Alden H. Wright
Michael D. Vose
Kenneth A. De Jong
Lothar M. Schmitt (Eds.)

Foundations of Genetic Algorithms

LNCS 3469

8th International Workshop, FOGA 2005
Aizu-Wakamatsu City, Japan, Januar 2005
Revised Selected Papers

Springer

Alden H. Wright Michael D. Vose
Kenneth A. De Jong Lothar M. Schmitt (Eds.)

Foundations of Genetic Algorithms

8th International Workshop, FOGA 2005
Aizu-Wakamatsu City, Japan, January 5-9, 2005
Revised Selected Papers

Springer

Volume Editors

Alden H. Wright
University of Montana
Computer Science Department, Missoula, MT 59812, USA
E-mail: alden.wright@umontana.edu

Michael D. Vose
University of Tennessee
Computer Science Department, 203 Claxton Complex, 1122 Volunteer Blvd.
Knoxville, TN 37996-3450, USA
E-mail: vose@cs.utk.edu

Kenneth A. De Jong
George Mason University
Computer Science Department, 4400 University Drive, Fairfax, VA 22030, USA
E-mail: kdejong@cs.gmu.edu

Lothar M. Schmitt
University of Aizu
Aizu-Wakamatsu, Fukushima 965-8580, Japan
E-mail: LMSchmitt@yahoo.com

Library of Congress Control Number: 2005928446

CR Subject Classification (1998): F.1-2, I.2, I.2.6, I.2.8, D.2.2

ISSN 0302-9743
ISBN-10 3-540-27237-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-27237-3 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Olgun Computergrafik
Printed on acid-free paper SPIN: 11513575 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 3469

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

Editorial Board

Lecture Notes in Computer Science

For information about Vols. 1–3478

please contact your bookseller or Springer

Vol. 3580: L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, M. Yung (Eds.), Automata, Languages and Programming. XXV, 1477 pages. 2005.

Vol. 3578: M. Gallagher, J. Hogan, F. Maire (Eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2005. XVI, 599 pages. 2005.

Vol. 3576: K. Etessami, S.K. Rajamani (Eds.), Computer Aided Verification. XV, 564 pages. 2005.

Vol. 3574: C. Boyd, J.M. González Nieto (Eds.), Information Security and Privacy. XIII, 586 pages. 2005.

Vol. 3573: S. Etalle (Ed.), Logic Based Program Synthesis and Transformation. VIII, 279 pages. 2005.

Vol. 3572: C. De Felice, A. Restivo (Eds.), Developments in Language Theory. XI, 409 pages. 2005.

Vol. 3571: L. Godo (Ed.), Symbolic and Quantitative Approaches to Reasoning with Uncertainty. XVI, 1028 pages. 2005. (Subseries LNAI).

Vol. 3570: A. S. Patrick, M. Yung (Eds.), Financial Cryptography and Data Security. XII, 376 pages. 2005.

Vol. 3569: F. Bacchus, T. Walsh (Eds.), Theory and Applications of Satisfiability Testing. XII, 492 pages. 2005.

Vol. 3567: M. Jackson, D. Nelson, S. Stirk (Eds.), Database: Enterprise, Skills and Innovation. XII, 185 pages. 2005.

Vol. 3565: G.E. Christensen, M. Sonka (Eds.), Information Processing in Medical Imaging. XXI, 777 pages. 2005.

Vol. 3562: J. Mira, J.R. Álvarez (Eds.), Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach, Part II. XXIV, 636 pages. 2005.

Vol. 3561: J. Mira, J.R. Álvarez (Eds.), Mechanisms, Symbols, and Models Underlying Cognition, Part I. XXIV, 532 pages. 2005.

Vol. 3560: V.K. Prasanna, S. Iyengar, P.G. Spirakis, M. Welsh (Eds.), Distributed Computing in Sensor Systems. XV, 423 pages. 2005.

Vol. 3559: P. Auer, R. Meir (Eds.), Learning Theory. XI, 692 pages. 2005. (Subseries LNAI).

Vol. 3557: H. Gilbert, H. Handschuh (Eds.), Fast Software Encryption. XI, 443 pages. 2005.

Vol. 3556: H. Baumeister, M. Marchesi, M. Holcombe (Eds.), Extreme Programming and Agile Processes in Software Engineering. XIV, 332 pages. 2005.

Vol. 3555: T. Vardanega, A. Wellings (Eds.), Reliable Software Technology – Ada-Europe 2005. XV, 273 pages. 2005.

Vol. 3554: A. Dey, B. Kokinov, D. Leake, R. Turner (Eds.), Modeling and Using Context. XIV, 572 pages. 2005. (Subseries LNAI).

Vol. 3553: T.D. Hämäläinen, A.D. Pimentel, J. Takala, S. Vassiliadis (Eds.), Embedded Computer Systems: Architectures, Modeling, and Simulation. XV, 476 pages. 2005.

Vol. 3552: H. de Meer, N. Bhatti (Eds.), Quality of Service – IWQoS 2005. XV, 400 pages. 2005.

Vol. 3551: T. Härder, W. Lehner (Eds.), Data Management in a Connected World. XIX, 371 pages. 2005.

Vol. 3548: K. Julisch, C. Kruegel (Eds.), Intrusion and Malware Detection and Vulnerability Assessment. X, 241 pages. 2005.

Vol. 3547: F. Bomarius, S. Komi-Sirviö (Eds.), Product Focused Software Process Improvement. XIII, 588 pages. 2005.

Vol. 3543: L. Kutvonen, N. Alonistioti (Eds.), Distributed Applications and Interoperable Systems. XI, 235 pages. 2005.

Vol. 3541: N.C. Oza, R. Polikar, J. Kittler, F. Roli (Eds.), Multiple Classifier Systems. XII, 430 pages. 2005.

Vol. 3540: H. Kalviainen, J. Parkkinen, A. Kaarna (Eds.), Image Analysis. XXII, 1270 pages. 2005.

Vol. 3537: A. Apostolico, M. Crochemore, K. Park (Eds.), Combinatorial Pattern Matching. XI, 444 pages. 2005.

Vol. 3536: G. Ciardo, P. Darondeau (Eds.), Applications and Theory of Petri Nets 2005. XI, 470 pages. 2005.

Vol. 3535: M. Steffen, G. Zavattaro (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 323 pages. 2005.

Vol. 3533: M. Ali, F. Esposito (Eds.), Innovations in Applied Artificial Intelligence. XX, 858 pages. 2005. (Subseries LNAI).

Vol. 3532: A. Gómez-Pérez, J. Euzenat (Eds.), The Semantic Web: Research and Applications. XV, 728 pages. 2005.

Vol. 3531: J. Ioannidis, A. Keromytis, M. Yung (Eds.), Applied Cryptography and Network Security. XI, 530 pages. 2005.

Vol. 3530: A. Prinz, R. Reed, J. Reed (Eds.), SDL 2005: Model Driven. XI, 361 pages. 2005.

Vol. 3528: P.S. Szczepaniak, J. Kacprzyk, A. Niewiadomski (Eds.), Advances in Web Intelligence. XVII, 513 pages. 2005. (Subseries LNAI).

Vol. 3527: R. Morrison, F. Oquendo (Eds.), Software Architecture. XII, 263 pages. 2005.

Vol. 3526: S.B. Cooper, B. Löwe, L. Torenvliet (Eds.), New Computational Paradigms. XVII, 574 pages. 2005.

Vol. 3525: A.E. Abdallah, C.B. Jones, J.W. Sanders (Eds.), Communicating Sequential Processes. XIV, 321 pages. 2005.

Vol. 3524: R. Barták, M. Milano (Eds.), Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. XI, 320 pages. 2005.

Vol. 3523: J.S. Marques, N. Pérez de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part II. XXVI, 733 pages. 2005.

Vol. 3522: J.S. Marques, N. Pérez de la Blanca, P. Pina (Eds.), Pattern Recognition and Image Analysis, Part I. XXVI, 703 pages. 2005.

Vol. 3521: N. Megiddo, Y. Xu, B. Zhu (Eds.), Algorithmic Applications in Management. XIII, 484 pages. 2005.

Vol. 3520: O. Pastor, J. Falcão e Cunha (Eds.), Advanced Information Systems Engineering. XVI, 584 pages. 2005.

Vol. 3519: H. Li, P. J. Olver, G. Sommer (Eds.), Computer Algebra and Geometric Algebra with Applications. IX, 449 pages. 2005.

Vol. 3518: T.B. Ho, D. Cheung, H. Liu (Eds.), Advances in Knowledge Discovery and Data Mining. XXI, 864 pages. 2005. (Subseries LNAI).

Vol. 3517: H.S. Baird, D.P. Lopresti (Eds.), Human Interactive Proofs. IX, 143 pages. 2005.

Vol. 3516: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part III. LXIII, 1143 pages. 2005.

Vol. 3515: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part II. LXIII, 1101 pages. 2005.

Vol. 3514: V.S. Sunderam, G.D.v. Albada, P.M.A. Sloot, J.J. Dongarra (Eds.), Computational Science – ICCS 2005, Part I. LXIII, 1089 pages. 2005.

Vol. 3513: A. Montoyo, R. Muñoz, E. Métais (Eds.), Natural Language Processing and Information Systems. XII, 408 pages. 2005.

Vol. 3512: J. Cabestany, A. Prieto, F. Sandoval (Eds.), Computational Intelligence and Bioinspired Systems. XXV, 1260 pages. 2005.

Vol. 3511: U.K. Wiil (Ed.), Metainformatics. VIII, 221 pages. 2005.

Vol. 3510: T. Braun, G. Carle, Y. Koucheryavy, V. Tsaoussidis (Eds.), Wired/Wireless Internet Communications. XIV, 366 pages. 2005.

Vol. 3509: M. Jünger, V. Kaibel (Eds.), Integer Programming and Combinatorial Optimization. XI, 484 pages. 2005.

Vol. 3508: P. Bresciani, P. Giorgini, B. Henderson-Sellers, G. Low, M. Winikoff (Eds.), Agent-Oriented Information Systems II. X, 227 pages. 2005. (Subseries LNAI).

Vol. 3507: F. Crestani, I. Ruthven (Eds.), Information Context: Nature, Impact, and Role. XIII, 253 pages. 2005.

Vol. 3506: C. Park, S. Chee (Eds.), Information Security and Cryptology – ICISC 2004. XIV, 490 pages. 2005.

Vol. 3505: V. Gorodetsky, J. Liu, V. A. Skormin (Eds.), Autonomous Intelligent Systems: Agents and Data Mining. XIII, 303 pages. 2005. (Subseries LNAI).

Vol. 3504: A.F. Frangi, P.I. Radeva, A. Santos, M. Hernandez (Eds.), Functional Imaging and Modeling of the Heart. XV, 489 pages. 2005.

Vol. 3503: S.E. Nikoletseas (Ed.), Experimental and Efficient Algorithms. XV, 624 pages. 2005.

Vol. 3502: F. Khendek, R. Dssouli (Eds.), Testing of Communicating Systems. X, 381 pages. 2005.

Vol. 3501: B. Kégl, G. Lapalme (Eds.), Advances in Artificial Intelligence. XV, 458 pages. 2005. (Subseries LNAI).

Vol. 3500: S. Miyano, J. Mesirov, S. Kasif, S. Istrail, P. Pevzner, M. Waterman (Eds.), Research in Computational Molecular Biology. XVII, 632 pages. 2005. (Subseries LNBI).

Vol. 3499: A. Pelc, M. Raynal (Eds.), Structural Information and Communication Complexity. X, 323 pages. 2005.

Vol. 3498: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part III. XLIX, 1077 pages. 2005.

Vol. 3497: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. XLIX, 947 pages. 2005.

Vol. 3496: J. Wang, X. Liao, Z. Yi (Eds.), Advances in Neural Networks – ISNN 2005, Part II. L, 1055 pages. 2005.

Vol. 3495: P. Kantor, G. Muresan, F. Roberts, D.D. Zeng, F.-Y. Wang, H. Chen, R.C. Merkle (Eds.), Intelligence and Security Informatics. XVIII, 674 pages. 2005.

Vol. 3494: R. Cramer (Ed.), Advances in Cryptology – EUROCRYPT 2005. XIV, 576 pages. 2005.

Vol. 3493: N. Fuhr, M. Lalmas, S. Malik, Z. Szlávik (Eds.), Advances in XML Information Retrieval. XI, 438 pages. 2005.

Vol. 3492: P. Blache, E. Stabler, J. Busquets, R. Moot (Eds.), Logical Aspects of Computational Linguistics. X, 363 pages. 2005. (Subseries LNAI).

Vol. 3489: G.T. Heineman, I. Crnkovic, H.W. Schmidt, J.A. Stafford, C. Szyperski, K. Wallnau (Eds.), Component-Based Software Engineering. XI, 358 pages. 2005.

Vol. 3488: M.-S. Hacid, N.V. Murray, Z.W. Raś, S. Tsumoto (Eds.), Foundations of Intelligent Systems. XIII, 700 pages. 2005. (Subseries LNAI).

Vol. 3486: T. Helleseth, D. Sarwate, H.-Y. Song, K. Yang (Eds.), Sequences and Their Applications - SETA 2004. XII, 451 pages. 2005.

Vol. 3483: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part IV. LXV, 1362 pages. 2005.

Vol. 3482: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part III. LXV, 1340 pages. 2005.

Vol. 3481: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part II. LXV, 1316 pages. 2005.

Vol. 3480: O. Gervasi, M.L. Gavrilova, V. Kumar, A. Laganà, H.P. Lee, Y. Mun, D. Taniar, C.J.K. Tan (Eds.), Computational Science and Its Applications – ICCSA 2005, Part I. LXV, 1234 pages. 2005.

Vol. 3479: T. Strang, C. Linnhoff-Popien (Eds.), Location- and Context-Awareness. XII, 378 pages. 2005.

Vol. 3469: A.H. Wright, M.D. Vose, K.A. De Jong, L.M. Schmitt (Eds.), Foundations of Genetic Algorithms. X, 315 pages. 2005

Preface

The 8th Workshop on the Foundations of Genetic Algorithms, FOGA-8, was held at the University of Aizu in Aizu-Wakamatsu City, Japan, January 5–9, 2005. This series of workshops was initiated in 1990 to encourage further research on the theoretical aspects of genetic algorithms, and the workshops have been held biennially ever since. The papers presented at these workshops are revised, edited and published as volumes during the year following each workshop. This series of (now eight) volumes provides an outstanding source of reference for the theoretical work in this field.

At the same time this series of volumes provides a clear picture of how the theoretical research has grown and matured along with the field to encompass many evolutionary computation paradigms including evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP), as well as the continuing growth in interactions with other fields such as mathematics, physics, and biology.

A tradition of these workshops is organize them in such a way as to encourage lots of interaction and discussion by restricting the number of papers presented and the number of attendees, and by holding the workshop in a relaxed and informal setting. This year's workshop was no exception. Thirty-two researchers met for 3 days to present and discuss 16 papers. The local organizer was Lothar Schmitt who, together with help and support from his university, provided the workshop facilities.

After the workshop was over, the authors were given the opportunity to revise their papers based on the feedback they received from the other participants. It is these revised papers that are included in this volume and follow the order in which they were presented at the workshop. In addition to these 16 papers, there were 2 invited talks: an opening presentation by Alden Wright and a closing presentation by Kenneth De Jong. These slides-only presentations are not included in this volume, but can be obtained from the authors upon request. A brief summary of these presentations is provided here.

Alden Wright opened the workshop with a presentation titled "Can Evolutionary Computation Theory Have Significance Outside of EC?" and subtitled "Can EC Theory Help Us To Understand Evolution?". The field of artificial life has been successful in reaching a wide audience with claims that artificial life experiments can give insight into natural evolution. Wright asked if EC theory can do the same? He proposed that EC theory might be relevant to some challenges in evolutionary research[1]. These included:

- Analysis of the evolution of rates of mutation and recombination. Do "optimal" rates evolve?
- Analysis of the evolution of the information content of genomes.

[1] Some of these challenges came from the website:
`http://evonet.sdsc.edu/evoscisociety/chall_and_oppors_in_e_res.htm`

- Analysis of genic selection and of conflict within genomes (e.g., segregation distortion, evolution of gene expression, etc.).
- How does evolution maintain the great complexity of organisms while also allowing "rapid" evolution in some areas?
- How is it possible that phenotypic variations do not destroy brittle interactions between the subsystems of an organism while still allowing for the variablity that allows for evolutionary innovations?

Wright suggested that investigation of the genotype-phenotype map might give insight into the last two challenges.

Kenneth De Jong closed the workshop with a presentation titled "Unifying EC Theory." In this presentation he argued that developing a more unified framework for EC theory was important for further progress in the field. This was based on the observation that historically the field has evolved around a number of EA demes (GAs, ESs, etc.), resulting in deme-specific terminology and theory. We now have deme-independent EC toolkits that provide creative mixing and matching of EA components, but we have no theory to guide EA design at this level.

De Jong outlined a strategy for developing such a theory. He suggested that we need to clearly distinguish between EAs as dynamical systems and EAs as problem solvers. Adopting a dynamical systems view allows us to answer questions about trajectories, fixed points, etc., and makes contact with a large body of existing theoretical work in evolutionary biology, evolutionary game theory, and dynamical systems theory. Adopting a problem-solving view allows us to answer questions about the effectiveness of EAs for optimization, search, machine learning, etc., and makes contact with a large body of existing theoretical work from computer science, operations research, and artificial intelligence.

De Jong argued that in both of these areas it is important to find a middle ground between theories that are too abstract to be helpful and too specific to be applicable to new situations. He gave several examples of how that might be done using a top-down strategy. He concluded by noting that several of the papers presented were nice examples of this middle theoretical ground, and expressed the hope that he would see more of them at the next FOGA.

In between those 2 presentations 16 papers were presented on a wide range of theoretical evolutionary computation topics. We hope that you find them as interesting and provocative as we did. We fully expect that these papers will serve as a catalyst for further progress to be reported at the next FOGA workshop in 2007.

March 2005

Alden Wright
Michael Vose
Kenneth De Jong
Lothar Schmitt

Organization

Program Co-chairs

Alden Wright (University of Montana, Missoula, USA)
Michael Vose (University of Tennessee, USA)
Kenneth De Jong (George Mason University, USA)
Lothar Schmitt (University of Aizu, Japan)

Reviewers

Lee Altenberg
Anne Auger
Thomas Baeck
Wolfgang Banzhaf
Hans-Georg Beyer
Jürgen Branke
Anthony Bucci
Forbes Burkowski
Kenneth De Jong
Rolf Drechsler
Stefan Droste
Anton Eremeev
Larry Eshelman
James Foster
Marcus Gallagher
Oliver Giel
Jens Gottlieb
Walter Gutjahr
Jonathan Hallam
Nikolaus Hansen
William Hart
Jun He
Robert Heckendorn
Jeff Horn
Christian Igel
Yaochu Jin
Jens Jägersküpper
Thomas Jansen
Tim Kovacs
William Langdon
Anthony Liekens
Sean Luke
Evelyne Lutton
Keith Matthias
Nicholas McPhee
Peter Merz
Martin Middendorf
Heinz Mühlenbein
Bart Naudts
Silja Meyer-Nieberg
Una-May O'Reilly
Riccardo Poli
Adam Prugel-Bennett
Colin Reeves
Franz Rothlauf
Jonathan Rowe
Günter Rudolph
Dave Schaffer
Lothar Schmitt
Marc Schoenauer
Hans-Paul Schwefel
Jonathan Shapiro
Jim Smith
William Spears
Peter Stadler
Chris Stephens
Matthew Streeter
Dirk Thierens
Marc Toussaint
Ingo Wegener
Karsten Weicker
Nicole Weicker
Darrell Whitley
Paul Wiegand
Carsten Witt
Alden Wright
Annie Wu
Xin Yao

Sponsoring Institutions

International Society for Genetic and Evolutionary Computation (ISGEC)
University of Aizu, Japan

University of Aizu Institutional Support

President T. Ikegami, Vice-President S. Tsunoyama
Professor R.H. Fujii, Professor L. Pichl
K. Ishikawa, Y. Nagashima, M. Nanaumi
K. Doi, S. Ito, M. Oouchi, K. Takeyasu

FOGA 2005 Poster

FOGA 2005 Participants

Table of Contents

Genetic Algorithms for the Variable Ordering Problem of Binary Decision Diagrams

Wolfgang Lenders* and Christel Baier

Universität Bonn, Institut für Informatik I, Römerstrasse 164, 53117 Bonn, Germany
wolfgang@lenders.it, baier@cs.uni-bonn.de

Abstract. Ordered binary decision diagrams (BDDs) yield a data structure for switching functions that has been proven to be very useful in many areas of computer science. The major problem with BDD-based calculations is the variable ordering problem which addresses the question of finding an ordering of the input variables which minimizes the size of the BDD-representation. In this paper, we discuss the use of genetic algorithms to improve the variable ordering of a given BDD. First, we explain the main features of an implementation and report on experimental studies. In this context, we present a new crossover technique that turned out to be very useful in combination with sifting as hybridization technique. Second, we provide a definition of a distance graph which can serve as formal framework for efficient schemes for the fitness evaluation.

1 Introduction

Ordered binary decision diagrams (BDDs for short) are data structures to represent switching functions that rely on a compactification of binary decision trees. More general, using appropriate binary encodings, BDDs can serve to represent discrete functions with a finite domain. They were first introduced by Lee [28] and Akers [1]. In the meantime, various variants of BDDs have been suggested in the literature and applied successfully in many areas of computer science. Most popular are Bryant's *(reduced) ordered binary decisions diagrams* [8] that require a fixed variable ordering on any path. They have been proven to be very useful for the verification of reactive systems, often called *symbolic model checking* [10, 32]. Other application areas of BDDs include VLSI design, graph algorithms, complexity theory, matrix-operations, data bases, artificial intelligence, and many more. See e.g. the text books [18, 25, 33, 36, 48].

The crucial point with ordered BDD-based computations is the *variable ordering problem*. For a wide range of switching functions, there are polynomial-sized BDDs for "good" variable orderings, while the BDDs under "bad" variable orderings have exponential size. Unfortunately, the problem of finding an optimal variable ordering is NP-complete [6, 45]. However, there are many reordering algorithms that improve the ordering of a given BDD. Most popular are Rudell's sifting algorithm [41] and the window permutation algorithm [21]. A first attempt to use genetic algorithms for the variable ordering problem for BDDs was presented by Drechsler, Becker and Göckel [15] where the main genetic operations are partially-mapped crossover and mutation.

* The paper is based on material of the diploma thesis by the first author Wolfgang Lenders which he submitted in August 2004 at the Department of Computer Science, Universität Bonn.

A.H. Wright et al. (Eds.): FOGA 2005, LNCS 3469, pp. 1–20, 2005.

A related approach using simulated annealing was suggested by Bollig, Löbbing and Wegener [5]. In experimental studies it turned out that these methods yield better results (smaller BDDs) than other dynamic reordering techniques, but they are comparably slow, see e.g. [42]. To speed up the computations, several approaches have been suggested, including advanced tricks for the parameter setting and treating sifting as a genetic operation that replaces crossover techniques [16, 46], evolutionary algorithms with learning heuristics [17], the use of computed tables and approximate fitness values [24] or parallel genetic algorithms [12].

The goal of our paper is orthogonal to the above mentioned strategies by presenting alternative techniques to improve the efficiency and quality of genetic reordering algorithms for BDDs, while still retaining the concept of crossover (in contrast to the approaches of [16, 46]). We concentrate here on the purely genetic part of such reordering algorithms. However, the techniques suggested here can easily be combined with other (non-genetic) methods to increase the efficiency, e.g. by using "ordinary" sifting as in [16, 46].

Unlike [16, 46] which uses inversion as the only genetic recombination technique, we discuss several crossover techniques and present a new one, called *alternating crossover* which attempts to maximize the benefits of hybridization, i.e., the combination of a deterministic search algorithm with a genetic algorithm. The idea in the context of BDD minimization relies in generating an interleaving of the parent's variable orderings (alternating crossover) and moving the variables with the sifting-technique to the next local optimum after (the hybridization step). Our experimental results show that alternating crossover outperforms other recombination techniques such as order, partially matched or cycle crossover and inversion, by means of the BDD-sizes, while no significant differences in the runtime could be observed.

The second contribution is a formal framework to speed up the calculation of the fitness values for the newly generated individuals. In fact, for the variable ordering problem, calculating the BDD-size under a given variable ordering is a time-consuming step. It is typically realized by a sequence of local (level-wise) reorganizations of the BDD, the so called *swap*-operator (see e.g. [48]). Even when the final BDD is smaller than the original one, an exponential blow-up for the intermediate BDDs is possible. Thus, strategies that support the fitness calculation of the new population are highly desirable. We introduce a formal notion of a *distance graph*, a weighted graph where the nodes are orderings and the edges are labeled with the minimal number of swaps necessary to transform one ordering into another one. Using (variants of) heuristic algorithms for the traveling salesperson problem a "short" tour in the distance graph through the newly generated orderings, for which the fitness values (BDD-sizes in our case) are required, yields an appropriate scheme for the fitness evaluation. The distance graph can also serve as formal framework for other techniques that support the fitness calculation as suggested in [24]. Moreover, the fitness computation via our visiting strategy can easily be modified to weaken the drawback of crossover operations that might lead to unfeasible BDD-sizes, e.g., if they generate individuals that are far from both parents and combine the bad attributes of the parents.

Throughout the paper, we concentrate on the use of our algorithm for the minimization of ordinary BDDs, but our methods are also applicable to other types of decision

diagrams, such as zero suppressed BDDs [36] algebraic decision diagrams, [2, 11] and their normalized version [39], and other DD-variants.

Organization of the paper. The basic concepts of binary decision diagrams and notations used in this paper are summarized in Section 2. Section 3 explains the main concepts of our genetic algorithm and its implementation we used for the experimental studies. Section 4 is concerned with alternating crossover. Our graph-based technique to reduce the runtime for the fitness calculation are described in Section 5. In Section 6, we report on experimental results. Section 7 concludes the paper.

2 Binary Decision Diagrams

In the remainder of this paper, we fix a finite set $\mathcal{Z} = \{z_1, \ldots, z_n\}$ of boolean variables and often refer to the variables by their indices (i.e., we identify index i with variable z_i). An evaluation for $\mathcal{Z}$ denotes a function that assigns a boolean value (0 or 1) to any variable $z_i \in \mathcal{Z}$. By a *switching function* over $\mathcal{Z}$, we mean a function f which maps any evaluation for $\mathcal{Z}$ to 0 or 1. If $z \in \mathcal{Z}$ then $f|_{z=0}$ and $f|_{z=1}$ denote the *cofactors* of f which arise by fixing the assignment $z \mapsto 0$ and $z \mapsto 1$ respectively. For instance, if $f = z_1 \wedge (z_2 \vee z_3)$ then $f|_{z_1=0} = 0$ and $f|_{z_1=1} = z_2 \vee z_3$.

The fact that there is no data structure for switching functions that is efficient for all switching functions becomes clear from the observation that the number of switching functions over $\mathcal{Z}$ grows double exponentially in the size of $\mathcal{Z}$. An *explicit* representation of switching functions using truth tables seems coherent, but a truth table for a switching function with n variables consists of 2^n lines and consequently its space complexity grows exponentially in the number of variables. *Implicit* descriptions, like propositional logic formulas and binary decision diagrams can be much more efficient.

Binary decision diagrams are a graph based representation of switching functions which rely on the decomposition of switching functions in their cofactors according to the *Shannon expansion* $f = (\neg z \wedge f|_{z=0}) \vee (z \wedge f|_{z=1})$. Formally, a BDD is an acyclic rooted directed graph where every inner node v is labeled with a variable and has two children, called the 0-successor and 1-successor. The terminal nodes are labeled with one of the truth values 0 or 1. In ordered BDDs (OBDD) [8], there is a variable ordering $\pi = (z_{i_1}, \ldots, z_{i_n})$ which is preserved on any path from the root to a terminal node. That is, if v is an inner node labeled with variable z_{i_ℓ} and w a child of v which is non-terminal and labeled with variable z_{i_r} then z_{i_ℓ} appears in π before z_{i_r}, i.e., $i_\ell < i_r$. In the sequel, we shall use the notation π-OBDD to denote an OBDD relying on the ordering π and we refer to any inner node labeled with variable z as a z-node.

The switching function represented by a terminal node agrees with the corresponding constant 0 or 1. The switching function of a z-node v with 0-successor w_0 and 1-successor w_1 is $f_v = (\neg z \wedge f_{w_0}) \vee (z \wedge f_{w_1})$. The switching function $f_{\mathcal{B}}$ represented by an OBDD $\mathcal{B}$ agrees with the switching function for its root node. Thus, given an evaluation for $\mathcal{Z}$, the truth value under $f_{\mathcal{B}}$ is obtained by traversing $\mathcal{B}$ starting in its root and branching in any inner node according to the given evaluation. Figure 1 depicts two π-OBDDs with the variable ordering $\pi = (z_1, z_2, z_3)$ for the function $f = (z_1 \wedge \neg z_2 \wedge z_3) \vee (\neg z_1 \wedge z_3 \wedge z_2)$. In the OBDD on the left, both z_3-nodes represent

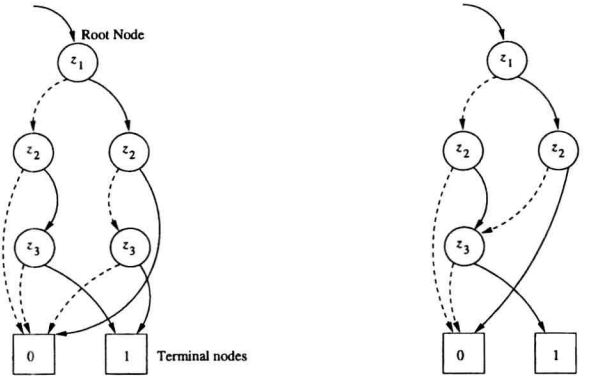

Fig. 1. OBDD and ROBDD

the same cofactor, namely $f|_{z_1=0,z_2=1} = f|_{z_1=1,z_2=0} = z_3$. Thus, a further reduction of the shown OBDD is possible by identifying the two z_3-nodes which yields the *reduced OBDD* (ROBDD) shown on the right. Intuitively, A OBDD is called reduced if it does not contain any redundancies. Formally, an ROBDD $\mathcal{B}$ denotes an OBDD such that $f_v \neq f_w$ for all nodes v, w in $\mathcal{B}$ with $v \neq w$. Given an π-OBDD, an equivalent π-ROBDD is obtained by identifying terminal nodes with the same value, identifying z-nodes with the same successors and eliminating all inner nodes where the 0- and 1-successor agree.

π-ROBDDs yield a *universal* representation for switching functions. (This follows from the fact that the above reduction procedure applied to the decision tree for a switching function with ordering π yields an π-ROBDD.) Moreover, the representation by π-ROBDDs is *canonical* up to isomorphism because the node-set of a π-ROBDD stands in one-to-one correspondence to the set of cofactors $f|_{z_{i_1}=b_1,\ldots,z_{i_k}=b_k}$ that can be obtained from f by assigning values to the "first" variables of π [1]. (Here, the range for k is $0,1,\ldots,n$, and $b_1,\ldots,b_k \in \{0,1\}$.)

ROBDDs yield a minimized OBDD-representation for a given switching function, provided the variable ordering is viewed to be fixed. However, by varying the ordering π the size of the BDD can be influenced. Figure 2 illustrates this observation by displaying two ROBDDs for the same switching function $f = (x_1 \wedge x_2) \vee (y_1 \wedge y_2) \vee (z_1 \wedge z_2)$ using different variable orderings. In the worst case, a ROBDD can have exponential size according to the number of variables n. There are functions, e.g. the middle bit of multiplication, whose ROBDD representation has exponential size for every variable ordering. Other functions, e.g. the most significant bit of addition, can vary between linear and exponential size depending on the chosen variable ordering while the number of any ROBDD for symmetric functions (e.g. n-ary disjunction or the parity function) is at most quadratic. See [9] and e.g. the text books [33, 48] for a detailed discussion of the complexity of ROBDDs.

Shared BDDs. Most BDD-packages follow the approach of [35] who suggested the simultaneous representation of several switching functions in one reduced graph (called

[1] Some of these cofactors might agree in which case they are represented by the same node

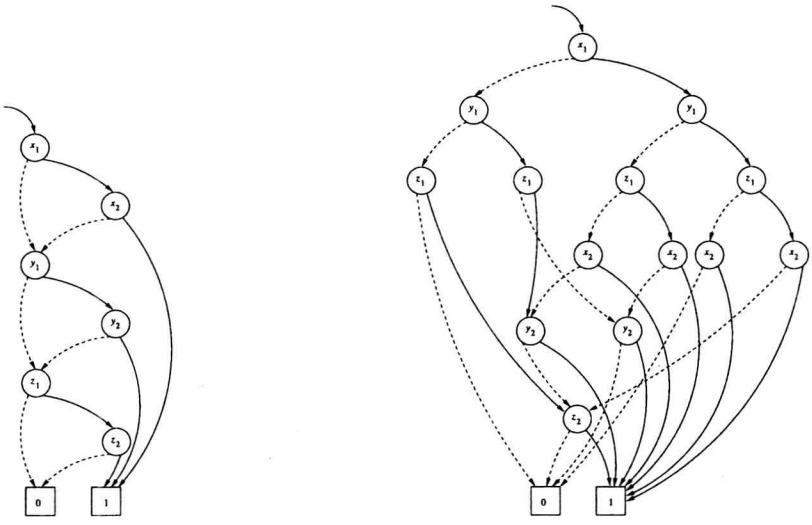

Fig. 2. Two BDDs for the same switching function using different variable orderings

shared or *multi-rooted BDD*) where the ROBDDs of the represented functions are realized as subgraphs and share the nodes for common cofactors. With several additional implementation tricks (appropriate hash tables, the ITE-operator to treat all boolean connectives, negated edges, etc.) the manipulation of switching functions and other BDD-based calculations can be realized efficiently, such as checking equivalence of switching functions in constant time or performing boolean combinations in time polynomial in the sizes of the ROBDDs for the arguments.

Throughout the paper the term BDD will refer to a shared BDD with negative edges. (This also applies for the number of BDD-nodes in the experimental results.)

The variable ordering problem. For the wide range of functions where the BDD-sizes range from polynomial to exponential, the variable ordering has an immense importance for BDD-applications, not only for reasons of memory requirement but also for the runtime of BDD manipulation operations. Beside some heuristics that compute a variable ordering from a given circuit description there is a wide range of *dynamic reordering* algorithms that attempt to improve the given variable ordering. The problem of finding an optimal variable ordering for a given BDD is known to be NP-complete [6, 45]. The best known algorithm that determines an optimal variable ordering requires exponential time [20]. However, there are several Greedy-heuristics that might return a suboptimal ordering. All these reordering algorithms are based on sequentially exchanging pairs of neighboring variables. This basic *swap* operation induces only local changes to the involved variables and can be carried out in constant time for each node that has to be handled. Thus, the running time of the operation $swap(z,z')$ on the BDD $\mathcal{B}$ with ordering π, where z and z' are adjacent in π, is linear in the number of z-nodes and the number of their incoming edges in $\mathcal{B}$. Using appropriate sequences of swap operations, any variable ordering can be transformed into another one.