty of Mathematics and Computer Science, University of Łódź, Poland
marstud@math.uni.lodz.pl

Abstract. For a general Markov chain model of genetic algorithm, we establish an upper bound for the number of iterations which must be executed in order to generate, with a prescribed probability, a population consisting entirely of minimal solutions to a multiobjective optimization problem. However, since populations may contain multiple copies of the same element, we can only guarantee that at least one minimal solution is found. Using this upper bound, we then derive a stopping criterion which ensures that at least one minimal element is a member of the last population generated.

Keywords: Random Heuristic Search, genetic algorithm, stopping criterion, multiobjective optimization.

1 Introduction

Obtaining sensible stopping criteria is an important issue in the theory of genetic algorithms. One of the possible approaches to this problem is to obtain upper bounds for the number of iterations necessary to ensure finding an optimal solution with a prescribed probability (see [1] and references therein). In an earlier paper [6], we have presented some results of this type for a general model of genetic algorithm, based on the theory developed in [4] and [7]. The aim of this paper is to modify the results of [6] so as to obtain some stopping criteria for the case of multiobjective optimization.

2 The RHS Algorithm as a Markov Chain

The RHS (*Random Heuristic Search*) algorithm, described in [7], is defined by an *initial population* $P^{(0)}$ and a *transition rule* τ which, for a given population $P^{(i)}$, determines a new population $P^{(i+1)}$. Iterating τ, we obtain a sequence of populations:

$$P^{(0)} \xrightarrow{\tau} P^{(1)} \xrightarrow{\tau} P^{(2)} \xrightarrow{\tau} \dots \tag{1}$$

Each population consists of a finite number of *individuals* which are elements of a given finite set Ω called the *search space*. Populations are *multisets*, which means that the same individual may appear more than once in a given population.

To simplify the notation, it is convenient to identify Ω with a subset of integers: $\Omega = \{0, 1, ..., n-1\}$. The number n is called the *size of search space*. Then a population can be represented as an *incidence vector* (see [4, p. 141]):

$$v = (v_0, v_1, ..., v_{n-1})^T, \tag{2}$$

where v_i is the number of copies of individual $i \in \Omega$ in the population ($v_i = 0$ if the i-th individual does not appear in the population). The *size of population* v is the number $r = \sum_{i=0}^{n-1} v_i$. We assume that all the populations appearing in sequence (1) have the same size r. Dividing each component of incidence vector (2) by r, we obtain the *population vector*

$$p = (p_0, p_1, ..., p_{n-1})^T,$$

where $p_i = v_i/r$ is the proportion of individual $i \in \Omega$ in the population. In this way, we obtain a more general representation of the population which is independent of population size. It follows that each vector p of this type belongs to the set

$$\Lambda := \left\{ x \in \mathbb{R}^n : x_i \geq 0 \ (\forall i), \ \sum_{i=0}^{n-1} x_i = 1 \right\},$$

which is a simplex in $\mathbb{R}^n$. However, not all points of this simplex correspond to finite populations. For a fixed $r \in \mathbb{N}$, the following subset of Λ consists of all populations of size r (see [7, p. 7]):

$$\Lambda_r := \frac{1}{r} \left\{ x \in \mathbb{R}^n : x_i \in \mathbb{N} \cup \{0\} \ (\forall i), \ \sum_{i=0}^{n-1} x_i = r \right\}.$$

We now define the mapping

$$\mathcal{G} : \Lambda \longrightarrow \Lambda,$$

called *heuristic* [7, p. 9] or *generational operator* [4, p. 144], in the following way: for a vector $p \in \Lambda$ representing the current population, $\mathcal{G}(p)$ is the probability distribution that is sampled independently r times (with replacement) to produce the next population after p. For each of these r choices, the probability of selecting an individual $i \in \Omega$ is equal to $\mathcal{G}(p)_i$, the i-th component of $\mathcal{G}(p)$.

A transition rule τ is called *admissible* if it is a composition of a heuristic $\mathcal{G}$ with drawing a sample in the way described above. Symbolically,

$$\tau(p) = \text{sample}(\mathcal{G}(p)), \quad \forall p \in \Lambda. \tag{3}$$

Of course, a transition rule defined this way is nondeterministic, i.e., by applying it repeatedly to the same vector p, we can obtain different results. It should also be noted that, although $\mathcal{G}(p)$ may not belong to Λ_r, the result of drawing an r-element sample is always a population of size r; therefore, it follows from (3) that $\tau(p) \in \Lambda_r$.

A sequence of random variables $\{X_t\}_{t \in \mathbb{N}_0}$ (where $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$) defined on the same probabilistic space $(Z, \mathcal{F}, \Pr)$, with values in a countable set S (the state space) is called a *Markov chain* if, for every $t \in \mathbb{N}$ and every sequence $s_0, s_1, ..., s_t \in S$, the following condition is satisfied:

$$\Pr\left(X_t = s_t \middle| X_{t-1} = s_{t-1}, ..., X_1 = s_1, X_0 = s_0\right) = \Pr\left(X_t = s_t \middle| X_{t-1} = s_{t-1}\right), \tag{4}$$

provided $\Pr(X_{t-1} = s_{t-1}, ..., X_1 = s_1, X_0 = s_0) > 0$.

A matrix is called *stochastic* if all its elements are nonnegative and the sum of every row is equal to 1. A stochastic matrix $\Pi(t) = [\pi_{i,j}(t)]_{i,j\in S}$ is called the *transition matrix of the Markov chain* $\{X_t\}_{t\in\mathbb{N}_0}$ at time t, $t \geq 1$, if $\pi_{i,j}(t) = \Pr\left(X_t = s_j \middle| X_{t-1} = s_i\right)$ for all $j \in S$ and i such that $\Pr(X_{t-1} = s_i) > 0$. A Markov chain is called *(temporally) homogeneous* if there exists a matrix $\Pi = [\pi_{i,j}]_{i,j\in S}$ being the transition matrix of this Markov chain at every time t.

Let us now return to the RHS algorithm. It generates a sequence of populations

$$\hat{p},\ \tau(\hat{p}),\ \tau^2(\hat{p}), ... , \tag{5}$$

where $\hat{p}$ is a fixed initial population. The RHS can be regarded as a Markov chain where the state space is Λ_r and the values of successive random vectors X_0, X_1, X_2,... are populations (5). Since $\hat{p}$ is fixed, we may assume that X_0 is a random vector taking on the single value $\hat{p}$ with probability 1.

We denote by $\Pr\left(q \mid p\right) = \Pr(\tau(p) = q)$ the probability of obtaining a population q in the current iteration of the RHS algorithm provided the previous population is p. It follows from [7, Thm. 3.4] that

$$\Pr(q \mid p) = r! \prod_{j=0}^{n-1} \frac{(\mathcal{G}(p)_j)^{rq_j}}{(rq_j)!}. \tag{6}$$

Since this probability does not depend on t, we deduce that the RHS algorithm is a homogeneous Markov chain with the constant transition matrix $\Pi = [\pi_{p,q}]_{p,q\in S}$, where $S = \Lambda_r$, and the elements

$$\pi_{p,q} = \Pr(q \mid p) \tag{7}$$

are given by (6).

3 The Transition Matrix of a Genetic Algorithm

In this section we consider a genetic algorithm as a particular case of the RHS. We assume that a single iteration of the genetic algorithm produces the next population form the current population according to the following procedure:

1. Choose two parents from the current population by using a selection method which can be described by some heuristic (see [7, § 4.2]). It should be noted that classical proportional (roulette wheel) selection is not suitable for multiobjective optimization. However, some variants of Pareto ranking or tournament selection, based on partial order, can be used.

2. Crossover the two parents to obtain a child.
3. Mutate the child.
4. Put the mutated child into the next population.
5. If the next population contains less than r members, return to step 1.

We assume that the composition of selection, crossover and mutation can be described in terms of some heuristic that does not vary in time. We also assume that mutation consists in replacing a given individual from Ω by another individual, with a prescribed probability. Let us denote by $u_{i,j}$ the probability that individual i mutates to j. In this way, we obtain a $n \times n$ matrix $U = [u_{i,j}]_{i,j\in\Omega}$. The probability of generating individual $j \in \Omega$ from population p by successive application of selection, crossover and mutation is equal to (compare with the first equation on p. 120 in [4])

$$\mathcal{G}(p)_j = \Pr([j] \mid p)_{scm} = \sum_{i=0}^{n-1} u_{i,j} \Pr([i] \mid p)_{sc}, \tag{8}$$

where the symbol $[i]$ means that we generate a single individual i (not a whole population as in (6)), the subscript *sc* means that we are dealing with the composition of selection and crossover, and the subscript *scm* indicates the composition of selection, crossover and mutation. To get a whole new population, one should draw an r-element sample from probability distribution (8). The probability of generating a population q in this way is equal, by (6) and (8), to

$$\Pr(q \mid p)_{scm} = r! \prod_{j=0}^{n-1} \frac{(\Pr([j] \mid p)_{scm})^{rq_j}}{(rq_j)!}. \tag{9}$$

According to (7), equation (9) gives also a formula for the transition matrix of our algorithm.

4 Stopping Criteria for a Genetic Algorithm

Consider the following multiobjective optimization problem. Let Ω be a finite search space defined in § 2, and let $f : \Omega \to F$ be a function being minimized, where $F = \{f(\omega) : \omega \in \Omega\}$ and $(F, \preceq)$ is a partially ordered set. An element $x^* \in F$ is called a *minimal element* of $(F, \preceq)$ if there is no $x \in F$ such that $x \prec x^*$, where the relation $\prec$ is defined by

$$(x \prec y) :\Leftrightarrow (x \preceq y \wedge x \neq y).$$

The set of all minimal elements of F is denoted by $\mathrm{Min}(F, \preceq)$. We define the set of optimal solutions in our multiobjective problem as follows:

$$\Omega^+ = \mathrm{Min}_f(\Omega, \preceq) := \{\omega \in \Omega : f(\omega) \in \mathrm{Min}(f(\Omega), \preceq)\}. \tag{10}$$

In particular, if F is a finite subset of the Euclidean space $\mathbb{R}^\gamma$, and $f = (f_1, ..., f_\gamma)$, where each component of f is being minimized independently, then the relation $\preceq$ in F can be defined by

$$(x \preceq y) :\Leftrightarrow (x_i \leq y_i,\ i = 1, ..., \gamma).$$

In this case, Ω^+ is the set of all Pareto optimal solutions of the respective multiobjective optimization problem.

Suppose that the goal of RHS is to find as many elements of Ω^+ as possible. We assume that the algorithm stops when, with a prescribed probability, among the populations generated so far there is at least one consisting only of individuals belonging to Ω^+. Let $\Omega^- := \Omega \backslash \Omega^+$, and let S^+ denote the subset of the state space $S = \Lambda_r$ consisting of all populations which do not contain an element of Ω^-:

$$S^+ := \left\{ p \in S : p_i = 0, \ \forall i \in \Omega^- \right\}. \tag{11}$$

Let $S^- := S \backslash S^+$. We denote by A_t the event that the population generated in iteration t contains at least one nonoptimal element:

$$A_t := \left\{ \tau^t(\hat{p}) \in S^- \right\}. \tag{12}$$

The following theorem will be used in the proof of Lemma 1 below.

Theorem 1. [2, p. 38] *Let $\{H_\theta\}_{\theta \in \Theta}$ (where Θ is an arbitrary index set) be a division of sample space Z onto events with positive probability. Then, for any events A and B such that $\Pr(B) > 0$, we have*

$$\Pr(A \mid B) = \sum_{\{\theta : \Pr(B \cap H_\theta) > 0\}} \Pr(A \mid B \cap H_\theta) \Pr(H_\theta \mid B).$$

The following lemma gives an upper bound of the probability that no population in S^+ has been generated in the first t iterations.

Lemma 1. *Suppose that, for some $\alpha \in (0, 1)$, we have that*

$$\sum_{q \in S^-} \pi_{p,q} \leq \alpha, \quad \forall p \in S. \tag{13}$$

Then

$$\Pr\left(A_1 \cap A_2 \cap ... \cap A_t\right) \leq \alpha^t \tag{14}$$

for all $t \in \mathbb{N}$.

Proof. We apply induction with respect to t. First, we shall verify that inequality (14) holds for $t = 1$. Indeed, from assumption (13) for $p = \hat{p}$, we obtain

$$\Pr(A_1) = \sum_{q \in S^-} \Pr(q \mid \hat{p}) = \sum_{q \in S^-} \pi_{\hat{p},q} \leq \alpha.$$

Suppose now that (14) holds for $t = s$. Without loss of generality, we may assume that

$$\Pr\left(A_1 \cap A_2 \cap ... \cap A_s\right) > 0, \tag{15}$$

since in the opposite case condition (14) is satisfied automatically for all $t \geq s$. Taking (15) into account, we obtain form the definition of conditional probability and from our assumption that

$$\begin{aligned}&\Pr\left(A_1 \cap A_2 \cap \ldots \cap A_s \cap A_{s+1}\right)\\ &= \Pr(A_1 \cap A_2 \cap \ldots \cap A_s) \Pr(A_{s+1} \mid A_1 \cap A_2 \cap \ldots \cap A_s)\\ &\leq \alpha^s \Pr(A_{s+1} \mid A_1 \cap A_2 \cap \ldots \cap A_s).\end{aligned} \tag{16}$$

Let $\{H_\theta^s\}_{\theta \in \Theta}$ be the family of all possible events of the form

$$H_\theta^s := \left\{\tau^k(\hat{p}) = p_{\theta,k},\ k = 1, \ldots, s\right\}. \tag{17}$$

In other words, H_θ^s is the event that the following sequence of populations has been generated in the first s iterations:

$$\hat{p},\ p_{\theta,1}\ , \ldots,\ p_{\theta,s}. \tag{18}$$

Since the set of populations S is finite, the family $\{H_\theta^s\}_{\theta \in \Theta}$ is also finite. We assume that, for different indices θ_1 and θ_2, the sequences of populations (18) are different, that is, $p_{\theta_1,k} \neq p_{\theta_2,k}$ for at least one $k \in \{1, \ldots, s\}$; then $H_{\theta_1}^s \neq H_{\theta_2}^s$. Discarding the events H_θ^s with probability zero, we may assume that $\Pr(H_\theta^s) > 0$ for all $\theta \in \Theta$. Hence, the family of events (17) is a division of Z onto disjoint events with positive probability. From Theorem 1, we obtain

$$\begin{aligned}&\Pr(A_{s+1} \mid A_1 \cap A_2 \cap \ldots \cap A_s)\\ &= \sum_{\{\theta : \Pr(A_1 \cap \ldots \cap A_s \cap H_\theta^s) > 0\}} \Pr(A_{s+1} \mid A_1 \cap \ldots \cap A_s \cap H_\theta^s) \Pr(H_\theta^s \mid A_1 \cap \ldots \cap A_s).\end{aligned} \tag{19}$$

If $\Pr(A_1 \cap \ldots \cap A_s \cap H_\theta^s) > 0$, then $A_1 \cap \ldots \cap A_s \cap H_\theta^s \neq \emptyset$. From this fact and from the definitions of the respective sets (equations (12) and (17)), it follows that $p_{\theta,1}, \ldots, p_{\theta,s} \in S^-$, and consequently, $H_\theta^s \subset A_1 \cap \ldots \cap A_s$. Hence,

$$A_1 \cap \ldots \cap A_s \cap H_\theta^s = H_\theta^s. \tag{20}$$

It follows from (19) and (20) that

$$\begin{aligned}&\Pr(A_{s+1} \mid A_1 \cap A_2 \cap \ldots \cap A_s)\\ &= \sum_{\{\theta : \Pr(A_1 \cap \ldots \cap A_s \cap H_\theta^s) > 0\}} \Pr(A_{s+1} \mid H_\theta^s) \Pr(H_\theta^s \mid A_1 \cap \ldots \cap A_s).\end{aligned} \tag{21}$$

We shall now estimate the term $\Pr(A_{s+1} \mid H_\theta^s)$. From the definition of a Markov chain (equation (4)) and from assumption (13), we obtain

$$\begin{aligned}\Pr(A_{s+1} \mid H_\theta^s) &= \Pr(\tau^{s+1}(\hat{p}) \in S^- \mid \tau^k(\hat{p}) = p_{\theta,k},\ k = 1, \ldots, s)\\ &= \sum_{q \in S^-} \Pr(\tau^{s+1}(\hat{p}) = q \mid \tau^k(\hat{p}) = p_{\theta,k},\ k = 1, \ldots, s)\\ &= \sum_{q \in S^-} \Pr(\tau^{s+1}(\hat{p}) = q \mid \tau^s(\hat{p}) = p_{\theta,s})\\ &= \sum_{q \in S^-} \pi_{p_{\theta,s},q} \leq \alpha.\end{aligned} \tag{22}$$

It follows from (21) and (22) that

$$\begin{aligned}&\Pr(A_{s+1} \mid A_1 \cap A_2 \cap ... \cap A_s)\\ &\leq \alpha \sum_{\{\theta:\Pr(A_1\cap...\cap A_s\cap H_\theta^s)>0\}} \Pr(H_\theta^s \mid A_1 \cap ... \cap A_s) = \alpha,\end{aligned} \tag{23}$$

where the last equality follows from the property that the family $\{H_\theta^s : \Pr(A_1 \cap ... \cap A_s \cap H_\theta^s) > 0\}$ is a division of $A_1 \cap ... \cap A_s$ onto events with positive probability. Combining (16) and (23), we get

$$\Pr\left(A_1 \cap A_2 \cap ... \cap A_s \cap A_{s+1}\right) \leq \alpha^{s+1},$$

which completes the proof of (14) by induction. ■

Using Lemma 1, we can easily obtain a lower bound for the probability that a population in S^+ has been generated in the first t iterations. Indeed, let B_t denote the event that population in S^+ has been generated in iteration t:

$$B_t := Z \backslash A_t := \left\{\tau^t(\hat{p}) \in S^+\right\}. \tag{24}$$

Then, assuming (13), we get from (24) and (14)

$$\begin{aligned}\Pr\left(B_1 \cup ... \cup B_t\right) &= \Pr((Z\backslash A_1) \cup ... \cup (Z\backslash A_t))\\ &= \Pr(Z\backslash(A_1 \cap ... \cap A_t))\\ &= 1 - \Pr(A_1 \cap ... \cap A_t) \geq 1 - \alpha^t.\end{aligned} \tag{25}$$

The following theorem gives a more precise lower bound of the form (25) for the genetic algorithm model considered in the earlier sections.

Theorem 2. *We consider the general model of genetic algorithm described in § 3, being a special case of the RHS algorithm with the heuristic $\mathcal{G}$ given by (8). Suppose that the set Ω^+ of optimal solutions has the form*

$$\Omega^+ = \{j_1, j_2, ..., j_m\}, \tag{26}$$

where the (possibly unknown) number m of these solutions is bounded from below by some known positive integer $\bar{m}$. Suppose also that there exists a number $\beta \in (0, 1/\bar{m})$ satisfying

$$u_{i,j} \geq \beta, \quad \forall i,j \in \Omega. \tag{27}$$

Then the probability of generating a population in S^+ in the first t iterations is at least $1 - (1 - (\bar{m}\beta)^r)^t$.

Remark 1. (a) Since any finite partially ordered set has minimal elements, the set Ω^+ is nonempty. Therefore, if no nontrivial lower bound $\bar{m}$ is known, we may always use $\bar{m} = 1$. (b) Condition (27) (with $\beta > 0$) is always satisfied for a genetic algorithm where individuals are strings of symbols from a finite cardinality alphabet, and the mutation is defined by a positive mutation rate (see [3, p. 444, Remark 1]).

Proof. We shall show that the assumption of Lemma 1 is satisfied with $\alpha = 1 - (\bar{m}\beta)^r$. Let $\Pr(S^- \,|\, p)_{scm}$ denote the probability of generating a population in S^- from population p by first applying heuristic $\mathcal{G}$ and then drawing an r-element sample from probability distribution $\mathcal{G}(p)$. Further, let $\Pr([\Omega^+] \,|\, p)_{scm}$ denote the probability of generating an individual in Ω^+ from population p by single application of the operations of selection, crossover and mutation (which is equivalent to drawing a one-element sample from $\mathcal{G}(p)$). Then the left-hand side of (13) can be rewritten as follows:

$$\begin{aligned}\sum_{q \in S^-} \pi_{p,q} &= \sum_{q \in S^-} \Pr(q \,|\, p)_{scm} = \Pr(S^- \,|\, p)_{scm} \\ &= 1 - \Pr(S^+ \,|\, p)_{scm} = 1 - (\Pr([\Omega^+] \,|\, p)_{scm})^r, \end{aligned} \tag{28}$$

where the last equality in (28) follows from the independence of r random variables constituting an r-element sample.

Now, using (8) and (27), we deduce that, for any $p \in S$ and $j \in \Omega$,

$$\Pr([j] \,|\, p)_{scm} \geq \beta \sum_{i=0}^{n-1} \Pr([i] \,|\, p)_{sc} = \beta, \tag{29}$$

where the final equality follows because we add probabilities of disjoint events whose union is the entire sample space Z. Taking into account the representation of Ω^+ given by (26), and using inequality (29), we get

$$\Pr([\Omega^+] \,|\, p)_{scm} = \sum_{l=1}^{m} \Pr([j_l] \,|\, p)_{scm} \geq \sum_{l=1}^{m} \beta = m\beta \geq \bar{m}\beta. \tag{30}$$

Conditions (28) and (30) imply

$$\sum_{q \in S^-} \pi_{p,q} \leq 1 - (\bar{m}\beta)^r.$$

Since by assumption $\beta \in (0, 1/\bar{m})$, so $\alpha = 1 - (\bar{m}\beta)^r \in (0,1)$. Therefore, we can apply Lemma 1 to obtain

$$\Pr(A_1 \cap \ldots \cap A_t) \leq (1 - (\bar{m}\beta)^r)^t,$$

for all $t \in \mathbb{N}$. By using (25), we can estimate the probability of obtaining a population in S^+ in the first t iterations as follows:

$$\Pr(B_1 \cup \ldots \cup B_t) = 1 - \Pr(A_1 \cap \ldots \cap A_t) \geq 1 - (1 - (\bar{m}\beta)^r)^t,$$

which concludes the proof of the theorem. ■

Corollary 1. *For any $\delta \in (0,1)$, we denote by $t_{\min}(\delta)$ the smallest number of iterations required to guarantee that a population in S^+ has been generated with probability δ. Then*

$$t_{\min}(\delta) \leq \left\lceil \frac{\ln(1-\delta)}{\ln(1 - (\bar{m}\beta)^r)} \right\rceil, \tag{31}$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

Proof. By choosing the number of iterations t satisfying the inequality

$$1 - (1 - (\bar{m}\beta)^r)^t \geq \delta, \tag{32}$$

we have guaranteed that a population in S^+ has been generated with probability at least δ. Inequality (32) is equivalent to the following one:

$$t \geq \frac{\ln(1-\delta)}{\ln(1-(\bar{m}\beta)^r)}. \tag{33}$$

For each positive integer t satisfying (32) (or equivalently, (33)), we have that $t_{\min}(\delta) \leq t$. Hence, by taking t equal to the right-hand side of (31), we get the desired inequality for $t_{\min}(\delta)$. ■

Corollary 1 can be used as a stopping criterion for our genetic algorithm in the following way. First, we choose the probability δ (guarantee level) with which we want to find optimal solutions. Then we stop the algorithm after t iterations, where t is the right-hand side of inequality (31). For a real-valued function f, we can compare every two individuals, and consequently, we can store in memory and update the best individual found so far (i.e., the one which has the smallest value of f). Then the best individual found in t iterations is an optimal solution with probability δ (see [6, p. 179]).The situation is more difficult in the multi-objective case. We only know that, with probability δ, one of the populations generated so far belongs to S^+, say $\tau^s(\hat{p}) \in S^+$, where $1 \leq s \leq t$. Unfortunately, we do not know the number s, and so we cannot identify this optimal population. A possible way to overcome this difficulty is discussed below.

The algorithm we shall describe is a combination of the RHS and the base VV (van Veldhuizen) algorithm described in [5, § 3.1]. Suppose we have some RHS satisfying the assumptions of Theorem 2. It generates a sequence (5) of populations, all of them being members of Λ_r. For each $p \in \Lambda_r$, we define the set of individuals represented in population p:

$$\operatorname{set}(p) := \{\omega \in \Omega : p_\omega \neq 0\}.$$

Then we construct a sequence $\{D_t\}$ of subsets of Ω as follows:

$$D_t := \operatorname{set}(\tau^t(\hat{p})), \quad t = 0, 1, \dots\, ,$$

where $\tau^0 :=$ id is the identity mapping. Finally, we define another sequence $\{E_t\}$ of sets recursively by

$$\begin{aligned} E_0 &: = \operatorname{Min}_f(D_0, \preceq), \\ E_{t+1} &: = \operatorname{Min}_f(E_t \cup D_{t+1}, \preceq), \quad t = 0, 1, \dots\, , \end{aligned}$$

where we have used the notation Min_f as in (10). It is shown in [5, Prop. 1] that the sets $f(E_t)$ converge with probability 1 to $\operatorname{Min}(F, \preceq)$ in the sense of some metric. We know from Corollary 1 that, with probability δ, in the first t iterations there has been generated a population belonging to S^+. This means that $D_s \subset \Omega^+$ for some s such that $1 \leq s \leq t$. Using this inclusion, it is not difficult to show that $D_s \subset E_t$ (we omit the details). Hence, among the elements of E_t there is at least one optimal element. Now we know the iteration counter t of E_t (this is our last iteration), but we have no method to identify which elements of E_t are optimal because the size of D_s is unknown. In extreme cases, D_s may even be a singleton. We also do not know if the elements of $E_t \backslash D_s$ are optimal or not. Therefore, it would be desirable to develop some method of eliminating nonoptimal elements form E_t. This will be the subject of further research.

References

1. Greenhalgh, D., Marshall, S.: Convergence Criteria for Genetic Algorithms. SIAM J. Comput. 30, 269–282 (2000)
2. Jakubowski, J., Sztencel, R.: Introduction to Probability Theory. Script, Warszawa (2001) (in Polish)
3. Koehler, G.J., Bhattacharya, S., Vose, M.D.: General Cardinality Genetic Algorithms. Evol. Comput. 5, 439–459 (1998)
4. Reeves, C.R., Rowe, J.E.: Genetic Algorithms — Principles and Perspectives: A Guide to GA Theory. Kluwer, Boston (2003)
5. Rudolph, G., Agapie, A.: Convergence Properties of Some Multi-objective Evolutionary Algorithms. In: Zalzala, A., et al. (eds.) Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), vol. 2, pp. 1010–1016. IEEE Press, Piscataway (2000)
6. Studniarski, M.: Stopping Criteria for a General Model of Genetic Algorithm. In: Twelfth National Conference on Evolutionary Computation and Global Optimization, Zawoja, Poland, pp. 173–181 (2009)
7. Vose, M.D.: The Simple Genetic Algorithm: Foundations and Theory. MIT Press, Cambridge (1999)

Defining and Optimizing Indicator-Based Diversity Measures in Multiobjective Search

Tamara Ulrich, Johannes Bader, and Lothar Thiele

Computer Engineering and Networks Laboratory, ETH Zurich
8092 Zurich, Switzerland
firstname.lastname@tik.ee.ethz.ch

Abstract. In this paper, we elaborate how decision space diversity can be integrated into indicator-based multiobjective search. We introduce DIOP, the diversity integrating multiobjective optimizer, which concurrently optimizes two set-based diversity measures, one in decision space and the other in objective space. We introduce a possibility to improve the diversity of a solution set, where the minimum proximity of these solutions to the Pareto-front is user-defined. Experiments show that DIOP is able to optimize both diversity measures and that the decision space diversity can indeed be improved if the required maximum distance of the solutions to the front is relaxed.

1 Motivation

The task of evolutionary multiobjective optimization (EMO) includes to find a set of Pareto-optimal solutions which is as diverse as possible to offer the decision maker a good selection of solutions. Traditionally, diversity relates to objective values. Only recently, multiobjective algorithms also aim at finding solutions that are diverse in the decision space. Maintaining multiple solutions which cover different parts of the decision space, e.g. different designs, offers many advantages: first, it enables the decision maker to choose among different designs with the same or at least equally preferable objective values; second, it helps the decision maker to gather information about the problem structure; and third, it can speed up search—for instance by improving exploration and preventing premature convergence.

Many algorithms have been proposed to promote diversity of solutions also in the decision space. However, the exact optimization goal is often far from clear. The Omni-Optimizer [4] for example is based on a crowding distance, which prefers solutions with large distance to the remaining solutions and alternates between the distance in the objective space and in the decision space. In this setting, the optimal set of solutions is not well-defined, nor is it easily possible to specify the desired tradeoff between diversity in the objective space and diversity in the decision space.

We here make the following assumptions about the preference of a decision maker:

1. The decision maker is interested in a set of solutions.
2. Each solution in this so-called *target population* should be close to optimal, i.e., not "far" from the Pareto-front in objective space.

3. The target population should cover large parts of the Pareto-front or regions nearby and should therefore offer objective space diversity.
4. The target population should cover large parts of the decision space, i.e. offering decision space diversity.

Diversity is inherently a property of sets of solutions rather than single solutions as individual solutions can only be divers with respect to others. Therefore, optimizing diversity is closely linked to the set-based view on multiobjective optimization as proposed in SPAM [21] for example. The advantages of formalizing the optimization goal by a set-based preference relation are twofold: A preference relation defines which of two sets is preferred, and therefore, the optimization goal of the multiobjective search is clearly defined. In addition, the convergence of algorithms using this preference relation can be proven under certain conditions.

This study makes the following contributions to optimization considering diversity:

– A new diversity measure of sets in decision space is proposed which has not been used in evolutionary multiobjective optimization before. This measure imposes less strict requirements on the decision space properties than other commonly used diversity measures. An efficient procedure is presented to use this set diversity as a selection criterion for solutions during the optimization process.
– We introduce the possibility of predefining a maximal distance to the Pareto-front, that must not be exceeded by any solution. This mechanism enables a decision maker to explore the tradeoff between diversity in decision space and solution optimality.
– We provide experimental results which compare the proposed method to the well-established Omni-Optimizer [4] and which show the influence of the different parameters involved.

2 Background and Notation

Consider a multiobjective optimization problem with a decision space X and an objective space $Z \subseteq \mathbb{R}^n = \{f(a) \,|\, a \in X\}$, where $f : X \to Z$ denotes a mapping from the decision space to the objective space with n objective functions $f = \{f_1, ..., f_n\}$ which are to be minimized. An element $a \in X$ of the decision space is also named a solution.

The underlying preference relation is weak Pareto dominance, where a solution $a \in X$ weakly dominates another solution $b \in X$, denoted $a \preceq b$, if and only if solution a is better or equal than b in all objectives, i.e., $a \preceq b$ iff $f(a) \leqslant f(b)$ or equivalently, iff $f_i(a) \leq f_i(b), \forall\, i \in \{1, ..., n\}$. Furthermore, we will use the notion of weak ε-Pareto-dominance defined as $a \preceq_\varepsilon b$ iff $f(a) - \varepsilon \leqslant f(b)$. In other words, suppose that we improve solution a in any objective by ε. Then $a \preceq_\varepsilon b$ iff the improved solution weakly dominates solution b.

Let $X^* \subseteq X$ denote the Pareto-optimal set, $X^* = \{x \,|\, \nexists a \in X : a \preceq x \wedge x \npreceq a\}$, let $T \subset X$ denote a target population of solutions, and let $q_{X^*} : X \to \mathbb{R}^{\geq 0}$ measure for each solution $x \in X$ the distance $q_{X^*}(x)$ to the Pareto-optimal set X^*. Let $D_o(T) : 2^X \to \mathbb{R}^{\geq 0}$ and $D_d(T) : 2^X \to \mathbb{R}^{\geq 0}$ measure the diversity of a set of solutions $T \subseteq X$ in the objective space ($D_o(T)$) and in the decision space ($D_d(T)$), respectively. Given this notation, the four optimization assumptions provided in Section 1 can be formalized as follows:

1. We are interested in a target population of solutions $T \subseteq X$, $|T| = \mu$, where μ denotes its size.
2. Optimality: $\forall t \in T : q_{X^*}(t) \leq \varepsilon$, where ε is given bound on the optimality of solutions in T.
3. Diversity in objective space: Determine T such that $D_o(T)$ is maximal among all possible target populations.
4. Diversity in decision space: Determine T such that $D_d(T)$ is maximal.

As a consequence, we are dealing with a *bi-objective optimization problem on sets of solutions*. Given this setting, different problems arise:

– In order to determine $q_{X^*}(T)$ one needs the knowledge of the Pareto-optimal set of solutions X^*, which in general is not known.
– The problem of optimizing diversity in objective and decision space is a bi-objective problem on the set of all possible populations. It is not clear which tradeoff the decision maker is interested in and how to express these tradeoffs in an optimization method.
– There are many choices for the distance and diversity measures q_{X^*}, D_o and D_d. Guidelines are necessary to choose appropriate measures (see the following Sec. 3).

3 Measuring Diversity–Approaches in Biology and in EAs

Typically, measures for the diversity of a set are based on the definition of a pairwise distance between any two elements. Therefore, we assume that we are given a distance measure $d : X^2 \rightarrow \mathbb{R}^{\geq 0}$ on the decision space. Here, we are often confronted with many different classes of decision spaces, such as vectors, graphs, trees or even programs. In order to be applicable to a large class of optimization domains, we would like to place as few restrictions on the structure of the decision space as possible, i.e. we do not require that X is an Euclidean space or that the triangle inequality is satisfied. Instead, we just assume X to be a semimetric space, i.e., $\forall a, b \in X$: $d(a, b) \geq 0$ (*non-negativity*), $d(a, b) = d(b, a)$ (*symmetry*), $d(a, a) = 0$ (*identity of indiscernibles*). Given such a distance measure, we now would like to define a set diversity measure $D : 2^X \rightarrow \mathbb{R}^{\geq 0}$ which assigns to each subset of the decision space a real value, i.e. its diversity.

There are many possible interpretations and concepts of set diversity, i.e. how a given number of solutions should be distributed such that they achieve an optimal set diversity. In order to get a first insight, let us consider a very simple example. Figure 1 shows the optimized distribution of 100 points in a two dimensional Euclidean space $X = [0, 1]^2$ for two diversity measures, namely the sum of all pairwise distances and the Solow-Polasky [12] measure. While the Solow-Polasky measure gives a grid-like structure, the sum of pairwise distance measure distributes all 100 solutions into the four corners. As a result, it appears that we need to define a set of formal requirements for a useful diversity measure.

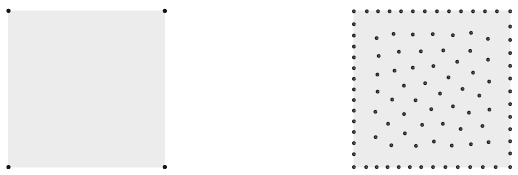


Fig. 1. Best distributions found by the hill-climber, for the sum of pairwise distances diversity measure (left) and the Solow-Polasky measure (right)

Measuring diversity of sets is much-discussed in biology, more specifically in the field of biodiversity. Just as for the decision maker's preference, no generally agreed-on definition exists neither in biology nor in the field of evolutionary algorithms. In the following, we discuss the most prominent classes of existing biodiversity measures with respect to their applicability to EAs. In particular we consider the following three requirements to a diversity measure D, first proposed by Solow and Polasky [12]:

P1: Monotonicity in Varieties. The diversity of a set of solutions A should increase when adding an individual b not yet in A, i.e., $D(A \cup b) > D(A)$ if $\min_{a \in A} d(a, b) > 0$. This fundamental property assures that increased species richness is reflected by the diversity measure [6].

P2: Twinning. Diversity should stay constant when adding an individual c already in A, i.e., $D(A \cup c) = D(A)$. Intuitively, if diversity is understood as the coverage of a space by a set of solutions [17], adding duplicates should not increase the coverage and the chosen diversity measure should reflect that property.

P3: Monotonicity in Distance. The diversity of set A should not decrease if all pairs of solutions are at least as dissimilar (measured by d) as before $D(A') \geq D(A)$, iff $d(a'_i, a'_j) \geq d(a_i, a_j), \forall a_i, a_j \in A, a'_i, a'_j \in A'$. So the more dissimilar solutions are, the better.

One straightforward way of measuring diversity is based on the *relative abundance* of each solution present in set A, e.g. [5]. But the degree of dissimilarity between individuals has no influence and the twinning property is not fulfilled. The second group of diversity measures is based on *taxonomy*, e.g. [19], but unfortunately building the taxonomic tree has a runtime which is exponential in the number of individuals. A very simple way of *aggregating the dissimilarity* information into a diversity measure is to sum up the values, $D(A) = \sum_{a \in A} \sum_{b \in B} d(a, b)$ [6]. Shir et al. for instance used this measure in their EA [11], while the Omni-Optimizer considers the distance d to the closest neighbors of a solution. However, these measures do not meet the twinning requirement and they promote having only two solutions with large distance duplicated multiple times. A completely new approach has been presented by Solow and Polasky [12]. Their measure is based on an *utilitarian view on individuals*, where the function $u : X \rightarrow \mathbb{R}^{\geq 0}$ defines the utility of any subset of solutions. This view of utility is equivalent to the method proposed in a previous study of the authors [17], where instead of utility the area covered by individuals has been considered. All three above requirements are fulfilled.

Table 1. Comparison of different diversity metrics with respect to the three properties: monotonicity in varieties (P1), twinning (P2), and monotonicity in distance (P3)

class	**method**	**P1**	**P2**	**P3**
relative abundance	Simpson, Shannon, Berger-Parker	no	no	yes
taxonomy	clustering	no	yes	no
	Weitzman	yes	yes	no
functions of distance	sum	yes	no	yes
	crowding distance	no	no	yes
utilitarian	Solow-Polasky	yes	yes	yes

In the *evolutionary algorithm literature*, decision space diversity has often been used to prevent premature convergence. Examples of measures can be found in [16], [10], [13], [18,17], [4], [14], [20], [7] or [8,4]. Most of these measures either require a specific structure of the decision space, they do not define a measure on sets, they make assumptions about the Pareto-front or the problem landscape or they do not satisfy the required properties.

Table 1 summarizes the different diversity measures in context of the three requirements $P1$, $P2$ and $P3$. As can be seen, only the measure by Solow-Polasky satisfies all three requirements, so we will apply this measure in the experimental study (Sec. 5). However, the algorithmic framework presented in this paper is also compatible with other measures.

4 Optimizing Diversity – A Novel Set-Based Algorithm

Now that we have presented some possibilities to measure diversity, be it to determine $D_o(A)$ or $D_d(A)$, the decision maker's preference 3 and 4 stated in Sec. 1 can be formally expressed. Optimizing those indicator-based set preferences can be accomplished within the SPAM framework [22]. There remain, however, a number of issues to be resolved which we are going to tackle with DIOP (Diversity Integrating Optimizer).

As the *Pareto-optimal set X^* in general is unknown*, we propose using a helper set, called the *archive* A, which approximates X^*. We therefore have two concurrent EAs, one which optimizes the target population and one which optimizes the archive population. This offers the advantage that the quality constraint (decision maker preference 2, Sec. 1) continuously tightens as the archive population improves. In order to benefit from one another, the two sets can exchange solutions, therefore improving the diversity in the archive and producing more solutions that satisfy the quality constraint in the target. This is useful as experiments have indicated that considering diverse solutions might speed up search for some problems [17].

Having an approximation A of the Pareto-optimal set X^*, a *distance metric* q_A has to be defined. We here propose to use $\preceq_\varepsilon$ to define the distance as the smallest ε to reach ε-dominance of any solution in A, i.e.,

$$q_A(x) := \min\{\varepsilon \mid \exists y \in A : x \preceq_\varepsilon y\} \ . \tag{1}$$

Algorithm 1. DIOP algorithm. Takes a parameter ε, an archive size μ^a, a target size μ^t, and a decision space X. Returns the optimized target set.

```
function DIOP($\varepsilon$, $\mu^a$, $\mu^k$)
    $A = \{x_1, ..., x_{\mu^a}\}, x_i \in X$ /* randomly initialize archive */
    $T = \{x_1, ..., x_{\mu^t}\}, x_i \in X$ /* randomly initialize target */
    while stopping criterion not met do
        $A' = variate(A \cup T)$ /* generate archive offspring */
        $A'' = archiveSelect(A \cup A' \cup T, \mu^a)$ /* select $\mu^a$ new individuals */
        /* Only use new archive if its $D_o$ value is better */
        if $D_o(A'') > D_o(A)$ then
            $A = A''$
        end if
        $T' = variate(A \cup T)$ /* generate target offspring */
        $T'' = targetSelect(A, T \cup T' \cup A, \mu^t, \varepsilon)$ /* select $\mu^t$ new individuals */
        /* Only use new target if its $D_d$ value is better */
        if $G(T'') > G(T)$ then
            $T = T''$
        end if
    end while
    Return $T$
end function
```

As the decision maker is only interested in solutions not exceeding a predefined distance ε to the Pareto-front, the diversity measures of an arbitrary set P is only calculated for those solutions $P^\varepsilon \subseteq P$ not exceeding the distance ε from the front approximation A, $P^\varepsilon = \{p \in P \,|\, q_A(p) \leq \varepsilon\}$.

In contrast to the framework of SPAM [22], DIOP needs to *maximize two indicators D_o and D_d instead of one*. Therefore, two sets can no longer be unambiguously compared in general, as we are dealing with a biobjective problem. Many other studies have implicitly tackled this tradeoff, however, to the best of the authors' knowledge, none of these approaches explicitly set the tradeoff, but use subpopulations [9,16], adapt mutation [18], use the diversity to the best single objective solution [10], use the contribution to the set diversity [13], use nondominated sorting [14], use a sequence of indicators [22], alternate between decision space and objective space diversity [4], use an unweighted sum of both measures [11], use diversity as an additional objective [15], adapt the variation process [20] or integrate diversity into the hypervolume indicator [17].

In this study, we propose to consider a weighted sum of the two diversity indicators:

$$G(T) := w_o \cdot D_o(T) + w_d \cdot D_d(T),\ |T| = \mu \text{ with } q_A(t) \leq \varepsilon\ \forall t \in T, w_o + w_d = 1 \quad (2)$$

This enables a flexible tradeoff between the two diversity indicator values D_o and D_d by using different weights.

The DIOP algorithm simultaneously evolves two population, namely the archive A which approximates X^*, and the target population T which maximizes $G(T)$. Offspring is always generated from the union of both sets, whereas the selection procedure uses different indicators for the archive and target. In each generation, a selected subset is only accepted if the corresponding indicator value is larger than the one of the parent population. The pseudocode of the proposed algorithm is shown in Algorithm 1.

The function $A'' = archiveSelect(A, \mu^a)$, selects μ^a solutions A'' from a set A. The selection goal is to maximize $D_o(A'')$. The function $P' = variate(P, m)$ generates

m offspring P' from a given set P. The method $T' = targetSelect(A, T, \mu^t, \varepsilon)$ selects μ^t solutions T' from set T. The goal here is to maximize $G(\{t \in T : q_A(t) \leq \varepsilon\})$.

5 Experimental Results

In this section, two main questions are investigated: first, how do the parameters of DIOP, i.e. ε and w_o, influence the obtained target population in terms of the two diversity measures D_d and D_o? Second, we compare DIOP on two test problems to the Omni-Optimizer [4] to assess its performance.

Experimental Setup: The method $variate(P, m)$ selects $m/2$ random pairs of solutions from P to generate the offspring population. These pairs are then recombined by the SBX crossover operator [2] and mutated by adding a new normally distributed value with standard deviation $1/\eta_m$. Solow-Polasky with $\eta_{SP} = 10$ is used to measure the decision space diversity D_d. To determine the objective space diversity D_o, the hypervolume indicator is used with iterative greedy environmental selection as described in [22]. To perform the target selection $targetSelect(T, n)$ according to $G(T)$ (Eq. 2), the following wide-spread greedy strategy is used: Starting with an empty set $T' = \{\}$, iteratively the solution $t_i \in T$ is added to T' which leads to the largest indicator increase $\Delta_{ti} G(T') := w_o(D_o(T' \cup t_i) - D_o(T')) + w_d(D_d(T' \cup t_i) - D_d(T'))$ Since determining the diversity measure of Solow-Polasky is costly (involving matrix inverses [12]), we use the following approximation: $D_d(T' \cup t_i) - D_d(T') \approx \min_{a \in T' \setminus t_i} d(t_i, a)$, i.e., we take the utility lost with respect to the closest individual as the overall utility loss.

Influence of $\boldsymbol{\varepsilon}$ and $\boldsymbol{w_o}$: To assess the influence of the parameters ε and w_o, DIOP is run on DTLZ2 [3] with 3 objectives and $d = 7$ decision variables. DTLZ2 was chosen as it is a well-known problem, its results are easy to interpret as the connection between decision space values and objective space values is known, and the true Pareto-front is known. Note though that DIOP can also be run on real-world problems with more complex decision spaces that are not metric. The variation parameters are set according to [3] with a crossover probability of 1 with $\eta_c = 15$ and a variable exchange probability of 0.5, as well as a mutation probability of $1/d$ with $\eta_m = 20$. We chose the archive and target size to be 50 and run the algorithm for 1000 generations. The parameter ε takes the values $\{0, 0.0865, 1\}$, the weights $w_o = \{0, 0.7692, 0.9091, 0.9677, 1\}$ are logarithmically spaced with $w_d = 1 - w_o$. The results are shown in Fig. 2 on the left hand side.

It can be seen that with an increasing ε and an increasing w_d value, the achievable diversity increases, while the hypervolume decreases. This illustrates how the tradeoff between hypervolume and decision space diversity can be set by the user. Figure 3 shows the non-dominated solutions for one run with $\varepsilon = 0$ and $w_o = \{0, 0.7699, 1\}$. The higher w_o is, the more solutions lie on the Pareto-front (50/8/1 out of 50 solutions lie on the front for $w_o = 1/0.7699/0$, respectively). The dominated solutions do not contribute to the hypervolume and are distributed within the quality constraint set by ε and A in such a way that they optimize diversity. This indicates how the tradeoff is set in practice: A subset of the final target population is distributed on the Pareto-front and optimizes the hypervolume, whereas the remaining solutions optimize decision space diversity. The number of solutions that optimize the hypervolume increases with w_o.

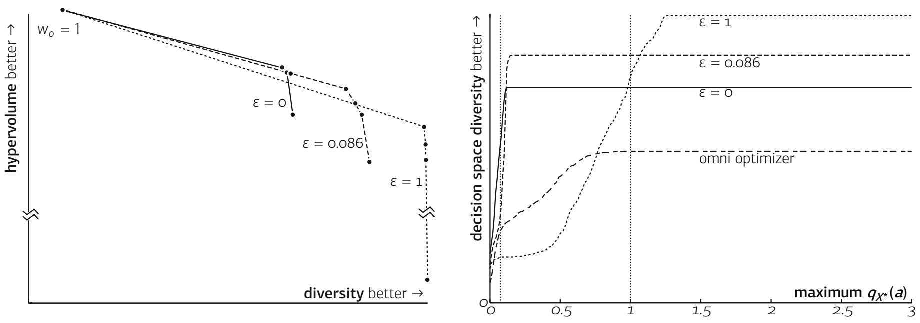


Fig. 2. Left: Influence of the ε values on the diversity according to Solow-Polasky (x-axis) and the hypervolume (y-axis) for $\varepsilon \in \{0, 0.86, 1\}$. For each ε, the weight w_o is decreased from 1 (leftmost data points) to 0 (rightmost data points). Right: Decision space diversity (y-axis) of all solutions within a certain distance (x-axis) of the true Pareto-front.

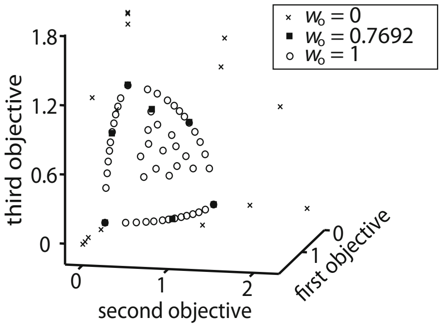


Fig. 3. Non-dominated solutions of one DTLZ2 run for three different weights

Comparison to the Omni-Optimizer. While the Omni-optimizer uses the same SBX crossover operator as DIOP, it uses an adaption of polynomial mutation with $\eta_m = 20$ [4] instead of the Gaussian mutation employed by DIOP.

As the first test problem we use the Omni-Test as described in [11] with 5 decision variables. The Omni-Test was chosen because it allows for an additional intuitive problem-specific diversity measure, which exploits the fact that the Pareto-optimal solutions are distributed over a total of 3^d clusters, where d is the number of decision variables. Therefore, the additional diversity measure can be defined as the number of clusters found by the algorithm. For optimization, we use the parameters from [11] with a population size of 50, 1000 generations, and $\varepsilon = 0$. As we use an archive of size 50 in addition to the target of size 50, we require more fitness evaluations than the Omni-Optimizer. In order to compensate for that, the Omni-Optimizer is run for twice as many generations, i.e. 2000. For the variation operators, we use the parameters from [4] with a crossover probability of 0.9 with $\eta_c = 1$, where the variables are exchanged with a probability of 0.5, and a mutation probability of $1/n$ with $\eta_m = 1$. Each algorithm was run 15 times with different random seeds. To test the two algorithms for statistically significant differences, the Kruskal-Wallis with post-hoc Conover-Inman procedure [1] is

Table 2. Omni-Test problem: Four measures, all to be maximized. Statistically significantly better/worse results of DIOP compared to the Omni-optimizer are marked with a $^+/^-$.

	Hypervolume	Diversity (Pairs)	Diversity (Solow)	Found Clusters
DIOP $w_o = 0.00$	$30.04 \pm 0.10^+$	$0.63 \pm 0.05^-$	$46.7 \pm 2.0^+$	$33.2 \pm 5.2^+$
DIOP $w_o = 0.77$	$30.21 \pm 0.03^+$	$0.64 \pm 0.05^-$	$47.9 \pm 2.3^+$	$37.3 \pm 5.2^+$
DIOP $w_o = 0.91$	$30.25 \pm 0.03^+$	$0.66 \pm 0.06^-$	$48.7 \pm 0.9^+$	$39.9 \pm 3.7^+$
DIOP $w_o = 0.97$	$30.31 \pm 0.02^+$	$0.65 \pm 0.06^-$	$48.5 \pm 1.0^+$	$39.5 \pm 4.0^+$
DIOP $w_o = 1.00$	$30.42 \pm 0.00^+$	$0.43 \pm 0.11^-$	$16.1 \pm 2.5^-$	$9.8 \pm 2.6^-$
Omni	29.94 ± 0.05	0.70 ± 0.04	34.7 ± 1.1	23.8 ± 1.4

applied with a significance level of 5%. The results are given in Table 2. It can be seen that DIOP achieves significantly better hypervolume values than the Omni-optimizer for all weight combinations, even if only decision space diversity is optimized. This is due to the fact that the solutions, while optimizing decision space diversity, must not be dominated by any archive solutions ($\varepsilon = 0$). Even though DIOP finds twice as many clusters as the Omni-Optimizer (except for $w_o = 1$, i.e. when the decision space diversity is not optimized at all), its pairwise distance measure is significantly worse than the Omni-Optimizers. This indicates that the pairwise distance measure does not accurately reflect the number of found clusters. DIOP's Solow-Polasky values, on the other hand, are significantly better than the Omni-optimizer's, as expected.

As a second test problem, we selected DTLZ2 with 3 objectives and 7 decision variables. In this test problem, the last 5 decision variables of a Pareto-optimal solution are equal to 0.5, whereas the first two variables define its location on the front. Solutions with values that differ from 0.5 in the last 5 variables are not Pareto-optimal. The population sizes and generation numbers are the same as for the Omni-Test problem. DIOP was run for $\varepsilon = \{0, 0.0856, 1\}$, with the weights set to $w_o = 0.9677$, $w_d = 0.0333$. The algorithms were again run 15 times with different seeds. The results are shown in Figure 2 on the right hand side. At each point in the figure, all solutions that are within the distance given on the x-axis from the true Pareto-front are used to calculate the decision space diversity D_d, which gives the corresponding y-axis value. The results show that the Omni-Optimizer has problems approximating the Pareto-front. Its diversity remains close to zero until about a distance of 0.5 from the front, which is due to the fact that it finds only few solutions that are closer than 0.5 to the true front. The DIOP population reaches its maximum diversity at a distance of around 0.2 from the front, which is an effect of the fact that not the true front but an approximation thereof is used during the optimization. For $\varepsilon = 0$ and $\varepsilon = 0.0856$, DIOPs population has a better diversity than the Omni-optimizer no matter what distance from the front is considered. For $\varepsilon = 1.0$, the solutions seem to be located in a distance interval between 0.5 and 1.2 from the front, which indicates that solutions further away from the front are more diverse than those close to the front. This matches the DTLZ2 problem; the further the last 5 decision variables are from their optimal value of 0.5, the better the diversity and the larger the distance to the front gets.

6 Conclusions

In this paper we investigate how decision space diversity can be integrated into indicator-based search. Experiments show that the algorithm can generate various tradeoffs between objective and decision space diversity, adjustable by the user. Furthermore, it is shown that the algorithm performs well when compared to the well-known Omni-Optimizer. In the future, DIOP should be tested on more complex, non-Euclidean problems. Also, it could be compared to other state-of-the-art multiobjective optimizers that do not optimize diversity, in order to quantify the increase in diversity that can be gained from using DIOP.

References

1. Conover, W.J.: Practical Nonparametric Statistics, 3rd edn. John Wiley, Chichester (1999)
2. Deb, K., Agrawal, S.: A niched-penalty approach for constraint handling in genetic algorithms. In: ICANNGA (1999)
3. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Test Problems for Evolutionary Multi-Objective Optimization. TIK Report 112, ETH Zurich (2001)
4. Deb, K., Tiwari, S.: Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. EJOR 185(3), 1062–1087 (2008)
5. Gaston, K.J., Spicer, J.I.: Biodiversity: An Introduction, 2nd edn. Wiley-Blackwell, Chichester (2004)
6. Izsák, J., Papp, L.: A link between ecological diversity indices and measures of biodiversity. Ecological Modelling 130(1-3), 151–156 (2000)
7. Li, X., Zheng, J., Xue, J.: A Diversity Metric for Multi-objective Evolutionary Algorithms. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 68–73. Springer, Heidelberg (2005)
8. Rudolph, G., Naujoks, B., Preuss, M.: Capabilities of EMOA to detect and preserve equivalent pareto subsets. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 36–50. Springer, Heidelberg (2007)
9. Sarma, J., Jong, K.A.D.: An analysis of the effects 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)
10. Shimodaira, H.: A diversity-control-oriented genetic algorithm (dcga): Performance in function optimization. In: Genetic and Evolutionary Computation Conference, p. 366 (2000)
11. Shir, O.M., Preuss, M., Naujoks, B., Emmerich, M.: Enhancing decision space diversity in evolutionary multiobjective algorithms. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 95–109. Springer, Heidelberg (2009)
12. Solow, A.R., Polasky, S.: Measuring biological diversity. Environmental and Ecological Statistics 1(2), 95–103 (1994)
13. Squillero, G., Tonda, A.P.: A novel methodology for diversity preservation in evolutionary algorithms. In: GECCO, pp. 2223–2226. ACM, New York (2008)
14. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
15. Toffolo, A., Benini, E.: Genetic diversity as an objective in multi-objective evolutionary algorithms. Evolutionary Computation 11(2), 151–167 (2003)
16. Tsutsui, S., Fujimoto, Y., Ghosh, A.: Forking genetic algorithms: Gas with search space division schemes. Evolutionary Computation 5(1), 61–80 (1997)
17. Ulrich, T., Bader, J., Zitzler, E.: Integrating Decision Space Diversity into Hypervolume-based Multiobjective Search. In: Genetic and Evolutionary Computation Conference (to appear, 2010)

18. Ursem, R.K.: Diversity-guided evolutionary algorithms. In: Congress on Evolutionary Computation, pp. 1633–1640 (2002)
19. Weitzman, M.: On diversity. The Quarterly Journal of Economics 107(2), 363–405 (1992)
20. Zhou, A., Zhang, Q., Jin, Y.: Approximating the set of pareto optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm. IEEE Transactions on Evolutionary Computation (2009) (accepted)
21. Zitzler, E., Thiele, L., Bader, J.: On Set-Based Multiobjective Optimization (Revised Version). TIK Report 300, ETH Zurich (2008)
22. Zitzler, E., Thiele, L., Bader, J.: On Set-Based Multiobjective Optimization. IEEE Transactions on Evolutionary Computation (2009) (to appear)

On Expected-Improvement Criteria for Model-based Multi-objective Optimization

Tobias Wagner[1], Michael Emmerich[2], André Deutz[2], and Wolfgang Ponweiser[3]

[1] Institute of Machining Technology (ISF)
Technische Universität Dortmund, 44227 Dortmund, Germany
wagner@isf.de

[2] Leiden Institute of Advanced Computer Science (LIACS)
Universiteit Leiden, 2333 CA Leiden, The Netherlands
{emmerich,deutz}@liacs.nl

[3] Automation and Control Institute
Vienna University of Technology, 1040 Vienna, Austria
ponweiser@acin.tuwien.ac.at

Abstract. Surrogate models, as used for the Design and Analysis of Computer Experiments (DACE), can significantly reduce the resources necessary in cases of expensive evaluations. They provide a prediction of the objective and of the corresponding uncertainty, which can then be combined to a figure of merit for a sequential optimization. In single-objective optimization, the expected improvement (EI) has proven to provide a combination that balances successfully between local and global search. Thus, it has recently been adapted to evolutionary multi-objective optimization (EMO) in different ways. In this paper, we provide an overview of the existing EI extensions for EMO and propose new formulations of the EI based on the hypervolume. We set up a list of necessary and desirable properties, which is used to reveal the strengths and weaknesses of the criteria by both theoretical and experimental analyses.

Keywords: Design and Analysis of Computer Experiments, Expected Improvement, Hypervolume Indicator, Multi-Objective Optimization.

1 Introduction

Surrogate modeling has become the method of choice to overcome the problem of expensive evaluations in EMO [1]. Using the evaluations already available, surrogate models of the objectives are created, which can then be used to filter or decide on candidate solutions. To accomplish this, a criterion which scalarizes the predictions of the models is required. This criterion should balance between a local refinement of the Pareto-front (PF) approximation and an improvement of the global model quality.

In this paper, such criteria for multi-objective optimization are presented, analyzed, and discussed. The main definitions are provided, existing criteria are summarized, and enhancements in the calculation of these criteria are proposed in section 2 and 3. For the evaluation of the criteria necessary requirements and desired properties are formulated in section 4. By means of both, formal and empirical, analyses, we study whether these requirements and properties are met by the various criteria. Concluding, a summary of the results and an outlook on further research topics are provided in section 5.

2 Single-Objective Optimization Based on EI

A surrogate model allows the objective function value $y = f(\mathbf{x})$ of a decision vector $\mathbf{x}$ to be predicted without an expensive evaluation. This prediction is denoted as $\hat{y}$.[1] Since an evaluation is particularly worthwhile if it provides an improvement to the current state of the optimization, often the improvement $I(\hat{y}, f_{min}) = \max\{f_{min} - \hat{y}, 0\}$ obtained with respect to the best currently known objective value f_{min} is maximized. Consequently, we consider minimization of the objectives. Many modeling techniques, such as the ones used in DACE [2], predict both the mean $\hat{y}$ and the standard deviation $\hat{s}$ of a normal distribution. Consequently, the probability density function (PDF) $\phi_{(\hat{y},\hat{s})}(y) = \phi_{(0,1)}(\frac{y-\hat{y}}{\hat{s}})$ and the cumulative density function (CDF) $\Phi_{(\hat{y},\hat{s})}(y) = \Phi_{(0,1)}(\frac{y-\hat{y}}{\hat{s}})$ of an objective value y can be computed (cf. Fig. 1). Based on the definition of the improvement $I(y, f_{min})$ and the PDF of y, the expected value of the improvement

$$EI(\hat{y}, \hat{s}, f_{min}) = \int_{-\infty}^{\infty} I(y, f_{min}) \underbrace{\phi_{(\hat{y},\hat{s})}(y)}_{\text{PDF}(y)} \, dy = \int_{-\infty}^{f_{min}} (f_{min} - y)\phi_{(\hat{y},\hat{s})}(y) \, dy \quad (1)$$

has been proposed as criterion by the Vilnius school of global optimization, e. g. [3]. Later, the EI has become popular as part of the single-objective Efficient Global Optimization (EGO) [4] approach.[2] EGO makes an extensive use of DACE models by only evaluating one solution in each iteration on the true objective function, refitting the model, and then determining the next candidate solution based on the EI. Since both predictions, $\hat{y}$ and $\hat{s}$, are considered, a balancing between a local search and a reduction of the model uncertainty is achieved. Thereby, the number of function evaluations could be significantly reduced for many global optimization problems – often below one hundred.

By expanding equation 1 and integrating the first factor, the EI can also be written as $f_{min}\Phi_{(\hat{y},\hat{s})}(f_{min}) - \int_{-\infty}^{f_{min}} y\phi_{(\hat{y},\hat{s})}(y) \, dy$, and thus

$$EI(\hat{y}, \hat{s}, f_{min}) = \left(f_{min} - \underbrace{\frac{\int_{-\infty}^{f_{min}} y\phi_{(\hat{y},\hat{s})}(y)dy}{\Phi_{(\hat{y},\hat{s})}(f_{min})}}_{\overline{y}} \right) \Phi_{(\hat{y},\hat{s})}(f_{min}). \quad (2)$$

[1] For notational simplicity, we omit the dependency of the predictions on $\mathbf{x}$.

[2] In the evolutionary computation community, EGO has become popular under the SPO (Sequential Parameter Optimization) acronym [5].

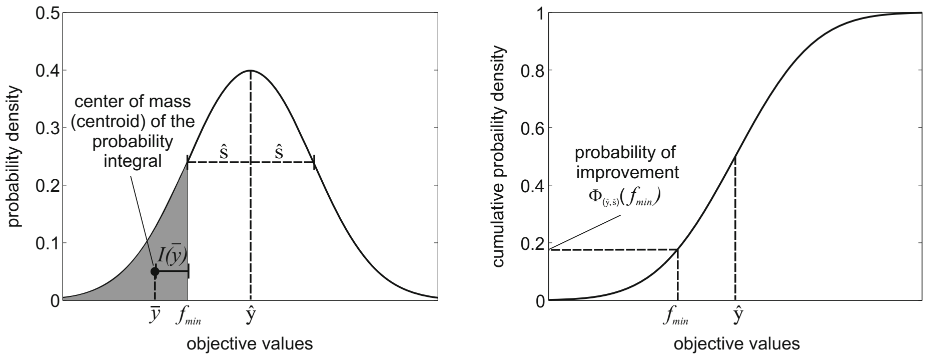


Fig. 1. Graphical explanation of the components $I(\overline{y}, f_{min})$ (left) and $\Phi_{(\hat{y},\hat{s})}(f_{min})$ (right) of the EI definition in equation 2

Consequently, the EI can be regarded as the improvement $I(\overline{y}, f_{min})$ obtained by the center of mass (centroid) $\overline{y}$ of the area under $\phi_{(\hat{y},\hat{s})}$ in the interval $]-\infty, f_{min}]$ weighted with the corresponding CDF $\Phi_{(\hat{y},\hat{s})}(f_{min})$ (cf. Fig. 1).

3 Multi-objective Optimization Based on EI

Over the last decade, a set-based view on multi-objective optimization has been established [6]. According to equation 1, a true multi-objective formulation $EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF})$ requires the PDF of $\mathbf{y}$ and a definition of the improvement $I(\mathbf{y}, \mathbf{A}_{PF})$ of the PF approximation $\mathbf{A}_{PF}$ obtained by a specific candidate vector $\mathbf{y}$. Despite the usually conflicting objectives in EMO, it is common practice [7,8,9,10,11] to make the independence assumption (correlation coefficient $\rho = 0$). Then, the multivariate PDF of $\mathbf{y}$ as $\prod_{i=1}^{m} \phi_{(\hat{y}_i,\hat{s}_i)}(y_i)$ can be directly computed, and the important aspect is the design of an appropriate improvement function.

An overview of recently proposed multi-objective EI definitions is given in Table 1. The acronyms introduced in this table are used throughout the paper. Unfortunately, we cannot describe the approaches due to space requirements. Detailed explanations will be found in the references given in Table 1.

Most of the presented approaches do not directly define a set-based improvement $I(\mathbf{y}, \mathbf{A}_{PF})$. Emmerich [7] proposed SExI – the expected increment of $\mathbf{y}$ to the hypervolume (HV) or $\mathcal{S}$-metric. The HV is the Lebesgue measure of the hyperspace dominated by $\mathbf{A}_{PF}$ and bounded by a reference point $\mathbf{r}$. A closed-form expression for SExI is based on integration over interval boxes determined by the coordinates of the points in $\mathbf{A}_{PF}$ [10]. Independently, Emmerich [7] and Ponweiser et al. [14] have proposed a EI criterion, whose computation is simpler. This measure is the increment of the hypervolume when $\mathbf{y}_{LCB} = \hat{\mathbf{y}} - \alpha\hat{\mathbf{s}}$ (lower confidence bound) is added to $\mathbf{A}_{PF}$. The gain factor α is computed based on a given probability level p as $\alpha(p) = -\Phi^{-1}(0.5 \sqrt[m]{p})$ (in this study $p = 0.5$ is used).

Table 1. Overview of existing multi-objective EI criteria

authors (reference)	acronym	definition of improvement	PDF	direct integration
Knowles [12]	ParEGO	single-objective EI of an augmented Tchebycheff aggregation	univariate	yes
Jeong and Obayashi [13]	EI-EMO	m single-objective EIs	univariate	yes
Liu et al. [9]	WS-EI	sum over single-objective EIs of different weighted sums (WS)	multivariate	partially (only subproblems)
Zhang et al. [11]	TA-EI	maximum over single-objective EIs of different Tchebycheff aggregations (TA)	multivariate	partially (only subproblems)
Keane [8]	Euclid	Euclidean distance to the nearest vector of the PF	multivariate	partially (only PDF)
Ponweiser et al. [14]	SMS-EGO	HV increment to the PF	multivariate	no
Emmerich et al. [7,10]	SExI	HV increment to the PF	multivariate	yes

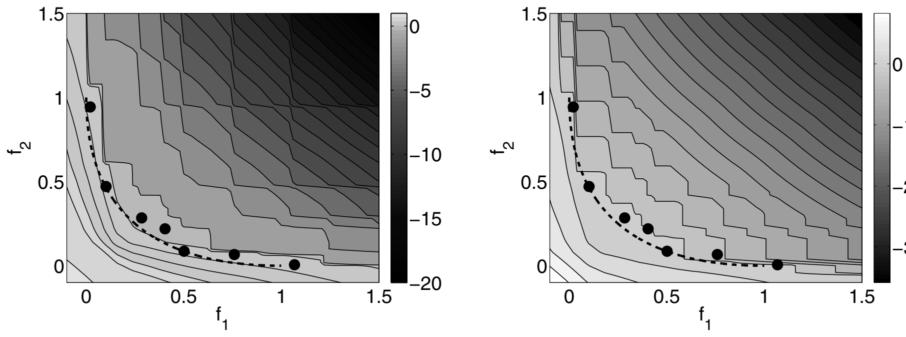


Fig. 2. Comparison of the old (left) and new (right) variant of SMS-EGO. The details of the calculation of the figure is described in section 4.

In order to guide search in dominated regions of the objective space, Ponweiser et al. [14] augmented this criterion by a penalty. In this paper, we introduce a new definition of this penalty. Still, a set of penalties for the ε-dominating solutions $\mathbf{y}^{(i)} \in \mathbf{A}_{PF}$ is computed

$$\Psi(\mathbf{y}_{LCB}) = \begin{cases} -1 + \prod_{j=1}^{m} \left(1 + (y_{LCB,j} - y_j^{(i)})\right) & \text{if } \mathbf{y}^{(i)} \preceq_\varepsilon \mathbf{y}_{LCB} \\ 0 & \text{otherwise} \end{cases} .$$

Whereas we computed the sum over all penalties $\sum \Psi$ in the old version, which resulted in discontinuities of the criterion whenever a dominating solution enters or drops out, we take only the maximum component of Ψ in the new one. This modification leads to a continuous global trend toward $\mathbf{A}_{PF}$ (cf. Fig 2).

4 Analysis and Evaluation

For a formally sound evaluation of multi-objective EI criteria, we propose the following necessary conditions. Given two different predictions of mean vectors $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}'$ and corresponding uncertainties $\hat{\mathbf{s}}$ and $\hat{\mathbf{s}}'$,

N1 the dominance relation between $\hat{\mathbf{y}}$ and $\hat{\mathbf{y}}'$ is preserved by the EI for $\hat{\mathbf{s}} = \hat{\mathbf{s}}'$:
$\hat{\mathbf{y}} \prec \hat{\mathbf{y}}' \wedge \hat{\mathbf{s}} = \hat{\mathbf{s}}' \Rightarrow EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF}) > EI(\hat{\mathbf{y}}', \hat{\mathbf{s}}', \mathbf{A}_{PF})$,
N2 the EI monotonically increases with $\hat{\mathbf{s}}$ for $I(\hat{\mathbf{y}}, \mathbf{A}_{PF}) \leq 0$ and $\hat{\mathbf{y}} = \hat{\mathbf{y}}'$:
$I(\hat{\mathbf{y}}, \mathbf{A}_{PF}) \leq 0 \wedge \hat{\mathbf{y}} = \hat{\mathbf{y}}' \wedge \hat{\mathbf{s}} > \hat{\mathbf{s}}' \Rightarrow EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF}) > EI(\hat{\mathbf{y}}', \hat{\mathbf{s}}', \mathbf{A}_{PF})$,
N3 the EI monotonically increases with $I(\hat{\mathbf{y}}, \mathbf{A}_{PF})$ for $\hat{\mathbf{s}} = \hat{\mathbf{s}}' = \mathbf{0}$:
$I(\hat{\mathbf{y}}, \mathbf{A}_{PF}) > I(\hat{\mathbf{y}}', \mathbf{A}_{PF}) \wedge \hat{\mathbf{s}} = \hat{\mathbf{s}}' = \mathbf{0} \Rightarrow EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF}) > EI(\hat{\mathbf{y}}', \hat{\mathbf{s}}', \mathbf{A}_{PF})$.

The necessary conditions can be analytically checked in most cases and should be considered during the design of a multi-objective EI criterion in order to identify conceptual problems. The restriction to solutions with no improvement in condition N2 and vanishing uncertainties in N3 was made since an increase in $\hat{\mathbf{s}}$ is related to a balancing between risk and opportunity for $I(\hat{\mathbf{y}}, \mathbf{A}_{PF}) > 0$.

Moreover, we compiled a second list, which includes properties that are desired with respect to the internal optimization:

D1 For small $\hat{\mathbf{s}}$ in relation to the range of $\mathbf{A}_{PF}$, a solution should be preferred whose $\hat{\mathbf{y}}$ improves the distribution and/or spread of $\mathbf{A}_{PF}$.
D2 Discontinuities and nondifferentiabilities of the criterion should be avoided, particularly if gradient-based methods are used for the internal optimization.
D3 The fitness landscape of the criterion should guide the optimizer to its global optimum, e. g., plateaus should be avoided and basin sizes should grow with the quality of the corresponding local optimum.
D4 The criterion should be easy to implement and efficient to calculate.

Since the importance of these properties depends on the internal optimization approach and on the application domain, their discussion can assist in choosing the right criterion for a given application.

An overview of the results of our analyses is provided in Table 2. Whenever possible, the necessary conditions N1-N3 were checked analytically.[3] In order to also provide a visual impression of the EI criteria and to allow the assessment of the desirable properties D1-D3, contour plots of the criteria were generated in a bi-objective space – omitting ParEGO and EI-EMO because of the a-priori reduction to the single-objective EI. The contour lines represent the evaluation of different $\hat{\mathbf{y}}$ for constant $\hat{\mathbf{s}}$ using MATLAB® implementations of the criteria based on code of the corresponding authors. The reference set $\mathbf{A}_{PF}$ of size $|\mathbf{A}_{PF}| = 7$ was created by the evaluation of a 65-point Latin Hypercube Design in the domain $[-1, 2]^2$ on the bi-objective generalized Schaffer problem [7] with exponent $\gamma = 0.5$ (convex). The true PF is located within the domain $[0, 1]^2$. In order to analyze the influence of $\hat{\mathbf{s}}$, predictions slightly outside the objective space were also considered. Therefore, the evaluation of the possible predictions $\hat{\mathbf{y}}$ were visualized in the domain $[-0.1, 1.5]^2$ using a constant $\hat{\mathbf{s}} = \mathbf{0.2}$. This relatively high value was chosen because the behavior for low $\hat{\mathbf{s}}$ can be derived analytically in most cases. For the calculation of the indicator-based criteria, the ideal point $\mathbf{i} = (-0.1, -0.1)$ and the reference point $\mathbf{r} = (2, 2)$ were chosen. Consequently,

[3] When not explicitly stated, we omit the special case of $\mathbf{s} = \mathbf{0}$, as it holds only for known evaluations which have a negligibly low probability of being evaluated again.

Table 2. Overview of the compliance of the multi-objective EI criteria with the defined conditions and properties

	ParEGO	EI-EMO	WS-EI	TA-EI	Euclid	SMS-EGO (old)	SMS-EGO (new)	SExI
N1	$\surd^*$	$\surd$	$\surd$	$\surd^*$	$-$	$\surd$	$\surd$	$\surd$
N2	$\surd$	$\surd$	$\surd$	$\surd$	$-$	$\surd$	$\surd$	$\surd^{**}$
N3	$\surd$	$(\surd)$	$-$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
D1	○	$-$	$--$	$+$	$++$	$++$	$++$	$+$
D2	$+$	$++$	$++$	$-$	$-$	$--$	$-$	$++$
D3	○	$+$	$-$	$+$	$+$	$-$	○	$+$
D4	$++$	$+$	$+$	○	$--$	$++$	$++$	$--$

*Only for weight vectors with strictly positive components.
**Empirical evidence, no formal proof could be provided until now.

it is ensured that all evaluated vectors are dominated by $\mathbf{i}$ and dominate $\mathbf{r}$. In practice, this can be accomplished by determining $\mathbf{i}$ and $\mathbf{r}$ by minimizing and maximizing the surrogate model of each objective. Thus, it is also assumed in the proofs of this section. If required, $N = 501$ uniformly distributed weight vectors including $(0, 1)$ and $(1, 0)$ were used. Due to space limitations, only a few of the contour plots can be shown in the paper. All figures computed for this study (also for $\gamma = 1$, $\gamma = 2$, and $\hat{\mathbf{s}} = \mathbf{0.01}$) can be found online.[4]

N1: For Euclid and SExI, $EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF}) = \int_{-\infty}^{\mathbf{r}} I(\mathbf{y}, \mathbf{A}_{PF}) \phi_{\hat{\mathbf{y}},\hat{\mathbf{s}}}(\mathbf{y})\, dy$ holds. By centering the PDF, we get $EI(\hat{\mathbf{y}}, \hat{\mathbf{s}}, \mathbf{A}_{PF}) = \int_{-\infty}^{\mathbf{r}} I(\mathbf{y} + \hat{\mathbf{y}}, \mathbf{A}_{PF}) \phi_{\mathbf{0},\hat{\mathbf{s}}}(\mathbf{y})\, dy$. Thus, N1, assuming equal $\hat{\mathbf{s}}$, is directly related to the compliance of $I(\mathbf{y}, \mathbf{A}_{PF})$ with the dominance relation. This relation also holds for $I(\mathbf{y}_{LCB}, \mathbf{A}_{PF})$ because the constant displacement $\alpha\hat{\mathbf{s}}$ can be neglected. Whereas the HV used in SExI and SMS-EGO is Pareto-compliant [15], the Euclidean distance is not (cf. Fig. 3).

All other approaches directly use $I(y, f_{min})$ of the single-objective EI. Thus, their compliance with N1 is related to the preprocessing before the EI computation. Both, the TA and the WS, are compliant with the dominance relation as long as no component of the weight vector is zero [16]. In this case, an improvement in the objective with the zero component is not reflected in the scalarization (cf. Fig. 4 (left) for $f_1 = 0$ or $f_2 = 0$). Consequently, ParEGO and TA-EI are only compliant with N1 if no such weight vectors are used. WS-EI takes the sum over the EI of all weight vectors. Thus, at least one weight vector with a positive component for each objective is required, which is very likely to be fulfilled. Despite the a-posteriori selection of the extreme solutions, all single-objective EIs are considered during EI-EMO. Thus, N1 holds for this criterion.

N2: It is has been shown by Jones et al. [4, pp. 172f.] that a higher $\hat{s}$ monotonically improves the single-objective EI, even when $I(\hat{y}, f_{min}) > 0$. Based on this result, N2 is fulfilled for ParEGO, EI-EMO, and all subproblems of TA-EI and WS-EI, which directly transfers to the final aggregation. Since the $\mathbf{y}_{LCB}$ is linearly improved by $\hat{\mathbf{s}}$, N2 also holds for both variants of SMS-EGO.

[4] http://www.pbase.com/emmerich/expected_improvement

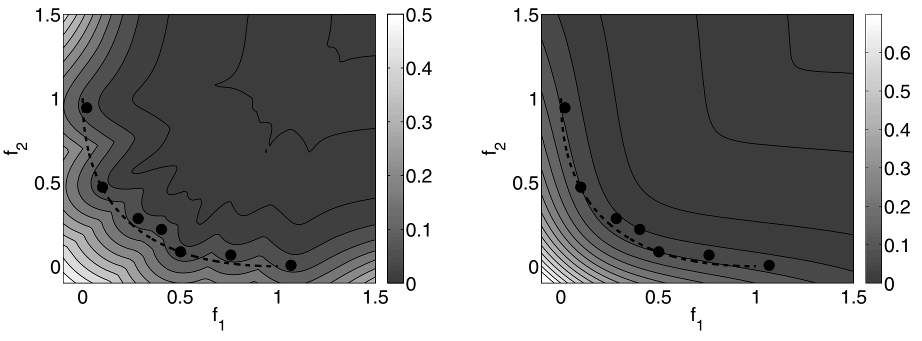


Fig. 3. Comparison of Euclid (left) and SExI (right)

For Euclid and SExI, we conducted an experiment, in which $\hat{s} = 0, 0.1, \ldots, 1$ were evaluated for each $\hat{y}$ of Fig. 2-4. For SExI no counterexample was found, but Euclid violated N2 in 797 of 4925 cases. This violation is often caused by a reduced minimum distance due to a movement of the centroid from the dominated to the nondominated area. The results of SExI provide empirical evidence for a compliance with N2, but no formal proof could be provided until now.

N3: SMS-EGO, SExI, and Euclid fulfill N3 by definition. If $\hat{\mathbf{s}} = \mathbf{0}$, no displacement of $\mathbf{y}_{LCB}$ occurs or the PDF becomes singular. This results in a direct evaluation of $I(\hat{\mathbf{y}}, \mathbf{A}_{PF})$. In ParEGO and EI-EMO, the single-objective EI is evaluated. Therefore, N3 also holds for the considered subproblems. The center solution of EI-EMO, however, is not related to a clearly formulated improvement, which does not allow a complete evaluation of this approach.

Given the final decision making, applied in TA-EI and WS-EI, both aim for a maximum improvement, either of a single subproblem (TA-EI) or of the sum over all subproblems (WS-EI). However, the separated computation of EIs and the subsequent aggregation is only straightforward for maximizing the improvement on a single subproblem. In WS-EI, the sum of the EIs substitutes the EI of the sum. Since the EI nonlinearly depends on $\hat{y}$ and $\hat{s}$, N3 is violated. In order to calculate the actual EI, the mean and the standard deviation of the sum of scalarizations have to be computed. To accomplish this, the equations for calculating each $\hat{y}_{sc}$ and $\hat{s}_{sc}$ can be applied again.

D1: It has been shown that the maximization of the HV increment produces well-distributed sets [7]. Given that $\mathbf{r}$ is sufficiently far away from $\mathbf{A}_{PF}$, the spread is also improved [17]. Therefore, all criteria based on the HV cope with D1 for sufficiently small $\hat{\mathbf{s}}$ (cf. N3). However, a comparison of Fig. 2 and Fig. 3 (right) reveals that the gap-filling property of the SExI fades away with increasing $\hat{\mathbf{s}}$ whereas it is conserved for SMS-EGO. This is caused by the fact that samples from $\mathcal{N}(\hat{\mathbf{y}}, \hat{\mathbf{s}})$ can improve the distribution or spread of $\mathbf{A}_{PF}$, even if $\hat{\mathbf{y}}$ does not improve it. Moreover, this property enhances the guidance to the most promising local optima, as discussed for D3. Since the maximization of the Euclidean distance to the neighboring solution is an established diversity-measure,

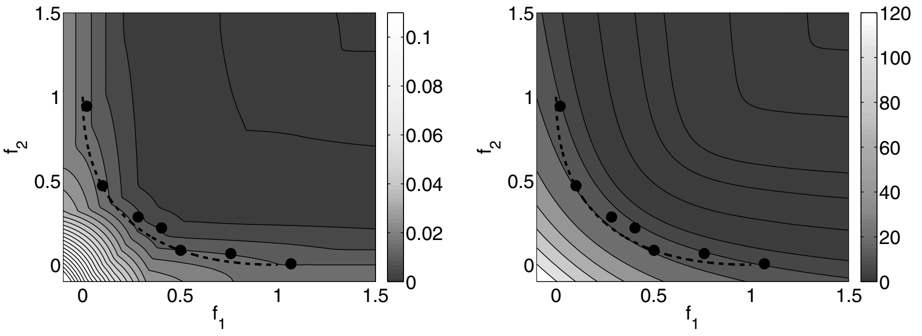


Fig. 4. Comparison of TA-EI (left) and WS-EI (right)

Euclid also copes with D1. Contrary to the direct approach, this also holds for high $\hat{\mathbf{s}}$ as shown in Fig. 3 (left).

For the scalarization-based approaches, only TA-EI copes with D1. As shown in Fig. 4 (left), the contour lines indicate the improvement by filling the gaps in the upper left part of $\mathbf{A}_{PF}$. Compared to TA-EI, which evaluates all weight vectors in each iteration, ParEGO randomly chooses a weight vector which may target toward an already crowded region of $\mathbf{A}_{PF}$, deteriorating its compliance with D1. The WS-EI is generally biased to the knee (convex) or to the extremes (concave) of $\mathbf{A}_{PF}$. The maximization of the sum of EIs produces an additional bias toward the center of the targets defined by the weight vectors (cf. Fig. 4, right). The a-posteriori selection of EI-EMO exclusively focuses on the extremes and the center of the EI Pareto front. Thus, only the spread of $\mathbf{A}_{PF}$ will be improved. A good distribution between the extremes cannot be accomplished.

D2: Only SExI, WS-EI and EI-EMO are continuous and differentiable over the whole domain. TA-EI and Euclid use maximum or minimum operations which lead to nondifferentiabilities of the corresponding criterion (cf. the left plots of Fig. 3 and Fig. 4). In ParEGO, the nondifferentiabilities are smoothed out by the surrogate model, making the actual EI criterion continuous and differentiable.

The problem of discontinuities in the old SMS-EGO approach and the answer of the new one has already been described in section 3. However, the nondifferentiabilities at the corners of the attainment surface of $\mathbf{A}_{PF}$ could not be resolved. This is shown in Fig. 2 by the contour lines in the proximity of $\mathbf{A}_{PF}$.

D3: All approaches relying on an EI formulation without penalties can show plateaus of zero EI based on the limited machine accuracy. This problem is overcome by the penalty functions used in SMS-EGO. Nevertheless, the fitness landscapes of SExI and TA-EI are evaluated best since these approaches show strong gradients to their local optima. This is shown in Fig 3 (right) and Fig. 4 (left). In contrast, the gradients in the landscape of SMS-EGO are very local, making the search for the global optimum difficult (cf. Fig. 2). The approach of Keane shows the most complex fitness landscape with many local optima. Nevertheless, the gradients to each local optimum are clearly defined even far from those, and the size of each basin grows with quality of the optimum (cf. Fig. 3, left). WS-EI fails to indicate the direction toward the optimum in the knee of $\mathbf{A}_{PF}$ by providing gradients that are normal to the true PF. In ParEGO, the original EI (equation 1) has to be optimized, which is known for multimodality and plateaus. For EI-EMO these problems are relaxed because the multi-objective optimization of the different EIs enhances diversity and avoids the premature convergence to one of the local optima of the single-objective EI.

D4: Besides the approaches based on a piecewise integration over the nondominated region which require a tedious partitioning of the objective space, all EI criteria are easy to implement. For ParEGO, only one model has to be computed making it the fastest of all approaches. The multi-objective optimization in EI-EMO slightly increases the runtime compared to the single-objective optimizations performed in all other approaches.

Regarding the empirical runtime for computing the figures in bi-objective space, SMS-EGO and Keane are the fastest approaches (≈ 4 s for 6561 evaluations). The scalarization-based EI criteria show a surprisingly high runtime for the recommended number of $N = 501$ weight vectors (Tchebycheff: 978 s, weighted sum: 398 s), deteriorating their rating in D4. The direct integration takes about 140 s for the bi-objective computations. However, the runtime of the SExI and Euclid may increase exponentially with the number of the objectives m.

5 Conclusions and Outlook

In this paper, we summarized, compared, and analyzed existing EI criteria for multi-objective optimization. For one of the criteria, an improved variant has been introduced. Moreover, we proposed necessary conditions and desired properties for a formal evaluation. Based on theoretical and empirical analyses, we showed that Euclid and WS-EI are not compliant with the dominance relation or provide no clear formulation of the desired improvement. Thus, these approaches should no longer be used. All other approaches considered in this study fulfill the necessary conditions. Depending on the application, the appropriate criterion can be chosen based on the performance on the desired properties (cf. Table 2).

For the scalarization-based approaches, improvements in the formulation and requirements for the weight vectors could be stated. However, a trade-off between accuracy and runtime still exists. To overcome this problem, an adaptive calculation of the corresponding optimal weight vector [16] during the integration seems promising. The problem of plateaus in the EI landscapes may be solved by combining the EI with penalty functions for values below the machine accuracy.

Acknowledgments

This paper is based on investigations of the collaborative research center SFB/TR TRR 30, which is kindly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. Knowles, J., Nakayama, H.: Meta-modeling in multiobjective optimization. In: Branke, J., et al. (eds.) Multiobjective Optimization – Interactive and Evolutionary Approaches, pp. 461–478. Springer, Berlin (2008)
2. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. 4(4), 409–435 (1989)
3. Mockus, J.B., Tiesis, V., Zilinskas, A.: The application of bayesian methods for seeking the extremum. In: Dixon, L.C.W., Szegö, G.P. (eds.) Towards Global Optimization, vol. 2, pp. 117–129. Amsterdam, New York (1978)
4. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Glob. Optim. 13(4), 455–492 (1998)
5. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: McKay, B., et al. (eds.) Proc. CEC, pp. 773–780. IEEE, Los Alamitos (2005)
6. Zitzler, E., Thiele, L., Bader, J.: On set-based multiobjective optimization. IEEE Trans. Evol. Comput. 14(1), 58–79 (2010)
7. Emmerich, M.: Single- and Multi-objective Evolutionary Design Optimization Assisted by Gaussian Random Field Metamodels. PhD thesis, Universität Dortmund (2005)
8. Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA J. 44(4), 879–891 (2006)
9. Liu, W., Zhang, Q., Tsang, E., Liu, C., Virginas, B.: On the performance of metamodel assisted MOEA/D. In: Kang, L., Liu, Y., Zeng, S., et al. (eds.) ISICA 2007. LNCS, vol. 4683, pp. 547–557. Springer, Heidelberg (2007)
10. Emmerich, M., Deutz, A.H., Klinkenberg, J.W.: The computation of the expected improvement in dominated hypervolume of pareto front approximations. Technical Report 4-2008 Leiden Institute of Advanced Computer Science, LIACS (2008), http://www.liacs.nl/~emmerich/TR-ExI.pdf
11. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive multiobjective optimization by MOEA/D with gaussian process model. IEEE Trans. Evol. Comput. (2010); Early Access (will be published)
12. Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. Evol. Comput. 10(1), 50–66 (2006)
13. Jeong, S., Obayashi, S.: Efficient global optimization (EGO) for multi-objective problem and data mining. In: Corne, D., et al. (eds.) Proc. CEC, pp. 2138–2145. IEEE, Los Alamitos (2005)
14. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited amount of evaluations using model-assisted $\mathcal{S}$-metric selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 784–794. Springer, Heidelberg (2008)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Trans. Evol. Comput. 7(2), 117–132 (2003)
16. Steuer, R.E.: Multiple Criteria Optimization: Theory, Computation, and Application. Wiley, New York (1986)
17. Auger, A., Bader, J., Brockhoff, D., Zitzler, E.: Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point. In: Garibay, I., et al. (eds.) Proc. FOGA, pp. 87–102. ACM, New York (2009)

Parameter Tuning Boosts Performance of Variation Operators in Multiobjective Optimization

Simon Wessing, Nicola Beume, Günter Rudolph, and Boris Naujoks

Fakultät für Informatik, Technische Universität Dortmund, Germany
{simon.wessing,nicola.beume,guenter.rudolph,boris.naujoks}@tu-dortmund.de

Abstract. Typically, the variation operators deployed in evolutionary multiobjective optimization algorithms (EMOA) are either simulated binary crossover with polynomial mutation or differential evolution operators. This empirical study aims at the development of a sound method how to assess which of these variation operators perform best in the multiobjective context. In case of the S-metric selection EMOA our main findings are: (1) The performance of the tuned operators improved significantly compared to the default parameterizations. (2) The performance of the two tuned variation operators is very similar. (3) The optimized parameter configurations for the considered problems are very different.

Keywords: parameter tuning, performance assessment, benchmarking, multiobjective variation operators, sequential parameter optimization.

1 Introduction

Numerous multiobjective evolutionary algorithms have been developed and studied in various benchmarks. However, clear evidence on which methods, operators, and even parameters are promising for certain test cases could not be received. Here, the functions from the well-known CEC 2007 [1] competition are dealt with investigating the performance of different variation operators for one special algorithm. The aim is to either propose one promising setting for this special scenario or to empirically prove that such settings differ along the considered test functions and operators.

The operators polynomial mutation (PM) and simulated binary crossover (SBX) devised by Deb et al. [2] are standard, have been incorporated in many algorithms, and considered in just as many benchmarks. In the competition on multiobjective optimization at the CEC 2007 [1], algorithms using differential evolution (DE) were among the best especially on the alterations of standard test functions and thereby recommend themselves. These variation operators are plugged into the SMS-EMOA [3], which is compliant and well working with both variation concepts as well as the test cases considered.

We aim at performing this study as professional as possible with state-of-the-art methodologies of experimental research, so that it may serve as a prototypic example to gain a very deep understanding of the object of investigation. In the past, experimental studies were mostly set up such that several algorithms are compared based on their default parameters (performing with unknown quality) or equal parameter values, e.g. for the population size. But a certain parameter value can of course be more suitable for one algorithm than for another. According to Bartz-Beielstein [4], a comparison of algorithms is fair only if these are both set up with optimal parameter configurations with respect to the optimization problem. To comply with this mindset, we use Sequential Parameter Optimization (SPO) [4] to find good configurations before comparing the performance. The tuning is an optimization problem with unknown optimum and better parameterization are likely to exist. Since we can neither determine the optimum nor prove the optimality of a configuration, the chosen methodology seems to be the best way to proceed.

Wessing and Naujoks [5] compared the established performance indicators I_H, I_{R2} and $I_{\epsilon+}$ [6] in combination with SPO, finding out that the hypervolume indicator I_H is the most suitable one for a comparison. For this reason, we are focusing on I_H exclusively. From the CEC 2007 testbed we have chosen the two-objective problems OKA2, SYM-PART, the shifted as well as rotated variants of ZDT problems, and three-objective DTLZ and WFG problems. Note that the results are not directly comparable to others obtained before in the environment of the CEC 2007 contest, since a number of bugs in the implementation of the benchmark have been fixed. In a second experiment, two aerodynamic test cases are analyzed.

The next section details the invoked algorithm and variation operators as well as the parameter tuning tool SPO. Section 3 contains our experimental studies and we summarize our work in Sec. 4.

2 Preliminaries

The SMS-EMOA [3] is an indicator-based steady-state algorithm which performs its selection such that the hypervolume dominated by the population in the objective space is maximized. Thereby, the algorithm aims at converging towards a good distribution along the Pareto front. Its conception does not include any prescribed variation operator but it has mostly been studied with PM and SBX.

Recall that in single-objective optimization the CMA-ES [7] is unchallenged as the most successful variation operator on most problems. However, this question is not settled yet in multi-objective optimization (MOO), because of the different requirements. The aim in MOO is not to converge to a single global optimum, but to approximate the whole Pareto-front, which requires appropriate diversity in the population.

Simulated Binary Crossover (SBX) was devised by Deb et al. [2] to carry over the behavior of single-point crossover in binary search spaces to real valued search spaces. It always creates two children from two parents. Polynomial Mutation (PM) utilizes the same probability distribution to vary a single individual. These two variation operators together will simply be called *SBX variation* in the remainder of the paper. They contain several parameters to be adjusted by the user. The variance of the distributions is controlled by the parameters $\eta_c, \eta_m \in \mathbb{R}_+$. For recombination and mutation the parameters p_c and p_m, respectively, describe the probability for each position in the genome to apply variation. This means that the impact of the η values is directly depending on the probabilities.

Another variation method that copes well with $(\mu+1)$ selection scheme of the chosen EMOA is Differential Evolution (DE), developed by Storn and Price [8]. The classic DE algorithm contains a special selection scheme, which lets the offspring only compete with its parent, achieving a crowding effect. However, only the DE variation is picked here to be used with the SMS-EMOA. We choose to focus on the two user-adjustable parameters F and CR for optimization. F is a scaling factor to vary the length of the difference vectors and CR controls the crossover rate, similar to p_c in SBX. In this work, we employ the plain SBX and DE versions that were originally proposed by their inventors.

The main idea of SPO is to treat optimizer runs as experiments, using methods from Design of Experiments (DoE) [9] and Design and Analysis of Computer Experiments (DACE) [10]. The optimizer's exogenous parameters are considered as the experiment's design variables. SPO begins with a latin hypercube sample (LHS) in the search space and creates a surrogate model from the results. In our case, DACE Kriging [11] is used for modeling. As the optimizer's results are stochastic, each point is sampled several times and the results are averaged. In an optimization loop, the model is then used to predict promising parameter configurations. The new candidates are evaluated and the data is fed back into the model. If no new best configuration is found in a step, the number of repetitions is increased.

3 Experiments

The first experiment investigates the performance of the algorithms and methodologies above on a set of well known mathematical test functions. A second experiment deals with two real-world problems.

3.1 DE vs. SBX on CEC 2007 Problems

Research Question: How does DE compare to SBX variation on the CEC 2007 test case collection [1]?

Table 1. The default values and region of interest (ROI) of parameters. The ROI is the range on which the search is conducted.

	DE			SBX				
Param.	μ_{DE}	CR	F	μ_{SBX}	η_c	η_m	p_c	p_m
Default	100	0.1	0.5	100	20.0	15.0	1.0	0.1
ROI	$\{6, \dots, 120\}$	[0, 1]	[0, 2]	$\{3, \dots, 120\}$	$[0, 40]$	$[0, 40]$	$[0, 1]$	$[0, 1]$

Preexperimental planning: For the experiment's preparation, some SPO runs were carried out to determine the parameters' regions of interest (see Tab. 1). Additionally, the optimization of SBX configurations on OKA2 and S_ZDT2 with 1000 problem evaluations was repeated 20 times, to get an estimate of SPO's reliability. It is not necessary that SPO always delivers the same parameterization as the optimized one, because not all parameters have influence on the performance. But it is desired that an algorithm set up with the final parameterization achieves Pareto front approximations of similar quality. Fig. 1 shows that the performance could be increased in all cases and we regard the variance as small enough for meaningful comparisons, even when only one SPO run is performed per problem. The tuned parameter configurations will be called DE* and SBX* in the remainder.

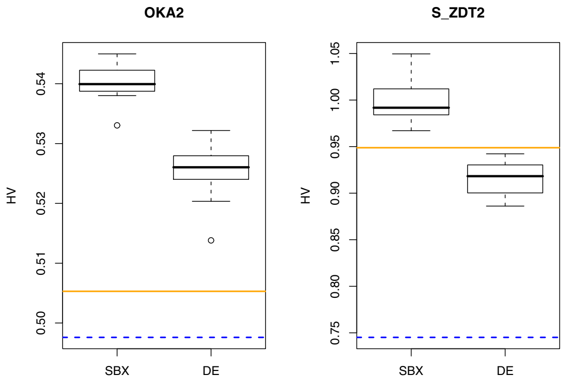


Fig. 1. Boxplots show the performance distributions in terms of dominated hypervolume (HV) of 20 SPO runs. Additional lines mark the default SBX (solid) and DE (dashed) configurations' mean performance.

Task: After SPO has finished, the new configurations are run 50 times and evaluated with I_H. These samples are compared to same-sized samples of the default configurations and each other. For each comparison, a two-sided U-Test [12] is employed. The null hypothesis is that there is no difference in means and we require a significance level of 5% to reject it.

Setup: SPO is applied to all 2-objective and 3-objective test problems in the CEC 2007 suite. The contained 5-objective problems are excluded, because of the SMS-EMOA's high runtime on these. Two different run lengths, namely $500 \cdot M$ and $5000 \cdot M$ function evaluations, of the SMS-EMOA are examined to detect possible *floor* or *ceiling effects*. Here, M denotes the number of the problem's objectives. Tables 1 and 2 show the regions of interest and the setup for the experiments. The default parameters for DE variation are chosen according to [13]. DE's lower bound for μ is higher than that for SBX, because it uses more parents for variation. The performance evaluation is generally done according to the CEC 2007 contest rules [1], i.e the whole objective space of each problem is approximately normalized to $[1, 2]^M$. The reference point for I_H is then set to

Table 2. The setup for Experiment 3.1

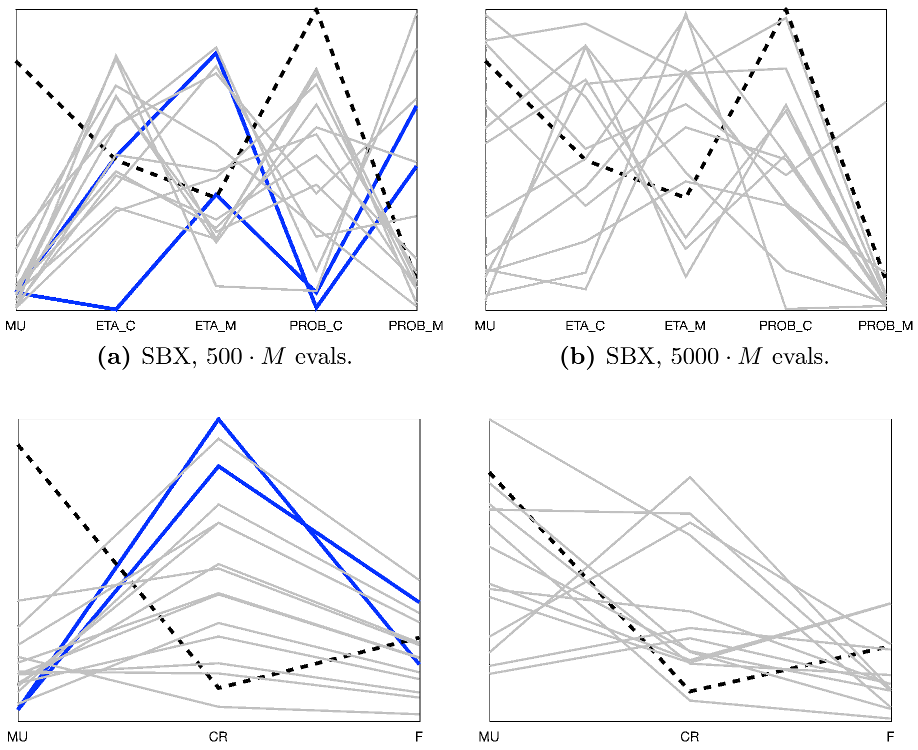
Problems	Two- and three-objective CEC 2007 problems
SPO budget	500 algorithm runs
Algorithm initialization	Uniform random
Stopping criterion	$500 \cdot M$ and $5000 \cdot M$ problem evaluations
Algorithm	SMS-EMOA
Parameters	DE: μ, CR, F; SBX: μ, η_c, η_m, p_c, p_m
Initial experimental design	Latin Hypercube (50 points, 3 repeats per point)
Performance measure	I_H

$(2.1, \ldots, 2.1)^T$, although Wessing and Naujoks [5] show that the whole approach can have drawbacks on some problems.

Results/Visualization: Tables 3 and 4 show the performance results of DE and SBX variation. Optimized configurations that are significantly better than the competing optimized configuration are highlighted in bold face. Figure 2 shows parallel plots of the configurations. More details on the parameter configurations are provided by Wessing [14].
Observations: SBX reaches significantly better mean values than DE for all test cases. For $500 \cdot M$ problem evaluations, SBX* is better than DE* on ten problems, while the opposite is true on only two problems (there is one tie). For $5000 \cdot M$ evaluations, SBX* wins seven times and DE* four times (there are two ties). Except for SBX* on R_ZDT4, both operators can always improve significantly compared to their default configurations. Figure 2 shows that small population sizes should be used on the short runs. Especially for long runs, low values of p_m are a good choice. It also seems to be promising to choose $CR > F$. The rest of the parameters does not follow any general trend.

Table 3. Mean hypervolume and standard deviation with $500 \cdot M$ evaluations

Problem	SBX	SBX*	DE	DE*
OKA2	0.5053 ±0.013	**0.5450** ±0.011	0.4976 ±0.013	0.5322 ±0.021
SYM-PART	1.1640 ±0.009	**1.2063** ±0.001	1.0335 ±0.023	1.1935 ±0.011
S_ZDT1	1.0189 ±0.020	**1.1024** ±0.015	0.8706 ±0.019	1.0403 ±0.026
S_ZDT2	0.9488 ±0.023	**1.0496** ±0.042	0.7451 ±0.028	0.9422 ±0.033
S_ZDT4	0.9505 ±0.027	1.0407 ±0.043	0.8489 ±0.025	1.0546 ±0.027
R_ZDT4	1.0994 ±0.018	1.1214 ±0.039	1.0605 ±0.022	**1.1350** ±0.024
S_ZDT6	0.7340 ±0.011	**0.7592** ±0.016	0.6573 ±0.007	0.7013 ±0.011
S_DTLZ2	1.3270 ±0.001	**1.3291** ±0.002	1.3189 ±0.003	1.3290 ±0.001
R_DTLZ2	1.3100 ±0.009	1.3204 ±0.008	1.2531 ±0.024	**1.3243** ±0.003
S_DTLZ3	1.3165 ±0.003	**1.3304** ±0.001	1.3054 ±0.005	1.3257 ±0.003
WFG1	0.9051 ±0.005	**0.9936** ±0.005	0.8677 ±0.012	0.9084 ±0.008
WFG8	1.1101 ±0.014	**1.2231** ±0.009	1.1062 ±0.015	1.1985 ±0.009
WFG9	1.1651 ±0.020	**1.2147** ±0.013	1.1474 ±0.024	1.2012 ±0.019



(a) SBX, $500 \cdot M$ evals.
(b) SBX, $5000 \cdot M$ evals.
(c) DE, $500 \cdot M$ evals.
(d) DE, $5000 \cdot M$ evals.

Fig. 2. Parallel plots of best parameter configurations found by SPO. Parameters for all 13 test functions are shown in light gray. Default configurations are marked as dashed lines. Results from Exp. 3.2 are shown as dark bold lines.

Table 4. Mean hypervolume and standard deviation with $5000 \cdot M$ evaluations

Problem	SBX	SBX*	DE	DE*
OKA2	0.5610 ±0.010	**0.5725** ±0.011	0.5480 ±0.008	0.5676 ±0.010
SYM-PART	1.2074 ±3.0e-4	1.2095 ±1.4e-4	1.1194 ±0.012	**1.2098** ±4.0e-5
S_ZDT1	1.1508 ±0.004	1.1684 ±0.005	0.8755 ±0.012	**1.1686** ±0.002
S_ZDT2	1.0668 ±0.006	**1.1271** ±0.009	0.7394 ±0.023	1.0928 ±0.018
S_ZDT4	1.1241 ±0.015	**1.2029** ±0.003	0.8441 ±0.020	1.1654 ±0.011
R_ZDT4	1.1908 ±0.010	1.1933 ±0.008	1.0845 ±0.015	1.1923 ±0.006
S_ZDT6	0.8658 ±0.007	0.9293 ±0.024	0.6584 ±0.004	**0.9574** ±0.016
S_DTLZ2	1.3301 ±1.1e-5	1.3302 ±4.0e-6	1.3224 ±5.7e-4	1.3302 ±1.6e-5
R_DTLZ2	1.3296 ±3.2e-5	**1.33000** ±3.2e-5	1.2638 ±1.3e-2	1.32997 ±4.2e-5
S_DTLZ3	1.3306 ±2.3e-4	**1.33099** ±1.4e-5	1.3028 ±2.2e-3	1.33097 ±2.1e-5
WFG1	0.9684 ±0.003	**1.0803** ±0.017	0.8929 ±0.005	1.0633 ±0.025
WFG8	1.2197 ±0.005	**1.2702** ±0.003	1.2003 ±0.046	1.2615 ±0.003
WFG9	1.2459 ±0.007	1.2592 ±0.008	1.1894 ±0.007	**1.2607** ±0.010

Discussion: The experiment shows that the decision which variation is chosen is less important than the decision to tune the chosen variation operator, because the differences between the default and optimized configurations are much bigger than between different optimized configurations. It is also obvious that the default setting is completely opposing the optimal configuration on some problems. SBX* winning more often might be due to a biased set of problems. The result is more balanced on the longer runs, so it would be interesting to test if DE* performance increases for run lengths extended even further.

3.2 DE vs. SBX on Aerodynamic Problems

From the academic test cases we have learned that optimal parameterizations differ considerably. As a consequence, there is no general near-optimal default parameterization. However, this could be due to artificial structures of the academic test cases. Therefore, the next experiment shall reveal whether our insights gained so far are transferable to real-world problems.

Research Question: How does DE compare to SBX variation on examples of real-world problems? How do the results compare to the ones from the previous experiment?

Preexperimental planning: We consider a 2-objective and a 3-objective aerodynamic problem, which have been subject of previous studies [15,16]. Due to the large calculation times for the computational fluid dynamics simulations, a restricted number of 1000 objective function evaluations is allowed and the SPO budget is slightly decreased to 300 algorithm runs (see Tab. 5).

Task: See Experiment 3.1.

Setup: In the first investigation, a two-objective airfoil design problem is considered (referred to as NACA, cf. [15]). Two regimes of flow conditions have been chosen, which vary in the flow parameter settings. Practitioners are interested in good compromise solutions ranging from considering mainly the first flow condition to the other way around. This way, a Pareto front according to this trade-off is highly appreciated. To achieve this Pareto front, two nearly optimal airfoils have been identified to become target airfoils, and a two-objective redesign test case is defined.

The second aerodynamic test case is a true design test case (referred to as RAE, cf. [16]). Here, the drag for some given airfoil is to be minimized for three different flow conditions. Different constraints were defined to guarantee for a minimum of structural feasibility of the received results. These constraints were

Table 5. Settings for Experiment 3.2 that differ from Tab. 2

Problems	NACA, RAE
SPO budget	300 algorithm runs
Stopping criterion	1000 problem evaluations
Initial experimental design	Latin Hypercube (25 points, 4 repeats per point)

of geometric, as well as of aerodynamic nature. Both kind of constraints were handled in different ways as can be gleaned from [16]. The baseline RAE 2822 design is always included in the initial population. The airfoil parametrization is done using Bezier points like for the NACA test case. While 18 degrees of freedom were allowed to control the airfoil in the NACA test case, the RAE test case features only six.

The default configurations and regions of interest are identical to those in Experiment 3.1 (see Tab. 1). Table 5 shows the differences in the experimental setup compared to Experiment 3.1. The reference point is set to $(0.4, 0.4)^T$ for the NACA problem and $(10, 10, 10)^T$ for the RAE problem.

Results/Visualization: Table 6 shows the found optimized configurations, which are also included in Fig. 2 as bold lines. Table 7 shows the performance results.

Table 6. Parameter results on the aerodynamic problems

	DE Configuration			SBX Configuration				
Problem	μ_{DE}	CR	F	μ_{SBX}	η_c	η_m	p_c	p_m
NACA	21	0.90	0.34	10	0.16	15.43	0.06	0.68
RAE	14	0.76	0.71	10	20.50	34.24	0.01	0.48

Table 7. Mean hypervolume and standard deviation on aerodynamic problems

Problem	SBX	SBX*	DE	DE*
NACA	0.1462 ± 0.0012	0.1501 ± 0.0007	0.1467 ± 0.0009	0.1502 ± 0.0007
RAE	993.663 ± 0.005	993.844 ± 0.022	993.672 ± 0.033	**993.869** ± 0.041

Observations: All optimized configurations are significant improvements over their default configurations. The difference between SBX* and DE* is not significant on NACA, but on RAE. The possible improvements by parameter tuning can be gleaned from Tab. 7: SPO is able to improve the NACA values by 2.7% using SBX* and 2.4% featuring DE*. However, the results on RAE cannot be improved accordingly, here the improvements are about 0.02%.

Interestingly, the same population size is identified for SBX variation on both test cases. For DE*, a roughly similar population size was identified for the RAE case as well, while the best value for the NACA case is twice as big. Concerning the operators' probabilities, SBX* variation focuses on mutation. The application probabilities for the recombination operator are very small, which means that η_c cannot have much influence. Generally, it is remarkable that SBX* and DE* are completely opposed to the default configurations.

Discussion: The improvement seems so low on RAE, because the initial population always contains the mentioned near-optimal baseline solution, which already dominates a hypervolume of 993.662. But in fact, the default SBX configuration fails to find any other feasible solution in 49 of the 50 runs. The default DE configuration 'only' fails in 39 runs. DE* and SBX*, on the other hand, achieve success rates of 100% for this measure.

4 Conclusions

On real-world problems, it is still common practice to use EMOA parameterizations obtained from unrelated test problems. Our experiments clearly put this into question. This work shows that the performance of the tuned operators improved *significantly* compared to the default parameterizations. Moreover, the performance of the two tuned variation operators is very similar, whereas the optimized parameter configurations for the considered problems are very different. As a consequence, parameter tuning should become standard. While we acknowledge that the proposed parameter optimization is computationally very expensive, ignoring the problem is not an option, because significantly improved solutions can be obtained. Moreover, Preuss et al. [17] outline a possible remedy by replacing the original, expensive problem by a surrogate model. An EA tuned on the surrogate yields a parameter configuration that performs better than the default parameterization on the original problem.

From the practitioners point of view there is a clear message: Regardless which variation operator is eventually chosen, make sure that the parameters are tuned. Do not trust in default settings! This is also the result of Smit and Eiben [18], who studied parameter tuning on single-objective problems. Thus, we recommend that publications include information on how much effort was put into finding any parameter values. Also, benchmarking contests should add rules for dealing with parameters. For example, the parameter tuning can be regarded as part of the problem, i.e. parameter tuning has to be performed within given budget restrictions. At least, it is not fair to compare algorithms with default or equal parameterizations.

References

1. 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 of multi-objective optimization algorithms. Technical report, Nanyang Technological University, Singapore (2007), http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-07/CEC07.htm
2. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
3. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)
4. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation - The New Experimentalism. Natural Computing Series. Springer, Berlin (2006)

5. Wessing, S., Naujoks, B.: Sequential Parameter Optimization for Multi-Objective Problems. In: Congress on Evolutionary Computation (CEC) within the World Congress on Computational Intelligence, WCCI, pp. 4063–4070 (2010) (accepted for publication)
6. Knowles, J.D., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich (2005)
7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
8. 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)
9. Montgomery, D.C.: Design and Analysis of Experiments, 4th edn. Wiley, New York (1997)
10. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–423 (1989)
11. Lophaven, S.N., Nielsen, H.B., Søndergaard, J.: DACE – A Matlab Kriging Toolbox. Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark (2002)
12. Hollander, M., Wolfe, D.A.: Nonparametric Statistical Methods. John Wiley & Sons, New York (1973)
13. Kukkonen, S., Lampinen, J.: Performance assessment of generalized differential evolution 3 (GDE3) with a given set of problems. In: Congress on Evolutionary Computation (CEC), pp. 3593–3600. IEEE Press, Piscataway (2007)
14. Wessing, S.: Towards optimal parameterizations of the S-metric selection evolutionary multi-objective algorithm. Algorithm Engineering Report TR09-2-006, Technische Universität Dortmund (2009)
15. Naujoks, B., Willmes, L., Bäck, T., Haase, W.: Evaluating multi-criteria evolutionary algorithms for airfoil optimisation. 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. 841–850. Springer, Heidelberg (2002)
16. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multi-objective evolutionary optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421–439 (2006)
17. Preuss, M., Rudolph, G., Wessing, S.: Tuning optimization algorithms for real-world problems by means of surrogate modeling. In: Genetic and Evolutionary Computation Conference, GECCO, pp. 401–408 (2010) (accepted for publication)
18. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 399–406. IEEE Press, Los Alamitos (2009)

Author Index

Agapitos, Alexandros I-294
Aguirre, Hernán I-657, I-677, II-11
Ahmed, Faraz I-304
Akbarzadeh, Vahab II-141
Akimoto, Youhei I-154
Allmendinger, Richard II-151, II-161
Al Moubayed, Noura II-1
Alt, Leonardo S. II-381
Amaya, Jhon Edgar I-445
Arabas, Jarosław I-114, II-411
Arenas, Maria I. García II-341, II-452
Arias Montaño, Alfredo II-21
Arnold, Dirk V. I-11
Auger, Anne I-11, I-52, I-586
Aulig, Nikola II-71
Azzini, Antonia I-344

Bäck, Thomas I-63, I-214
Bader, Johannes I-586, I-707
Bannenberg, Tobias II-240
Bartha, Attila II-432
Beluch, Witold II-171
Bender, Axel I-284
Bentley, Peter J. I-434
Bernardino, Anabela Moreira II-229
Bernardino, Eugénia Moreira II-229
Beume, Nicola I-597, I-728
Birattari, Mauro II-331
Bischl, Bernd I-314
Bogdański, Marcin I-526
Borgulya, Istvan II-280
Borschbach, Markus II-442
Böttcher, Süntje I-1
Bouvry, Pascal II-320
Brabazon, Anthony I-294
Bredeche, Nicolas II-290
Bringmann, Karl I-607
Brockhoff, Dimo I-11, I-586
Bullinaria, John A. II-51
Burczyński, Tadeusz II-171
Burguillo, Juan C. I-455
Burke, Edmund K. I-465
Byrski, Aleksander I-475

Calvo, Roberto Wolfler II-219
Castillo, Pedro Ángel II-341
Castro-Gutierrez, Juan II-31
Chantler, Michael J. I-384
Chen, Shu-Heng II-199
Chen, Wenxiang II-300
Coello Coello, Carlos A. I-576, I-657, II-21, II-250
Collet, Pierre II-111
Corne, David I-424
Corne, David W. I-22, I-384
Cotta, Carlos I-445, I-475, II-421
Couldrey, Christine I-374
Czajkowski, Marcin I-324

Davison, Timothy II-310
Delarboulas, Pierre I-334
Deutz, André I-63, I-718
Doerr, Benjamin I-1, I-32, I-42, I-174, I-184
Dorigo, Marco II-331
Dorronsoro, Bernabé II-320
Dracopoulos, Dimitris C. II-181
Dragoni, Mauro I-344
Drugan, Madalina M. I-485
Dudek, Scott M. I-394
Dumitrescu, Dumitru I-506, II-361, II-432

El-Sourani, Nail II-442
Emmerich, Michael I-63, I-214, I-718

Farooq, Muddassar I-304
Ferrante, Eliseo II-331
Ferreira, Giordano B. II-381
Fialho, Álvaro I-194
Friedrich, Tobias I-607
Funke, Daniel II-41

Gagné, Christian II-141
Gajda, Ewa I-617
Gallagher, Marcus I-94
Galván-Lopéz, Edgar I-164
García, Antonio M. Mora II-341, II-452
García-Sánchez, Pablo II-341, II-452
Garcia-Najera, Abel II-51

Gardner, Matthew II-61
Garrett, Deon II-351
Gauci, Jason I-354
Gibbs, Jonathon I-496
Glasmachers, Tobias I-627
Goldberg, Leslie Ann I-32, I-174
Gómez-Pulido, Juan Antonio II-229
González, Jesús II-341
Gorse, Denise II-209
Graening, Lars II-71
Guervós, Juan J. Merelo II-341, II-452
Guo, Qiang I-465

Hains, Doug I-566
Hansen, Nikolaus I-11
Hao, Jin-Kao I-556
Hattori, Kiyohiko II-121
Hauschild, Mark II-462
Hirsch, Christian I-687, II-131
Hitotsuyanagi, Yasuhiro I-516, II-91
Hoai, Nguyen Xuan II-533
Hohm, Tim I-11
Hosny, Manar I. II-189
Howe, Adele I-566

Iclănzan, David I-506
Igel, Christian II-260
Iordache, Serban II-81
Ishibuchi, Hisao I-516, II-91

Jackson, David II-472
Jacob, Christian II-310, II-401
Jaimes, Antonio López I-657
Jansen, Thomas I-42
Jebalia, Mohamed I-52
Jiang, He I-546, I-637
Johannsen, Daniel I-184

Kampouridis, Michael II-199
Kendall, Graham I-465, I-496
Kerschbaum, Florian II-41
Knowles, Joshua II-151, II-161
Ko, Albert Hung-Ren II-141
Kobayashi, Shigenobu I-154, I-536
Kołodziej, Joanna I-526
Koohestani, Behrooz II-482
Kötzing, Timo I-184
Kratsch, Stefan I-204
Krawiec, Krzysztof II-492
Krętowski, Marek I-324
Krohn, Jean II-209
Kruisselbrink, Johannes I-63, I-214
Kuś, Wacław II-171

Labadi, Nacima II-219
Landa-Silva, Dario II-31
Laredo, Juan L. Jiménez II-341, II-452
Lässig, Jörg I-224, I-234
Laumanns, Marco I-597
Lehre, Per Kristian I-204, I-244
Leiva, Antonio J. Fernández I-445
Lichocki, Paweł II-492
Li, Ke I-647
Li, Miqing I-647
Logofătu, Doina II-361, II-432
Loshchilov, Ilya I-364
Louchet, Jean I-414
Luerssen, Martin II-502
Lung, Rodica Ioana II-432
Luo, Zhongxuan I-546
Lutton, Evelyne I-414
Lü, Zhipeng I-556

Mańdziuk, Jacek II-543
Martínez, Saúl Zapotecas I-576
Martins, Luiz G.A. II-381
Mathews, Nithin II-331
Matsushima, Hiroyasu II-121
Mavrovouniotis, Michalis II-371
McCall, John I-424, II-1
McDermott, James I-164
McKay, Bob II-533
McNabb, Andrew II-61
Melechovský, Jan II-219
Merelo, Juan J. II-421
Mersmann, Olaf I-73, II-101
Mezura-Montes, Efrén II-21
Montanier, Jean-Marc II-290
Moore, Jason H. I-404
Mora, Antonio M. II-421
Moraglio, Alberto I-83
Morgan, Rachael I-94
Mostaghim, Sanaz II-101
Mumford, Christine L. II-189

Nagata, Yuichi I-154, I-536
Nagy, Réka II-432
Naujoks, Boris I-728
Neumann, Frank I-1, I-184, I-204, I-667
Nojima, Yusuke I-516, II-91

Ochoa, Gabriela I-104
Olhofer, Markus II-71
Oliveira, Gina M.B. II-381
Oliveto, Pietro Simone I-204
O'Neill, Michael I-164, I-294, II-533
Ono, Isao I-154
Opara, Karol I-114
Otani, Masayuki II-121
Otto, Stephan II-240
Özcan, Ender I-496

Parizeau, Marc II-141
Pasia, Joseph M. I-677
Peleteiro, Ana I-455
Pelikan, Martin II-462
Pereira, Francisco B. II-523
Pérez, José Moreno II-31
Pestelacci, Enea II-512
Petrovski, Andrei II-1
Piccoli, Riccardo II-181
Poli, Riccardo II-482
Ponsich, Antonin II-250
Ponweiser, Wolfgang I-718
Potter, Mitchell A. I-374
Powers, David II-502
Preuss, Mike I-73, I-314

Qian, Chao I-144

Raidl, Günther R. II-391
Ren, Zhilei I-546, I-637
Reynolds, Alan P. I-22, I-384
Ritchie, Marylyn D. I-394
Rocchisani, Jean-Marie I-414
Rohlfshagen, Philipp I-284
Ros, Raymond I-194
Rudolph, Günter I-597, I-728
Runarsson, Thomas Philip II-421
Ruthmair, Mario II-391

Sánchez-Pérez, Juan Manuel II-229
Sarraf Shirazi, Abbas II-401
Sato, Hiroyuki II-121
Schaefer, Robert I-475, I-617
Schaul, Tom I-627
Schmeck, Hartmut I-687, II-131
Schmidhuber, Jürgen I-627
Schoenauer, Marc I-194, I-334, I-364
Sebag, Michèle I-194, I-334, I-364
Seppi, Kevin II-61
Shahzad, Farrukh I-304
Sharma, Deepak II-111
Shen, Ruimin I-647
Shimada, Tomohiro II-121
Shukla, Pradyumn Kumar I-687, II-131
Smołka, Maciej I-475, I-617
Stanley, Kenneth O. I-354, II-270
Studniarski, Marcin I-697
Sudholt, Dirk I-42, I-124, I-224, I-234
Szczepankiewicz, Adam II-411

Takadama, Keiki II-121
Tanaka, Kiyoshi I-657, I-677, II-11
Tang, Ke II-300
Tavares, Jorge II-523
Tettamanzi, Andrea G.B. I-344
Teytaud, Fabien I-254
Teytaud, Olivier I-254
Theile, Madeleine I-184, I-667
Thiele, Lothar I-707
Thierens, Dirk I-264, I-485
Tinós, Renato I-274
Tomassini, Marco I-104, II-512
Trautmann, Heike I-73, II-101, II-260
Tsang, Edward II-199
Tsukamoto, Noritaka II-91
Turgut, Ali Emre II-331
Turner, Stephen D. I-394

Ulrich, Tamara I-707
Urbanowicz, Ryan J. I-404
Uy, Nguyen Quang II-533

Valdivieso, Pedro Ángel Castillo II-452
Vatolkin, Igor I-314
Vega-Rodríguez, Miguel Angel II-229
Verel, Sébastien I-104
Vidal, Franck P. I-414
von Mammen, Sebastian II-310, II-401
Voß, Thomas II-260

Wagner, Tobias I-718
Wakamatsu, Yoshihiko I-516
Walędzik, Karol II-543
Wang, Yang I-556
Weicker, Karsten I-134
Weise, Thomas II-300
Wessing, Simon I-728
Whitacre, James M. I-284
Whitley, Darrell I-566
Winzen, Carola I-42

Woolley, Brian G. II-270
Wroniak, Tomasz II-411
Wu, Yanghui I-424

Xhafa, Fatos I-526
Xuan, Jifeng I-546

Yang, Shengxiang I-274, II-371
Yang, Zhenyu II-300
Yao, Xin I-284
Yu, Yang I-144
Yuan, Qizhao I-647

Zaharie, Daniela II-432
Zangeneh, Laleh I-434
Zarges, Christine I-42
Zhang, Shuyan I-637
Zheng, Jinhua I-647
Zhou, Zhi-Hua I-144