Marc Ebner Michael O'Neill
Anikó Ekárt Leonardo Vanneschi
Anna Isabel Esparcia-Alcázar (Eds.)

LNCS 4445

Genetic Programming

10th European Conference, EuroGP 2007
Valencia, Spain, April 2007
Proceedings

Springer

Marc Ebner Michael O'Neill
Anikó Ekárt Leonardo Vanneschi
Anna Isabel Esparcia-Alcázar (Eds.)

Genetic Programming

10th European Conference, EuroGP 2007
Valencia, Spain, April 11-13, 2007
Proceedings

Springer

Volume Editors

Marc Ebner
Universität Würzburg, Germany
E-mail: ebner@informatik.uni-wuerzburg.de

Michael O'Neill
University College Dublin, Ireland
E-mail: m.oneill@ucd.ie

Anikó Ekárt
Aston University, Birmingham, UK
E-mail: ekarta@aston.ac.uk

Leonardo Vanneschi
University of Milano–Bicocca, Italy
E-mail: vanneschi@disco.unimib.it

Anna Isabel Esparcia-Alcázar
Universidad Politécnica de Valencia, Spain
E-mail: anna@iti.upv.es

Cover illustration: Morphogenesis series #12 by Jon McCormack, 2006

Library of Congress Control Number: 2007923720

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

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

ISSN 0302-9743
ISBN-10 3-540-71602-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-71602-0 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12041664 06/3180 5 4 3 2 1 0

Lecture Notes in Computer Science 4445

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

Editorial Board

Preface

In 2007 the European Conference on Genetic Programming (EuroGP) reached in its tenth year. What started out as a small workshop in 1998 in Paris has grown considerably in size over the years both in number of attendees as well as number of submissions. EuroGP is the only conference worldwide devoted exclusively to genetic programming and all aspects of evolutionary generation of computer programs. For the tenth year we came together to exchange our ideas on the automatic generation of programs inspired by Darwinian evolution. The main operators are reproduction, variation and selection. In nature, heritable traits are passed from one generation to the next. Variations are introduced through accidental mutations or by recombining genetic material from parents. Selection occurs as a result of limited resources. The very same process is used when trying to evolve programs using artificial evolution. The desired task for the programs to perform is specified via the fitness function.

This year we received a record number of 71 submissions. A rigorous, double-blind, selection mechanism was applied to the submitted papers. We accepted 21 plenary talks (30% acceptance rate) and 14 poster presentations (49% global acceptance rate for talks and posters). Each submissions was reviewed by at least three members of the international Program Committee from 19 different countries. Each reviewer was asked for keywords specifying their own area of expertise. Submissions were then appropriately matched to the reviewers based on their expertise using the optimal assignment of the conference management software (MyReview) originally developed by Philippe Rigaux, Bertrand Chardon and colleagues from the Université Paris-Sud Orsay, France. This version of the MyReview system has been developed with funding from the European Coordinated Action ONCE-CS (Open Network Connecting Excellence in Complex Systems), funded under FP6 framework by the FET division (contract 15539). Only small adjustments were then made manually to balance the work load better.

Papers were accepted for presentation at the conference based on the reviewers' recommendations and also taking into account originality, significance of results, clarity of representation, replicability, writing style, bibliography and relevance to the conference. As a result, 35 high-quality papers are included in these proceedings which address fundamental and theoretical issues such as crossover bias, issues such as chess game playing, real-time evaluation of VoIP, multi-objective optimization, evolution of recursive sorting algorithms, density estimation for inverse problem solving, image filter evolution, predicting prime numbers, data mining, grammatical genetic programming, layered learning, expression simplification, neutrality and evolvability, iterated function systems, particle swarm optimization, or open-ended evolution. The use of genetic

programming for several different applications shows that the method is a very general problem-solving paradigm.

The 10th European Conference on Genetic Programming took place during April 2007 11–13 in Valencia, Spain. The present volume contains all contributions that were accepted for publication either as talks or posters. All previous proceedings have been published by Springer in the *Lecture Notes in Computer Science* series. EuroGP was co-located with EvoCOP 2007, the seventh European Conference on Evolutionary Computation in Combinatorial Optimization, and also EvoBIO, fifth European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, and the series of EvoWorkshops, focusing on applications of evolutionary computation. Evo* (pronounced EvoStar) is the new umbrella name for the three co-located conferences and the EvoWorkshops series, the increasingly important international event exclusively dedicated to all aspects of evolutionary computing.

Many people helped to make the conference a success. We would like to express our gratitude to the members of the Program Committee for their thorough reviews of all papers submitted to the conference. Their constructive comments made it possible for the authors to improve their original submissions for final publication. We also thank the following institutions. The Universidad Politécnica de Valéncia provided institutional and financial support, as well as the lending of their premises and also helped with the organization and administration. The Instituto Tecnológico de Informática cooperated with regard to the local organization. The Spanish Ministerio de Educación y Ciencia also provided financial support for which we are very grateful. We are thankful to Marc Schoenauer, INRIA, France, for managing the MyReview conference management software.

We especially thank Jennifer Willies and the School of Computing, Napier University. Without her dedicated work and continued involvement with the EuroGP conference from the initial start in 1998 to what now has become Evo*, this event would not be what it is today.

April 2007

Marc Ebner
Michael O'Neill
Anikó Ekárt
Leonardo Vanneschi
Anna Isabel Esparcia-Alcázar

Organization

Administrative details were handled by Jennifer Willies, Napier University, School of Computing, Scotland, UK.

Organizing Committee

Program Co-chairs	Marc Ebner (Universität Würzburg, Germany) Michael O'Neill (University College Dublin, Ireland)
Publication Chair	Anikó Ekárt (Aston University, UK)
Local Chair	Anna Isabel Esparcia-Alcázar (Universidad Politécnica de Valencia, Spain)
Publicity Chair	Leonardo Vanneschi (University of Milano-Bicocca, Italy)

Program Committee

Abbass, Hussein. UNSW@ADFA, Australia
Altenberg, Lee. University of Hawaii at Manoa, USA
Aydin, Mehmet Emin. University of Bedfordshire, UK
Azad, R. Muhammad Atif. University of Limerick, Ireland
Banzhaf, Wolfgang. Memorial University of Newfoundland, Canada
Bentley, Peter. University College London, UK
Brabazon, Anthony. University College Dublin, Ireland
Bredeche, Nicolas. Université Paris-Sud, France
Burke, Edmund Kieran. University of Nottingham, UK
Cagnoni, Stefano. University of Parma, Italy
Cheang, Sin Man. Hong Kong Institute of Vocational Education, China
Collard, Philippe. I3S laboratory-UNSA, France
Collet, Pierre. Université du Littoral, Côte d'Opale, France
Costa, Ernesto. University of Coimbra, Portugal
de Jong, Edwin. Utrecht University, The Netherlands
Defoin Platel, Michael. Laboratoire d'Océanographie de Villefranche, France
Dempsey, Ian. University College Dublin, Ireland
Divina, Federico. Tilburg University, The Netherlands
Ebner, Marc. Universität Würzburg, Germany
Ekárt, Anikó. Aston University, UK
Essam, Daryl. University of New South Wales, Australia
Fernández de Vega, Francisco. University of Extremadura, Spain
Folino, Gianluigi. ICAR-CNR, Italy
Fonlupt, Cyril. LIL - Université du Littoral, Côte d'Opale, France
Gagné, Christian. Informatique WGZ Inc., Canada
Giacobini, Mario. University of Turin, Italy

Gustafson, Steven. GE Global Research, USA
Hao, Jin-Kao. University of Angers, France
Harvey, Inman. University of Sussex, UK
Hoai, Nguyen Xuan. The Vietnamese Military Technical Academy, Vietnam
Hornby, Greg. University of California, Santa Cruz, USA
Howard, Daniel. QinetiQ, UK
Johnson, Colin. University of Kent, UK
Kalganova, Tatiana. Brunel University, UK
Keijzer, Maarten. Chordiant Software, The Netherlands
Keller, Robert E. University of Essex, UK
Kendall, Graham. University of Nottingham, UK
Khan, Asifullah. Pakistan Inst. of Engineering and Applied Sciences, Pakistan
Kim, Daeeun. University of Leicester, UK
Kubalik, Jiri. Czech Technical University, Czech Republic
Levine, John. University of Strathclyde, UK
Lopes, Heitor Silverio. Federal Technological University of Parana, Brazil
Lucas, Simon. University of Essex, UK
Machado, Penousal. University of Coimbra, Portugal
Martin, Peter. Naiad Consulting Limited, UK
McKay, Bob. Seoul National University, Korea
Mehnen, Jörn. University of Dortmund, Germany
Miller, Julian. University of York, UK
Nicolau, Miguel. INRIA, France
Nievola, Julio Cesar. PUCPR, Brazil
O'Neill, Michael. University College Dublin, Ireland
O'Reilly, Una-May. Massachusetts Institute of Technology, USA
Pizzuti, Clara. ICAR-CNR, Italy
Poli, Riccardo. University of Essex, UK
Ray, Thomas. University of Oklahoma, USA
Robilliard, Denis. Université du Littoral, Cote d'Opale, France
Schoenauer, Marc. INRIA, France
Sekanina, Lukás. Brno University of Technology, Czech Republic
Sipper, Moshe. Ben-Gurion University, Israel
Skourikhine, Alexei. Los Alamos National Laboratory, USA
Soule, Terence. University of Idaho, USA
Tettamanzi, Andrea. University of Milan, Italy
Thompson, Adrian. University of Sussex, UK
Tomassini, Marco. University of Lausanne, Switzerland
van Hemert, Jano. University of Edinburgh, UK
Vanneschi, Leonardo. University of Milano-Bicocca, Italy
Verel, Sébastien. University of Nice-Sophia Antipolis, France
Yu, Tina. Memorial University of Newfoundland, Canada

Lecture Notes in Computer Science

For information about Vols. 1–4334

please contact your bookseller or Springer

Vol. 4446: C. Cotta, J. van Hemert (Eds.), Evolutionary Computation in Combinatorial Optimization. XII, 241 pages. 2007.

Vol. 4445: M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, A.I. Esparcia-Alcázar (Eds.), Genetic Programming. XI, 382 pages. 2007.

Vol. 4444: T. Reps, M. Sagiv, J. Bauer (Eds.), Program Analysis and Compilation, Theory and Practice. X, 361 pages. 2007.

Vol. 4430: C.C. Yang, D. Zeng, M. Chau, K. Chang, Q. Yang, X. Cheng, J. Wang, F.-Y. Wang, H. Chen (Eds.), Intelligence and Security Informatics. XII, 330 pages. 2007.

Vol. 4429: R. Lu, J.H. Siekmann, C. Ullrich (Eds.), Cognitive Systems. X, 161 pages. 2007. (Sublibrary LNAI).

Vol. 4427: S. Uhlig, K. Papagiannaki, O. Bonaventure (Eds.), Passive and Active Network Measurement. XI, 274 pages. 2007.

Vol. 4425: G. Amati, C. Carpineto, G. Romano (Eds.), Advances in Information Retrieval. XIX, 759 pages. 2007.

Vol. 4424: O. Grumberg, M. Huth (Eds.), Tools and Algorithms for the Construction and Analysis of Systems. XX, 738 pages. 2007.

Vol. 4423: H. Seidl (Ed.), Foundations of Software Science and Computational Structures. XVI, 379 pages. 2007.

Vol. 4422: M.B. Dwyer, A. Lopes (Eds.), Fundamental Approaches to Software Engineering. XV, 440 pages. 2007.

Vol. 4421: R. De Nicola (Ed.), Programming Languages and Systems. XVII, 538 pages. 2007.

Vol. 4420: S. Krishnamurthi, M. Odersky (Eds.), Compiler Construction. XIV, 233 pages. 2007.

Vol. 4419: P.C. Diniz, E. Marques, K. Bertels, M.M. Fernandes, J.M.P. Cardoso (Eds.), Reconfigurable Computing: Architectures, Tools and Applications. XIV, 391 pages. 2007.

Vol. 4418: A. Gagalowicz, W. Philips (Eds.), Computer Vision/Computer Graphics Collaboration Techniques. XV, 620 pages. 2007.

Vol. 4416: A. Bemporad, A. Bicchi, G. Buttazzo (Eds.), Hybrid Systems: Computation and Control. XVII, 797 pages. 2007.

Vol. 4415: P. Lukowicz, L. Thiele, G. Tröster (Eds.), Architecture of Computing Systems - ARCS 2007. X, 297 pages. 2007.

Vol. 4414: S. Hochreiter, R. Wagner (Eds.), Bioinformatics Research and Development. XVI, 482 pages. 2007. (Sublibrary LNBI).

Vol. 4410: A. Branco (Ed.), Anaphora: Analysis, Algorithms and Applications. X, 191 pages. 2007. (Sublibrary LNAI).

Vol. 4407: G. Puebla (Ed.), Logic-Based Program Synthesis and Transformation. VIII, 237 pages. 2007.

Vol. 4405: L. Padgham, F. Zambonelli (Eds.), Agent-Oriented Software Engineering VII. XII, 225 pages. 2007.

Vol. 4403: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization. XIX, 954 pages. 2007.

Vol. 4400: J.F. Peters, A. Skowron, V.W. Marek, E. Orłowska, R. Slowinski, W. Ziarko (Eds.), Transactions on Rough Sets VII, Part II. X, 381 pages. 2007.

Vol. 4399: X. Llorà, T. Kovacs, K. Takadama, P.L. Lanzi, S.W. Wilson, W. Stolzmann (Eds.), Learning Classifier Systems. XII, 345 pages. 2007. (Sublibrary LNAI).

Vol. 4398: S. Marchand-Maillet, E. Bruno, A. Nürnberger, M. Detyniecki (Eds.), Adaptive Multimedia Retrieval: User, Context, and Feedback. XI, 269 pages. 2007.

Vol. 4397: C. Stephanidis, M. Pieper (Eds.), Universal Access in Ambient Intelligence Environments. XV, 467 pages. 2007.

Vol. 4396: J. García-Vidal, L. Cerdà-Alabern (Eds.), Wireless Systems and Mobility in Next Generation Internet. IX, 271 pages. 2007.

Vol. 4395: M. Daydé, J.M.L.M. Palma, Á.L.G.A. Coutinho, E. Pacitti, J.C. Lopes (Eds.), High Performance Computing for Computational Science - VECPAR 2006. XXIV, 721 pages. 2007.

Vol. 4394: A. Gelbukh (Ed.), Computational Linguistics and Intelligent Text Processing. XVI, 648 pages. 2007.

Vol. 4393: W. Thomas, P. Weil (Eds.), STACS 2007. XVIII, 708 pages. 2007.

Vol. 4392: S.P. Vadhan (Ed.), Theory of Cryptography. XI, 595 pages. 2007.

Vol. 4391: Y. Stylianou, M. Faundez-Zanuy, A. Esposito (Eds.), Progress in Nonlinear Speech Processing. XII, 269 pages. 2007.

Vol. 4390: S.O. Kuznetsov, S. Schmidt (Eds.), Formal Concept Analysis. X, 329 pages. 2007. (Sublibrary LNAI).

Vol. 4389: D. Weyns, H.V.D. Parunak, F. Michel (Eds.), Environments for Multi-Agent Systems III. X, 273 pages. 2007. (Sublibrary LNAI).

Vol. 4385: K. Coninx, K. Luyten, K.A. Schneider (Eds.), Task Models and Diagrams for Users Interface Design. XI, 355 pages. 2007.

Vol. 4384: T. Washio, K. Satoh, H. Takeda, A. Inokuchi (Eds.), New Frontiers in Artificial Intelligence. IX, 401 pages. 2007. (Sublibrary LNAI).

Vol. 4383: E. Bin, A. Ziv, S. Ur (Eds.), Hardware and Software, Verification and Testing. XII, 235 pages. 2007.

Vol. 4381: J. Akiyama, W.Y.C. Chen, M. Kano, X. Li, Q. Yu (Eds.), Discrete Geometry, Combinatorics and Graph Theory. XI, 289 pages. 2007.

Vol. 4380: S. Spaccapietra, P. Atzeni, F. Fages, M.-S. Hacid, M. Kifer, J. Mylopoulos, B. Pernici, P. Shvaiko, J. Trujillo, I. Zaihrayeu (Eds.), Journal on Data Semantics VIII. XV, 219 pages. 2007.

Vol. 4378: I. Virbitskaite, A. Voronkov (Eds.), Perspectives of Systems Informatics. XIV, 496 pages. 2007.

Vol. 4377: M. Abe (Ed.), Topics in Cryptology – CT-RSA 2007. XI, 403 pages. 2006.

Vol. 4376: E. Frachtenberg, U. Schwiegelshohn (Eds.), Job Scheduling Strategies for Parallel Processing. VII, 257 pages. 2007.

Vol. 4374: J.F. Peters, A. Skowron, I. Düntsch, J. Grzymała-Busse, E. Orłowska, L. Polkowski (Eds.), Transactions on Rough Sets VI, Part I. XII, 499 pages. 2007.

Vol. 4373: K. Langendoen, T. Voigt (Eds.), Wireless Sensor Networks. XIII, 358 pages. 2007.

Vol. 4372: M. Kaufmann, D. Wagner (Eds.), Graph Drawing. XIV, 454 pages. 2007.

Vol. 4371: K. Inoue, K. Satoh, F. Toni (Eds.), Computational Logic in Multi-Agent Systems. X, 315 pages. 2007. (Sublibrary LNAI).

Vol. 4370: P.P Lévy, B. Le Grand, F. Poulet, M. Soto, L. Darago, L. Toubiana, J.-F. Vibert (Eds.), Pixelization Paradigm. XV, 279 pages. 2007.

Vol. 4369: M. Umeda, A. Wolf, O. Bartenstein, U. Geske, D. Seipel, O. Takata (Eds.), Declarative Programming for Knowledge Management. X, 229 pages. 2006. (Sublibrary LNAI).

Vol. 4368: T. Erlebach, C. Kaklamanis (Eds.), Approximation and Online Algorithms. X, 345 pages. 2007.

Vol. 4367: K. De Bosschere, D. Kaeli, P. Stenström, D. Whalley, T. Ungerer (Eds.), High Performance Embedded Architectures and Compilers. XI, 307 pages. 2007.

Vol. 4366: K. Tuyls, R. Westra, Y. Saeys, A. Nowé (Eds.), Knowledge Discovery and Emergent Complexity in Bioinformatics. IX, 183 pages. 2007. (Sublibrary LNBI).

Vol. 4364: T. Kühne (Ed.), Models in Software Engineering. XI, 332 pages. 2007.

Vol. 4362: J. van Leeuwen, G.F. Italiano, W. van der Hoek, C. Meinel, H. Sack, F. Plášil (Eds.), SOFSEM 2007: Theory and Practice of Computer Science. XXI, 937 pages. 2007.

Vol. 4361: H.J. Hoogeboom, G. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing. IX, 555 pages. 2006.

Vol. 4360: W. Dubitzky, A. Schuster, P.M.A. Sloot, M. Schroeder, M. Romberg (Eds.), Distributed, High-Performance and Grid Computing in Computational Biology. X, 192 pages. 2007. (Sublibrary LNBI).

Vol. 4358: R. Vidal, A. Heyden, Y. Ma (Eds.), Dynamical Vision. IX, 329 pages. 2007.

Vol. 4357: L. Buttyán, V. Gligor, D. Westhoff (Eds.), Security and Privacy in Ad-Hoc and Sensor Networks. X, 193 pages. 2006.

Vol. 4355: J. Julliand, O. Kouchnarenko (Eds.), B 2007: Formal Specification and Development in B. XIII, 293 pages. 2006.

Vol. 4354: M. Hanus (Ed.), Practical Aspects of Declarative Languages. X, 335 pages. 2006.

Vol. 4353: T. Schwentick, D. Suciu (Eds.), Database Theory – ICDT 2007. XI, 419 pages. 2006.

Vol. 4352: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part II. XVIII, 743 pages. 2006.

Vol. 4351: T.-J. Cham, J. Cai, C. Dorai, D. Rajan, T.-S. Chua, L.-T. Chia (Eds.), Advances in Multimedia Modeling, Part I. XIX, 797 pages. 2006.

Vol. 4349: B. Cook, A. Podelski (Eds.), Verification, Model Checking, and Abstract Interpretation. XI, 395 pages. 2007.

Vol. 4348: S.T. Taft, R.A. Duff, R.L. Brukardt, E. Ploedereder, P. Leroy (Eds.), Ada 2005 Reference Manual. XXII, 765 pages. 2006.

Vol. 4347: J. Lopez (Ed.), Critical Information Infrastructures Security. X, 286 pages. 2006.

Vol. 4346: L. Brim, B. Haverkort, M. Leucker, J. van de Pol (Eds.), Formal Methods: Applications and Technology. X, 363 pages. 2007.

Vol. 4345: N. Maglaveras, I. Chouvarda, V. Koutkias, R. Brause (Eds.), Biological and Medical Data Analysis. XIII, 496 pages. 2006. (Sublibrary LNBI).

Vol. 4344: V. Gruhn, F. Oquendo (Eds.), Software Architecture. X, 245 pages. 2006.

Vol. 4342: H. de Swart, E. Orłowska, G. Schmidt, M. Roubens (Eds.), Theory and Applications of Relational Structures as Knowledge Instruments II. X, 373 pages. 2006. (Sublibrary LNAI).

Vol. 4341: P.Q. Nguyen (Ed.), Progress in Cryptology - VIETCRYPT 2006. XI, 385 pages. 2006.

Vol. 4340: R. Prodan, T. Fahringer, Grid Computing. XXIII, 317 pages. 2007.

Vol. 4339: E. Ayguadé, G. Baumgartner, J. Ramanujam, P. Sadayappan (Eds.), Languages and Compilers for Parallel Computing. XI, 476 pages. 2006.

Vol. 4338: P. Kalra, S. Peleg (Eds.), Computer Vision, Graphics and Image Processing. XV, 965 pages. 2006.

Vol. 4337: S. Arun-Kumar, N. Garg (Eds.), FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science. XIII, 430 pages. 2006.

Vol. 4336: V.R. Basili, D. Rombach, K. Schneider, B. Kitchenham, D. Pfahl, R.W. Selby, Empirical Software Engineering Issues. XVII, 193 pages. 2007.

Vol. 4335: S.A. Brueckner, S. Hassas, M. Jelasity, D. Yamins (Eds.), Engineering Self-Organising Systems. XII, 212 pages. 2007. (Sublibrary LNAI).

¥562.00元

Table of Contents

A Grammatical Genetic Programming Approach to Modularity in Genetic Algorithms 1
Erik Hemberg, Conor Gilligan, Michael O'Neill, and Anthony Brabazon

An Empirical Boosting Scheme for ROC-Based Genetic Programming Classifiers 12
Denis Robilliard, Virginie Marion-Poty, Sébastien Mahler, and Cyril Fonlupt

Confidence Intervals for Computational Effort Comparisons 23
Matthew Walker, Howard Edwards, and Chris Messom

Crossover Bias in Genetic Programming 33
Maarten Keijzer and James Foster

Density Estimation with Genetic Programming for Inverse Problem Solving 45
Michael Defoin Platel, Sébastien Vérel, Manuel Clergue, and Malik Chami

Empirical Analysis of GP Tree-Fragments 55
Will Smart, Peter Andreae, and Mengjie Zhang

Empirical Comparison of Evolutionary Representations of the Inverse Problem for Iterated Function Systems 68
Anargyros Sarafopoulos and Bernard Buxton

Evolution of an Efficient Search Algorithm for the Mate-In-N Problem in Chess 78
Ami Hauptman and Moshe Sipper

Fast Genetic Programming on GPUs 90
Simon Harding and Wolfgang Banzhaf

FIFTH^{TM}: A Stack Based GP Language for Vector Processing 102
Kenneth Holladay, Kay Robbins, and Jeffery von Ronne

Genetic Programming with Fitness Based on Model Checking 114
Colin G. Johnson

Geometric Particle Swarm Optimisation 125
Alberto Moraglio, Cecilia Di Chio, and Riccardo Poli

GP Classifier Problem Decomposition Using First-Price and Second-Price Auctions 137
Peter Lichodzijewski and Malcolm I. Heywood

Layered Learning in Boolean GP Problems 148
David Jackson and Adrian P. Gibbons

Mining Distributed Evolving Data Streams Using Fractal GP Ensembles 160
Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano

Multi-objective Genetic Programming for Improving the Performance of TCP 170
Cyril Fillon and Alberto Bartoli

On Population Size and Neutrality: Facilitating the Evolution of Evolvability 181
Richard M. Downing

On the Limiting Distribution of Program Sizes in Tree-Based Genetic Programming 193
Riccardo Poli, William B. Langdon, and Stephen Dignum

Predicting Prime Numbers Using Cartesian Genetic Programming 205
James Alfred Walker and Julian Francis Miller

Real-Time, Non-intrusive Evaluation of VoIP 217
Adil Raja, R. Muhammad Atif Azad, Colin Flanagan, and Conor Ryan

Training Binary GP Classifiers Efficiently: A Pareto-coevolutionary Approach 229
Michal Lemczyk and Malcolm I. Heywood

Posters

A Comprehensive View of Fitness Landscapes with Neutrality and Fitness Clouds 241
Leonardo Vanneschi, Marco Tomassini, Philippe Collard, Sébastien Vérel, Yuri Pirola, and Giancarlo Mauri

Analysing the Regularity of Genomes Using Compression and Expression Simplification 251
Jungseok Shin, Moonyoung Kang, R.I. (Bob) McKay, Xuan Nguyen, Tuan-Hao Hoang, Naoki Mori, and Daryl Essam

Changing the Genospace: Solving GA Problems with Cartesian Genetic Programming 261
James Alfred Walker and Julian Francis Miller

Code Regulation in Open Ended Evolution 271
Lidia Yamamoto

Data Mining of Genetic Programming Run Logs 281
Vic Ciesielski and Xiang Li

Evolving a Statistics Class Using Object Oriented Evolutionary Programming 291
Alexandros Agapitos and Simon M. Lucas

Evolving Modular Recursive Sorting Algorithms 301
Alexandros Agapitos and Simon M. Lucas

Fitness Landscape Analysis and Image Filter Evolution Using Functional-Level CGP 311
Karel Slaný and Lukáš Sekanina

Genetic Programming Heuristics for Multiple Machine Scheduling 321
Domagoj Jakobović, Leonardo Jelenković, and Leo Budin

Group-Foraging with Particle Swarms and Genetic Programming 331
Cecilia Di Chio and Paolo Di Chio

Multiple Interactive Outputs in a Single Tree: An Empirical Investigation 341
Edgar Galván-López and Katya Rodríguez-Vázquez

Parsimony Doesn't Mean Simplicity: Genetic Programming for Inductive Inference on Noisy Data 351
Ivanoe De Falco, Antonio Della Cioppa, Domenico Maisto, Umberto Scafuri, and Ernesto Tarantino

The Holland Broadcast Language and the Modeling of Biochemical Networks 361
James Decraene, George G. Mitchell, Barry McMullin, and Ciaran Kelly

The Induction of Finite Transducers Using Genetic Programming 371
Amashini Naidoo and Nelishia Pillay

Author Index 381

A Grammatical Genetic Programming Approach to Modularity in Genetic Algorithms

Erik Hemberg[1], Conor Gilligan[1], Michael O'Neill[1], and Anthony Brabazon[2]

[1] UCD Natural Computing Research & Applications
School of Computer Science and Informatics
University College Dublin, Ireland
erik.hemberg@ucd.ie, conor.gilligan@ucd.ie, m.oneill@ucd.ie
[2] UCD Natural Computing Research & Applications
School of Business
University College Dublin, Ireland
anthony.brabazon@ucd.ie

Abstract. The ability of Genetic Programming to scale to problems of increasing difficulty operates on the premise that it is possible to capture regularities that exist in a problem environment by decomposition of the problem into a hierarchy of modules. As computer scientists and more generally as humans we tend to adopt a similar divide-and-conquer strategy in our problem solving. In this paper we consider the adoption of such a strategy for Genetic Algorithms. By adopting a modular representation in a Genetic Algorithm we can make efficiency gains that enable superior scaling characteristics to problems of increasing size. We present a comparison of two modular Genetic Algorithms, one of which is a Grammatical Genetic Programming algorithm, the meta-Grammar Genetic Algorithm (mGGA), which generates binary string sentences instead of traditional GP trees. A number of problems instances are tackled which extend the Checkerboard problem by introducing different kinds of regularity and noise. The results demonstrate some limitations of the modular GA (MGA) representation and how the mGGA can overcome these. The mGGA shows improved scaling when compared the MGA.

1 Introduction

In the natural world examples of modularity and hierarchies abound, ranging the biological evolution of cells to form tissues and organs to the physical structure of matter from the sub-atomic level up. In most examples of problem solving by humans, regularities in the problem environment are exploited in a divide-and-conquer approach through the construction of sub-solutions, which may then be reused and combined in a hierarchical fashion to solve the problem as a whole. Similarly Genetic Programming provides as components of its problem solving toolkit the ability to automatically create, modify and delete modules, which can be used in a hierarchical fashion. The objectives of this study are to investigate the adoption of principles from Genetic Programming [1] such as modularity and reuse (see Chapter 16 in [2]) for application to Genetic Algorithms, and to

M. Ebner et al. (Eds.): EuroGP 2007, LNCS 4445, pp. 1–11, 2007.

couple these to an adaptive representation that allows the type and usage of these principles to be evolved through the use of evolvable grammars. The goal being the development of an evolutionary algorithm with good scaling characteristics, and an adaptable representation that will facilitate its application to noisy, dynamic, problem environments. To this end a grammar-based Genetic Programming approach is adopted, in which the grammars represent the construction of syntactically correct genotypes of the Genetic Algorithm. In particular, we compare the representations and performance of the meta-Grammar Genetic Algorithm (mGGA) [3] to the Modular Genetic Algorithm (MGA) [4], highlighting some of the MGA's representational limitations, and demonstrate the potential of a more expressive representation in the form of the mGGA to scale to problems of increasing size and difficulty. Additionally, we consider the introduction of noise into the Checkerboard problem, in order to assess how the representations might generalise into noisy, real-world problem domains. The remainder of the paper is structured as follows. Section 2 provides background on earlier work in modular GAs and describes the meta-Grammar Genetic Algorithm. Section 3 details the experimental approach adopted and results, and finally section 4 details conclusions and future work.

2 Background

There has been a large body of research on modularity in Genetic Programming and effects on its scalability, however the same cannot be stated for the Genetic Algorithm (GA). In this section we present two modular representations as implemented in the Modular GA [4] and the meta-Grammar GA [3].

2.1 Modular Genetic Algorithm

Garibay et al. introduced the Modular Genetic Algorithm, which was shown to signficantly outperform a standard Genetic Algorithm on a scalable problem with regularities [4]. The genome of an MGA individual is a vector of genes, where each gene is comprised of two components, the `number-of-repetitions` and some `function` which is repeated according to the value of the repetitions field. For example, if we had a function (`one()`) that always returned the value `1` when called and another (`zero()`) that returned the value `0` we have a representation that can generate binary strings. A sample individual comprised of three genes might look like: `{2, zero()}, {4, one()}, {2, zero()}`, which would produce the binary string `00111100`. The MGA was shown to have superior ability to scale to problems of increasing complexity than a standard GA.

2.2 Grammatical Evolution by Grammatical Evolution

The grammar-based Genetic Programming approach upon which this study is based is the Grammatical Evolution by Grammatical Evolution algorithm [5], which is in turn based on the Grammatical Evolution algorithm [6,7,8,9]. This is

a meta-Grammar Evolutionary Algorithm in which the input grammar is used to specify the construction of another syntactically correct grammar. The generated grammar is then used in a mapping process to construct a solution. In order to allow evolution of a grammar (Grammatical Evolution by Grammatical Evolution $(GE)^2$), we must provide a grammar to specify the form a grammar can take. This is an example of the richness of the expressiveness of grammars that makes the GE approach so powerful. See [6,10,11] for further examples of what can be represented with grammars and [12] for an alternative approach to grammar evolution. By allowing an Evolutionary Algorithm to adapt its representation (in this case through the evolution of the grammar) it provides the population with enhanced robustness in the face of a dynamic environment, in particular, and also to automatically incorporate biases into the search process. In this case we can allow the meta-Grammar Genetic Algorithm to evolve biases towards different building blocks of varying sizes. In this approach we therefore have two distinct grammars, the *universal grammar* (or grammars' grammar) and the *solution grammar*. The notion of a universal grammar is adopted from linguistics and refers to a universal set of syntactic rules that hold for spoken languages [13]. It has been proposed that during a child's development the universal grammar undergoes modifications through learning that allows the development of communication in their parents native language(s) [14]. In $(GE)^2$ the universal grammar dictates the construction of the solution grammar. In this study two separate, variable-length, genotypic binary chromosomes were used, the first chromosome to generate the solution grammar from the universal grammar and the second chromosome generates the solution itself. Crossover operates between homologous chromosomes, that is, the solution grammar chromosome from the first parent recombines with the solution grammar chromosome from the second parent, with the same occurring for the solution chromosomes. In order for evolution to be successful it must co-evolve both the meta-Grammar and the structure of solutions based on the evolved meta-Grammar, and as such the search space is larger than in standard Grammatical Evolution.

2.3 Meta-grammars for Bitstrings

A simple grammar for a fixed-length (8 bits in the following example) binary string individual of a Genetic Algorithm is provided below. In the generative grammar each bit position (denoted as `<bit>`) can become either of the boolean values. A standard variable-length Grammatical Evolution individual can then be allowed to specify what each bit value will be by selecting the appropriate `<bit>` production rule for each position in the `<bitstring>`.

```
<bitstring> ::= <bit><bit><bit><bit><bit><bit><bit><bit>
<bit> ::= 1 | 0
```

The above grammar can be extended to incorporate the reuse of groups of bits (building blocks). In this example all building blocks that are mutliples of two are provided, although it would be possible to create a grammar that adopted more complex arrangements of building blocks.

```
<bitstring> ::= <bbk4><bbk4> | <bbk2><bbk2><bbk2><bbk2>
              | <bbk1><bbk1><bbk1t><bbk1><bbk1><bbk1><bbk1><bbk1>
<bbk4> ::= <bit><bit><bit><bit>
<bbk2> ::= <bit><bit>
<bbk1> ::= <bit>
<bit> ::= 1 | 0
```

The above grammars are static, and as such can only allow one building block of size four and of size two in the second example. It would be better to allow our search algorithm the potential to uncover a number of building blocks of any one size from which a Grammatical Evolution individual could choose from. This would facilitate the application of such a Grammatical GA to:

- problems with more than one building block type for each building block size,
- to search on one building block while maintaining a *reasonable* temporary building block solution,
- and to be able to switch between building blocks in the case of dynamic environments.

All of this can be achieved through the adoption of meta-Grammars as were adopted earlier in [5]. An example of such a grammar for an 8-bit individual is given below.

```
<g> ::=  "<bitstring> ::=" <reps>
              "<bbk4> ::=" <bbk4t>
              "<bbk2> ::=" <bbk2t>
              "<bbk1> ::=" <bbk1t>
               "<bit> ::=" <val>

<bbk4t> ::= <bit><bit><bit><bit>
<bbk2t> ::= <bit><bit>
<bbk1t> ::= <bit>
<reps> ::= <rept> | <rept> "|" <reps>
<rept> ::= "<bbk4><bbk4>" | "<bbk2><bbk2><bbk2><bbk2>"
         | "<bbk1><bbk1><bbk1><bbk1><bbk1><bbk1><bbk1><bbk1>"
<bit> ::= "<bit>" | 1 | 0
<val> ::= <valt>  | <valt> "|" <val>
<valt> ::= 1 | 0
```

In this case the grammar specifies the construction of another generative bitstring grammar. The subsequent bitstring grammar that can be produced from the above meta-grammar is restricted such that it can contain building blocks of size 8. Some of the bits of the building blocks can be fully specified as a boolean value or may be left as unfilled for the second step in the mapping process. An example bitstring grammar produced from the above meta-grammar could be:

```
<bitstring> ::= <bit>11<bit>00<bit><bit> | <bbk2><bbk2><bbk2><bbk2>
              | 11011101 | <bbk4><bbk4> | <bbk4><bbk4>
<bbk4> ::= <bit>11<bit>
<bbk2> ::= 11
<bbk1> ::= 1
<bit> ::= 1 | 0 | 0 | 1
```

To allow the creation of multiple building blocks of different sizes the following grammar could be adopted (again shown for 8-bit strings).