Studies in Computational Intelligence, Volume 13

Editor-in-chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series
can be found on our homepage:
springer.com

Vol. 1. Tetsuya Hoya
Artificial Mind System – Kernel Memory Approach, 2005
ISBN 3-540-26072-2

Vol. 2. Saman K. Halgamuge, Lipo Wang (Eds.)
Computational Intelligence for Modelling and Prediction, 2005
ISBN 3-540-26071-4

Vol. 3. Bożena Kostek
Perception-Based Data Processing in Acoustics, 2005
ISBN 3-540-25729-2

Vol. 4. Saman K. Halgamuge, Lipo Wang (Eds.)
Classification and Clustering for Knowledge Discovery, 2005
ISBN 3-540-26073-0

Vol. 5. Da Ruan, Guoqing Chen, Etienne E. Kerre, Geert Wets (Eds.)
Intelligent Data Mining, 2005
ISBN 3-540-26256-3

Vol. 6. Tsau Young Lin, Setsuo Ohsuga, Churn-Jung Liau, Xiaohua Hu, Shusaku Tsumoto (Eds.)
Foundations of Data Mining and Knowledge Discovery, 2005
ISBN 3-540-26257-1

Vol. 7. Bruno Apolloni, Ashish Ghosh, Ferda Alpaslan, Lakhmi C. Jain, Srikanta Patnaik (Eds.)
Machine Learning and Robot Perception, 2005
ISBN 3-540-26549-X

Vol. 8. Srikanta Patnaik, Lakhmi C. Jain, Spyros G. Tzafestas, Germano Resconi, Amit Konar (Eds.)
Innovations in Robot Mobility and Control, 2005
ISBN 3-540-26892-8

Vol. 9. Tsau Young Lin, Setsuo Ohsuga, Churn-Jung Liau, Xiaohua Hu (Eds.)
Foundations and Novel Approaches in Data Mining, 2005
ISBN 3-540-28315-3

Vol. 10. Andrzej P. Wierzbicki, Yoshiteru Nakamori
Creative Space, 2005
ISBN 3-540-28458-3

Vol. 11. Antoni Ligêza
Logical Foundations for Rule-Based Systems, 2006
ISBN 3-540-29117-2

Vol. 13. Nadia Nedjah, Ajith Abraham, Luiza de Macedo Mourelle (Eds.)
Genetic Systems Programming, 2006
ISBN 3-540-29849-5

Nadia Nedjah
Ajith Abraham
Luiza de Macedo Mourelle
(Eds.)

Genetic Systems Programming

Theory and Experiences

Springer

Dr. Nadia Nedjah

Dr. Luiza de Macedo Mourelle

Faculdade de Engenharia
Universidade do Estado
do Rio de Janeiro
Rua São Francisco Xavier
524, 20550-900 Maracanã, Rio de Janeiro
Brazil
E-mail: nadia@eng.uerj.br

Dr. Ajith Abraham

School for Computer Science
and, Engineering
Chung-Ang University
Heukseok-dong 221
156-756 Seoul, Korea
Republic of (South Korea)
E-mail: ajith.abraham@ieee.org

Library of Congress Control Number: 2005936350

ISSN print edition: 1860-949X
ISSN electronic edition: 1860-9503
ISBN-10 3-540-29849-5 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29849-6 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Printed in The Netherlands

Typesetting: by the authors and TechBooks using a Springer LaTeX macro package

Printed on acid-free paper SPIN: 11521433 89/TechBooks 5 4 3 2 1 0

Foreword

The editors of this volume, Nadia Nedjah, Ajith Abraham and Luiza de Macedo Mourelle, have done a superb job of assembling some of the most innovative and intriguing applications and additions to the methodology and theory of genetic programming – an automatic programming technique that starts from a high-level statement of what needs to be done and automatically creates a computer program to solve the problem.

When the genetic algorithm first appeared in the 1960s and 1970s, it was an academic curiosity that was primarily useful in understanding certain aspects of how evolution worked in nature. In the 1980s, in tandem with the increased availability of computing power, practical applications of genetic and evolutionary computation first began to appear in specialized fields. In the 1990s, the relentless iteration of Moore's law – which tracks the 100-fold increase in computational power every 10 years – enabled genetic and evolutionary computation to deliver the first results that were comparable and competitive with the work of creative humans. As can be seen from the preface and table of contents, the field has already begun the 21st century with a cornucopia of applications, as well as additions to the methodology and theory, including applications to information security systems, compilers, data mining systems, stock market prediction systems, robotics, and automatic programming.

Looking forward three decades, there will be a 1,000,000-fold increase in computational power. Given the impressive human-competitive results already delivered by genetic programming and other techniques of evolutionary computation, the best is yet to come.

September 2005 *Professor John R. Koza*

Preface

Designing complex programs such as operating systems, compilers, filing systems, data base systems, etc. is an old ever lasting research area. Genetic programming is a relatively new promising and growing research area. Among other uses, it provides efficient tools to deal with hard problems by evolving creative and competitive solutions. Systems Programming is generally strewn with such hard problems. This book is devoted to reporting innovative and significant progress about the contribution of genetic programming in systems programming. The contributions of this book clearly demonstrate that genetic programming is very effective in solving hard and yet-open problems in systems programming. Followed by an introductory chapter, in the remaining contributed chapters, the reader can easily learn about systems where genetic programming can be applied successfully. These include but are not limited to, information security systems (see Chapter 3), compilers (see Chapter 4), data mining systems (see Chapter 5), stock market prediction systems (see Chapter 6), robots (see Chapter 8) and automatic programming (see Chapters 7 and 9).

In Chapter 1, which is entitled *Evolutionary Computation: from Genetic Algorithms to Genetic Programming*, the authors introduce and review the development of the field of evolutionary computations from standard genetic algorithms to genetic programming, passing by evolution strategies and evolutionary programming. The main differences among the different evolutionary computation techniques are also illustrated in this Chapter.

In Chapter 2, which is entitled *Automatically Defined Functions in Gene Expression Programming*, the author introduces the cellular system of Gene Expression Programming with Automatically Defined Functions (ADF) and discusses the importance of ADFs in Automatic Programming by comparing the performance of sophisticated learning systems with ADFs with much simpler ones without ADFs on a benchmark problem of symbolic regression.

In Chapter 3, which is entitled *Evolving Intrusion Detection Systems*, the authors present an Intrusion Detection System (IDS), which is a program that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. An IDS does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. The authors also evaluate the performances of two Genetic Programming techniques for IDS namely Linear Genetic Programming (LGP) and Multi-Expression Programming (MEP). They compare the obtained results with some machine learning techniques like Support Vector Machines (SVM) and Decision Trees (DT). The authors claim that empirical results clearly show that GP techniques could play an important role in designing real time intrusion detection systems.

In Chapter 4, which is entitled *Evolutionary Pattern Matching Using Genetic Programming*, the authors apply GP to the hard problem of engineering pattern matching automata for non-sequential pattern set, which is almost always the case in functional programming. They engineer good traversal orders that allow one to design an efficient adaptive pattern-matchers that visit necessary positions only. The authors claim that doing so the evolved pattern matching automata improves time and space requirements of pattern-matching as well as the termination properties of term evaluation.

In Chapter 5, which is entitled *Genetic Programming in Data Modelling*, the author demonstrates some abilities of Genetic Programming (GP) in Data Modelling (DM). The author shows that GP can make data collected in large databases more useful and understandable. The author concentrates on mathematical modelling, classification, prediction and modelling of time series.

In Chapter 6, which is entitled *Stock Market Modeling Using Genetic Programming Ensembles*, the authors introduce and use two Genetic Programming (GP) techniques: Multi-Expression Programming (MEP) and Linear Genetic Programming (LGP) for the prediction of two stock indices. They compare the performance of the GP techniques with an artificial neural network trained using Levenberg-Marquardt algorithm and Takagi-Sugeno neuro-fuzzy model. As a case study, the authors consider Nasdaq-100 index of Nasdaq Stock Market and the S&P CNX NIFTY stock index as test data. Based on the empirical results obtained the authors conclude that Genetic Programming techniques are promising methods for stock prediction. Finally, they formulate an ensemble of these two techniques using a multiobjective evolutionary algorithm and claim that results reached by ensemble of GP techniques are better than the results obtained by each GP technique individually.

In Chapter 7, which is entitled *Evolutionary Digital Circuit Design Using Genetic Programming*, the authors study two different circuit encodings used for digital circuit evolution. The first approach is based on genetic programming, wherein digital circuits consist of their data flow based specifications. In this approach, individuals are internally represented by the abstract trees/DAG of the corresponding circuit specifications. In the second approach, digital circuits are thought of as a map of rooted gates. So individuals are

represented by two-dimensional arrays of cells. The authors compare the impact of both individual representations on the evolution process of digital circuits. The authors reach the conclusion that employing either of these approaches yields circuits of almost the same characteristics in terms of space and response time. However, the evolutionary process is much shorter with the second linear encoding.

In Chapter 8, which is entitled *Evolving Complex Robotic Behaviors Using Genetic Programming*, the author reviews different methods for evolving complex robotic behaviors. The methods surveyed use two different approaches: The first one introduces hierarchy into GP by using library of procedures or new primitive functions and the second one uses GP to evolve the building modules of robot controller hierarchy. The author comments on including practical issues of evolution as well as comparison between the two approaches.

In Chapter 9, which is entitled *Automatic Synthesis of Microcontroller Assembly Code Through Linear Genetic Programming*, the authors focus on the potential of linear genetic programming in the automatic synthesis of microcontroller assembly language programs. For them, these programs implement strategies for time-optimal or sub-optimal control of the system to be controlled, based on mathematical modeling through dynamic equations. They also believe that within this application class, the best model is the one used in linear genetic programming, in which each chromosome is represented by an instruction list. The authors find the synthesis of programs that implement optimal-time control strategies for microcontrollers, directly in assembly language, as an attractive alternative that overcomes the difficulties presented by the conventional design of optimal control systems. This chapter widens the perspective of broad usage of genetic programming in automatic control.

We are very much grateful to the authors of this volume and to the reviewers for their tremendous service by critically reviewing the chapters. The editors would like also to thank Prof. Janusz Kacprzyk, the editor-in-chief of the Studies in Computational Intelligence Book Series and Dr. Thomas Ditzinger, Springer Verlag, Germany for the editorial assistance and excellent cooperative collaboration to produce this important scientific work. We hope that the reader will share our excitement to present this volume on **Genetic Systems Programming** and will find it useful.

Brazil
August 2005

Nadia Nedjah
Ajith Abraham
Luiza M. Mourelle

Contents

1 Evolutionary Computation: from Genetic Algorithms to Genetic Programming
Ajith Abraham, Nadia Nedjah, Luiza de Macedo Mourelle 1
1.1 Introduction 2
1.1.1 Advantages of Evolutionary Algorithms 3
1.2 Genetic Algorithms 3
1.2.1 Encoding and Decoding 4
1.2.2 Schema Theorem and Selection Strategies 5
1.2.3 Reproduction Operators 6
1.3 Evolution Strategies 9
1.3.1 Mutation in Evolution Strategies 9
1.3.2 Crossover (Recombination) in Evolution Strategies 10
1.3.3 Controling the Evolution 10
1.4 Evolutionary Programming 11
1.5 Genetic Programming 12
1.5.1 Computer Program Encoding 13
1.5.2 Reproduction of Computer Programs 14
1.6 Variants of Genetic Programming 15
1.6.1 Linear Genetic Programming 16
1.6.2 Gene Expression Programming (GEP) 16
1.6.3 Multi Expression Programming 17
1.6.4 Cartesian Genetic Programming 18
1.6.5 Traceless Genetic Programming (TGP) 18
1.6.6 Grammatical Evolution 19
1.6.7 Genetic Algorithm for Deriving Software (GADS) 19
1.7 Summary 19
References 19

2 Automatically Defined Functions in Gene Expression Programming
Cândida Ferreira 21
2.1 Genetic Algorithms: Historical Background 21
2.1.1 Genetic Algorithms 22
2.1.2 Genetic Programming 22
2.1.3 Gene Expression Programming 25
2.2 The Architecture of GEP Individuals 27
2.2.1 Open Reading Frames and Genes 28
2.2.2 Structural Organization of Genes 30
2.2.3 Multigenic Chromosomes and Linking Functions 32
2.3 Chromosome Domains and Random Numerical Constants 33
2.4 Cells and the Creation of Automatically Defined Functions 36
2.4.1 Homeotic Genes and the Cellular System of GEP 37
2.4.2 Multicellular Systems 37
2.4.3 Incorporating Random Numerical Constants in ADFs 39
2.5 Analyzing the Importance of ADFs in Automatic Programming 40
2.5.1 General Settings 40
2.5.2 Results without ADFs 42
2.5.3 Results with ADFs 46
2.6 Summary 54
References 55

3 Evolving Intrusion Detection Systems
Ajith Abraham, Crina Grosan 57
3.1 Introduction 57
3.2 Intrusion Detection 58
3.3 Related Research 60
3.4 Evolving IDS Using Genetic Programming (GP) 63
3.4.1 Linear Genetic Programming (LGP) 63
3.4.2 Multi Expression Programming (MEP) 64
3.4.3 Solution Representation 64
3.4.4 Fitness Assignment 66
3.5 Machine Learning Techniques 66
3.5.1 Decision Trees 67
3.5.2 Support Vector Machines (SVMs) 67
3.6 Experiment Setup and Results 68
3.7 Conclusions 77
References 77

4 Evolutionary Pattern Matching Using Genetic Programming
Nadia Nedjah, Luiza de Macedo Mourelle 81
4.1 Introduction 82
4.2 Preliminary Notation and Terminology 83
4.3 Adaptive Pattern Matching 86
4.3.1 Constructing Adaptive Automata 87
4.3.2 Example of Adaptive Automaton Construction 89
4.4 Heuristics for Good Traversal Orders 90
4.4.1 Inspecting Indexes First 90
4.4.2 Selecting Partial Indexes 90
4.4.3 A Good Traversal Order 91
4.5 Genetically-Programmed Matching Automata 92
4.5.1 Encoding of adaptive matching automata 93
4.5.2 Decoding of Traversal Orders 94
4.5.3 Genetic Operators 94
4.5.4 Fitness function 99
4.6 Comparative Results 101
4.7 Summary 102
References 103

5 Genetic Programming in Data Modelling
Halina Kwasnicka, Ewa Szpunar-Huk 105
5.1 Introduction 105
5.2 Genetic Programming in Mathematical Modelling 106
5.2.1 Adaptation of GP to Mathematical Modelling 107
5.2.2 An educational Example – the Hybrid of Genetic Programming and Genetic Algorithm 108
5.2.3 General Remarks 113
5.3 Decision Models for Classification Tasks 114
5.3.1 Adaptation of GP to Clasification Task 115
5.3.2 An Example of Classification Rules Discovering using GP . . . 117
5.3.3 Used Algorithm 117
5.3.4 General Remarks 120
5.4 GP for Prediction Task and Time Series Odelling 120
5.4.1 Adaptation of GP to Prediction Task 121
5.4.2 Adaptation of GP to Prediction Tasks 122
5.4.3 Hybrid GP and GA to Develope Solar Cycle's Model 125
5.4.4 General Remarks 127
5.5 Summary 129
References 129

6 Stock Market Modeling Using Genetic Programming Ensembles
Crina Grosan, Ajith Abraham . . . 131
6.1 Introduction . . . 131
6.2 Modeling Stock Market Prediction . . . 132
6.3 Intelligent Paradigms . . . 134
6.3.1 Multi Expression Programming (MEP) . . . 134
6.3.2 Linear Genetic Programming (LGP) . . . 135
6.3.3 Artificial Neural Network (ANN) . . . 136
6.3.4 Neuro-Fuzzy System . . . 138
6.4 Ensemble of GP Techniques . . . 138
6.4.1 Nondominated Sorting Genetic Algorithm II (NSGA II) . . . 139
6.5 Experiment Results . . . 140
6.5.1 Parameter Settings . . . 140
6.5.2 Comparisons of Results Obtained by Intelligent Paradigms . . . 143
6.6 Summary . . . 144
References . . . 144

7 Evolutionary Digital Circuit Design Using Genetic Programming
Nadia Nedjah, Luiza de Macedo Mourelle . . . 147
7.1 Introduction . . . 147
7.2 Principles of Evolutionary Hardware Design . . . 148
7.3 Circuit Designs = Programs . . . 152
7.3.1 Encoding . . . 152
7.3.2 Genetic Operators . . . 153
7.4 Circuit Designs = Schematics . . . 155
7.4.1 Encoding . . . 155
7.4.2 Genetic Operators . . . 158
7.5 Result Comparison . . . 161
7.6 Conclusion . . . 171
References . . . 171

8 Evolving Complex Robotic Behaviors Using Genetic Programming
Michael Botros . . . 173
8.1 Introducing Khepera Robot . . . 173
8.2 Evolving Complex Behaviors by Introducing Hierarchy to GP . . . 175
8.2.1 Genetic Programming with Subroutine Library . . . 176
8.2.2 Neural Networks and the Control of Mobile Robots . . . 179
8.2.3 Using Neural Networks as Elements of the Function Set . . . 181
8.2.4 Comments . . . 182
8.3 Evolving Complex Behaviors by Introducing Hierarchy to the Controller . . . 184

8.3.1 Subsumption Architecture . . . 184
8.3.2 Action Selection Architecture . . . 185
8.3.3 Using GP to Evolve Modules of Subsumption Architecture . . 186
8.3.4 Using GP to Evolve Modules of Action Selection Architecture . . . 188
8.4 Comments . . . 189
8.5 Summary . . . 190
References . . . 191

9 Automatic Synthesis of Microcontroller Assembly Code Through Linear Genetic Programming
Douglas Mota Dias, Marco Aurélio C. Pacheco, José F. M. Amaral . . . 193
9.1 Introduction . . . 194
9.2 Survey on Genetic Programming Applied to Synthesis of Assembly 195
9.2.1 JB Language . . . 195
9.2.2 VRM-M . . . 196
9.2.3 AIMGP . . . 196
9.2.4 GEMS . . . 197
9.2.5 Discussion . . . 197
9.3 Design with Microcontrollers . . . 198
9.3.1 Microcontrollers . . . 198
9.3.2 Time-Optimal Control . . . 199
9.4 Microcontroller Platform . . . 200
9.5 Linear Genetic Programming . . . 201
9.5.1 Evolution of Assembly Language Programs . . . 202
9.6 System for Automatic Synthesis of Microcontroller Assembly . . . 202
9.6.1 Evolutionary Kernel . . . 202
9.6.2 Plant Simulator . . . 206
9.6.3 Microcontroller Simulator . . . 206
9.6.4 A/D and D/A Converters . . . 207
9.6.5 Overview . . . 207
9.7 Case Studies . . . 208
9.7.1 Cart Centering . . . 208
9.7.2 Inverted Pendulum . . . 218
9.8 Summary . . . 225
References . . . 226

Reviewer List . . . 233

List of Figures

1.1 Flow chart of an evolutionary algorithm 2
1.2 Flow chart of basic genetic algorithm iteration 4
1.3 Roulette wheel selection 7
1.4 Types of crossover operators 8
1.5 A simple tree structure of GP 13
1.6 Illustration of crossover operator 14
1.7 Illustration of mutation operator in GP 15
2.1 Tree crossover in Genetic Programming. The arrows indicate the crossover points. 23
2.2 Tree mutation in Genetic Programming 24
2.3 Permutation in Genetic Programming 24
2.4 Illustration of a hypothetical event of point mutation in Genetic Programming 25
2.5 The flowchart of Gene Expression Programming. 27
2.6 Expression of GEP genes as sub-ETs 33
2.7 The overall structure of an S-expression 36
2.8 Expression of a unicellular system with three Automatically Defined Functions 38
2.9 Expression of a multicellular system with three Automatically Defined Functions 39
2.10 Expression of a multicellular system with Automatically Defined Functions containing random numerical constants. 41
3.1 Network protected by an IDS 59
3.2 A generic intrusion detection model 60
3.3 Relation between accuracy and number of generations: (a) normal mode (b) probe attacks 71
3.4 Relation between accuracy and number of generations: (c) DoS attacks (d) U2R attacks 71
3.5 Relation between accuracy and number of generations: (e) R2L attacks 72

3.6 Relationship between the best result obtained for training data set and the average of results obtained for training /test data: (a) normal mode (b) probe attacks 72
3.7 Relationship between the best result obtained for training data set and the average of results obtained for training /test data: (c) DoS attacks (d) U2R attacks 73
3.8 Relationship between the best result obtained for training data set and the average of results obtained for training /test data: (e) R2L attacks 73
3.9 Growth of program codes for normal mode 74
3.10 Growth of program codes for probe attacks 74
3.11 Growth of program codes for DOS attacks 75
3.12 Growth of program codes for U2R attacks 75
3.13 Growth of program codes for R2L attacks 76
4.1 An adaptive automaton for $\{1 : f\omega a\omega, 2 : f\omega\omega a, 3 : f\omega g\omega\omega g\omega\omega\}$ 89
4.2 Simplified representation of the adaptive automaton of Figure 4.1 or corresponding traversal order 93
4.3 Internal representation of the traversal order of Figure 4.2 94
4.4 Another possible traversal order for Π 95
4.5 An adaptive automaton for $\{1 : f\omega a\omega, 2 : f\omega\omega a, 3 : f\omega g\omega\omega g\omega\omega\}$ 95
4.6 Single-point crossover of traversal orders 96
4.7 Double-point crossover of traversal orders 97
4.8 Mutation of non-final state to another non-final state 98
4.9 Mutation of non-final state to a final state 98
4.10 Mutation of final state to another final state 99
5.1 An examplary tree in GP and the coded expression 107
5.2 An example of crossover in TA 113
5.3 The scheme of the GP in conjunction with GA used as TA algorithms 113
5.4 An exemplary tree in GESTIN algorithm 123
5.5 The scheme of GESTIN algorithm 123
5.6 Changes in Sun activity 125
5.7 Data preprocessing 126
5.8 Sunspot approximation – short-time test 127
5.9 The best approximation of sunspots – short-time test 127
5.10 The best approximation of sunspots – long-time test 128
6.1 Training and test data sets for Nasdaq-100 Index 133
6.2 Training and test data sets for NIFTY index 133
6.3 Architecture of ANFIS 138
7.1 VHDL data-flow specification for the circuit whose behaviour is given in Table 7.2 151
7.2 Chromosome with respect to the first encoding 152
7.3 Single-point crossover of circuit specifications 153
7.4 Double-point crossover of circuit specifications 154
7.5 The impact of the crossover operator for circuit specifications .. 154

7.6 Operand node mutation for circuit specification 156
7.7 Operator node mutation for circuit specification 156
7.8 Chromosome with respect to the second encoding 157
7.9 Four-point crossover of circuit schematics 158
7.10 Triple-point crossover of circuit schematics 159
7.11 Double-point crossover of circuit schematics 159
7.12 Single-point crossover of circuit schematics 160
7.13 The impact of the crossover operator for circuit schematics 161
7.14 Mutation of circuit schematics 162
7.15 Encoding 1: evolved circuit for the benchmark a 163
7.16 Encoding 1: Data-Flow specification of the evolved circuit for the benchmark a 164
7.17 Encoding 1: evolved circuit for the benchmark b 164
7.18 Encoding 1: Data-flow specifcation of the evolved circuit for the benchmark b 164
7.19 Encoding 1: evolved circuit for the benchmark c in Table 7.6 . . . 165
7.20 Encoding 1: Data-flow specification of the evolved circuit for the benchmark c 165
7.21 Encoding 1: evolved circuit for the benchmark d 166
7.22 Encoding 1: Data-flow specification of the evolved circuit for the benchmark d 166
7.23 Encoding 2: evolved circuit for the benchmark a in Table 7.6 . . . 167
7.24 Encoding 2: Data-flow specification of the evolved circuit for the benchmark a 167
7.25 Encoding 2: evolved circuit for the benchmark b in Table 7.6 . . . 167
7.26 Encoding 2: Data-flow specification of the evolved circuit for the benchmark b 168
7.27 Encoding 2: evolved circuit for the benchmark c in Table 7.6 . . . 168
7.28 Encoding 2: Data-flow specification of the evolved circuit for the benchmark c 168
7.29 Encoding 2: evolved circuit for the benchmark d in Table 7.6 . . . 169
7.30 Encoding 2: Data-flow specification of the evolved circuit for the benchmark d 169
7.31 Fitness comparison 170
7.32 Performance Comparison 170
8.1 Miniature mobile robot Khepera (with permission of K-team) . . 174
8.2 The position of the eight sensors on the robot (with permission of K-team) 174
8.3 Example of program tree in conventional GP and GP implementing ADF 176
8.4 Environment of the experiment 178
8.5 Model of a single neuron 179
8.6 Artificial neural network with three layers 180
8.7 Subsumption architecture 184
8.8 Action selection architecture 185

8.9 Action selection architecture with tree-like structure 186
8.10 Khepera robot with the gripper mounted on the top of it (with permission of K-team) 187
9.1 Structure of a program individual in AIMGP 197
9.2 The GEMS crossover and mutation process 198
9.3 Block diagram of the system 203
9.4 Crossover of two individuals 204
9.5 Mutation on an individual 205
9.6 Operation of the synthesis system 208
9.7 The cart-centering problem controlled by a PIC 209
9.8 Flowchart of the evaluation function for the cart-centering problem 212
9.9 Evolution of the cart-centering problem, without control over the length of the individuals 215
9.10 Evolution of the cart-centering problem, with control over the length of the individuals 216
9.11 The inverted pendulum problem controlled by a PIC 219
9.12 Evolution of the inverted pendulum problem, with control over the length of the individuals. 222

List of Tables

2.1 Settings for the sextic polynomial problem using a unigenic system with (**ugGEP-RNC**) and without (**ugGEP**) random numerical constants 43
2.2 Settings for the sextic polynomial problem using a multigenic system with (**mgGEP-RNC**) and without (**mgGEP**) random numerical constants 45
2.3 Settings and performance for the sextic polynomial problem using a unicellular system encoding 1, 2, 3, and 4 ADFs 48
2.4 Settings and performance for the sextic polynomial problem using a unicellular system encoding 1, 2, 3, and 4 ADFs with random numerical constants 50
2.5 Settings and performance for the sextic polynomial problem using a multicellular system encoding 1, 2, 3, and 4 ADFs 51
2.6 Settings and performance for the sextic polynomial problem using a multicellular system encoding 1, 2, 3, and 4 ADFs with random numerical constants 53
3.1 Variables for intrusion detection data set 69
3.2 Parameter settings for LGP 70
3.3 Parameters used by MEP 76
3.4 Functions evolved by MEP 76
3.5 Performance comparison 77
4.1 Space and time requirements for miscellaneous benchmarks 102
5.1 Sample of test functions and number of generations needed to find that function by the algorithm 111
5.2 The result given by GP+GA for different number of additional variables 111
5.3 Obtained result in Minerals classification [16] 119
5.4 Indexes in outer nodes 122
5.5 Results of 10 different strategies for one learning and three testing time periods of different value of WIG index 124
6.1 Values of parameters used by MEP 141

6.2 Performance measures obtained by MEP or population sizes . . . 141
6.3 Performance measures obtained by MEP for different chromosome lengths . . . 142
6.4 LGP parameter settings . . . 142
6.5 Parameters values used by NSGAII for ensembling MEP and LGP . . . 142
6.6 Results obtained by intelligent paradigms for Nasdaq and Nifty test data . . . 143
7.1 Gates symbols, number of gate-equivalent . . . 149
7.2 Truth table of the circuit whose schematics are given in Fig. 7.1 151
7.3 Performance comparison of the specification crossover variations 155
7.4 Chromosome for the circuit of Fig. 7.8 . . . 157
7.5 Performance comparison of the schematics crossover variations . 161
7.6 Truth table of the circuit used as benchmarks to compare both encoding methodologies . . . 163
7.7 Number of gate-equivalent and generations required to evolve the circuits presented . . . 169
7.8 Performance comparison of the impact of the studied encodings 170
9.1 Taxonomy over the evolution of assembly language programs . . . 196
9.2 Instruction subset of the PIC18F452 . . . 201
9.3 Summary of the evolutionary experiment with the cart-centering problem . . . 213
9.4 Summary of the evolutionary experiment with the inverted pendulum problem . . . 221