Practical Handbook of
GENETIC ALGORITHMS
Complex Coding Systems Volume III

Edited by Lance D. Chambers

CRC PRESS
Boca Raton London New York Washington, D.C.
Preface

This is the third and probably last book in the Practical Handbook of Genetic Algorithm series. The first volume dealt with applications, the second with new work in the field, and the present volume with computer code. I believe this set of three volumes completes main sections of the field that should be addressed, but I would be happy to be corrected on that score by anyone patient enough to let me know what other sectors should be covered.

Given the feedback I have received from readers of the two previous volumes, I am happy to present this third volume in the series to those who have an interest in the field. It presents computer code segments that can be downloaded from the CRC Press website and used by those undertaking research and/or are building real-world applications in the GA arena. It also contains detailed chapters covering the code and offering in-depth explanations and examples on the use of that code. A few chapters have been included because of their new/quirky uses of GAs; I believe readers will benefit from their content and perspectives. The remaining chapters contain material that is of more traditional value. They cover "problems" that are real and often encountered in the field.

I have not tested the code. You will need to do that either on your own or in collaboration with the author/s or others, if necessary.

The main reason for development of this volume was the apparent paucity of material containing GA code in a single reference volume. There are a few books, papers, and articles that have code segments but not a single work that is dedicated to it. There was perceived to be a need for a reference that offered an array of code covering a number of applications. This book responds to that need and is designed to be of value to programmers who do not want to start from the beginning in developing solutions to particular problems but would rather have a work to which they can refer for code to start the process in a serious manner or where a particular problem solution or computation approach is sought.

The code presented by each author is available at the CRC Press website at: www.crcpress.com and is available for downloading at no cost.

There is much to absorb and understand in this volume. If you have any particular queries on any of the material, contact details are given for each author on the first page of each contribution. Don't hesitate to get further help from the "horses mouth". Each of the authors would be happy to help; they are as proud of their contribution as I am to have edited them.

Hopefully, the chapters will also lead readers to new understandings and new ventures. There is still much work to be done in the application of GAs to the solution of problems that have, until now, been considered relatively intractable. The value that the GA fraternity has brought to academic, business, scientific, and social systems in the few years we have been in practice is something we should all view with pride. I do!

We are a young field with far to go and much to do and contribute. In less than three decades we have made a significant impact in many areas of human...
struggle and achievement and will continue to do so into the foreseeable future. It is my hope that this book will contribute to that progress.

I am very proud to have been able, with the help of CRC Press and their excellent and very supportive staff (I must particularly thank Nora and Bob - many thanks for all your help and support), to have had the opportunity to edit these three volumes and to have developed associations with authors of the chapters and some readers.

Lance Chambers
Perth, Western Australia
26\textsuperscript{th} Sept, 1998

Ichambers@transport.wa.gov.au

NOTE: Against the advice of my editor (even editors have editors) I have not Americanised (sic) the spelling of English spelling contributors. So as you read you will find a number of words with s's where many may have expected z's and you may also find a large number of u's where you might least expect them as in the word 'colour' and 'behaviour'. Please don't be perturbed. I believe each author has the right to see their work in a form they recognise as theirs. I have also not altered the referencing form employed by the authors.

Ultimately, however, I am responsible for all alterations, errors and omissions.
For Jenny, Lynsey, and Adrian
Thanks for the time and support.

All my love, Lance.
Contents

Chapter 1 A Lamarckian Evolution Strategy for Genetic Algorithms

1.1. Introduction
1.2. Implementation
   1.2.1 Basic implementation
1.3. Example Application
1.4. Discussion

Chapter 2 The Generalisation and Solving of Timetable Scheduling Problems

2.1. Introduction
   2.1.1 Aims
2.2. Introduction to Timetabling
   2.2.1 Definition of a Timetable
   2.2.2 Problems Related With Producing Timetables
   2.2.3 NP-Hardness of a Problem
2.3. A General Model for Timetabling Problems
   2.3.1 Generalising Timetable problems
   2.3.2 Hard Constraints on Resource Allocation
   2.3.3 Soft Constraints on Resource Allocation
   2.3.4 Timetable Templates
2.4. An Investigation Into Techniques for Producing Timetables
   2.4.1 Job-Shop Scheduling
   2.4.2 Graph Colouring
   2.4.3 Genetic Algorithms
   2.4.4 Choice of Algorithm for the General Timetable Problem
2.5. Genetic Algorithms
   2.5.1 Introduction to Evolution and Genetics
   2.5.2 Evolutionary Computing
      2.5.2.1 Solution Representation - The Chromosome Structure
      2.5.2.2 Initial Population
      2.5.2.3 Evaluation Functions
      2.5.2.4 Selection
      2.5.2.5 Recombination and Mutation operators
   2.6. A Genetic Algorithm For Our General Timetabling Problem.
      2.6.1 Chromosome Structure
      2.6.2 The Initial Population
      2.6.3 Evaluation Function
         2.6.3.1 Timetable Clashes
         2.6.3.2 Subject slot satisfaction

©1999 by CRC Press LLC
Chapter 3 Implementing Fast and Flexible Parallel Genetic Algorithms

3.1 Introduction
3.2 GAs and parallel computers
   3.2.1 Master-slave parallel GAs
   3.2.2 Multiple-population parallel GAs
3.3 Implementation
   3.3.1 Simple genetic algorithms
   3.3.2 Master-slave parallel GAs
   3.3.3 Multiple-deme parallel GAs
3.4 Test results
3.5 Conclusions and Extensions

Chapter 4 Pattern Evolver

4.1 Introduction
4.2 Mutation
4.3 Measuring Grayness
4.4 Adaptive Peaks as a Hindrance and Resilience as a Solution
4.5 A Sample Run
4.6 Local Maxima
4.7 Testing the Influence of Macromutations
4.8 Closing Remarks
4.9 How to Obtain Pattern Evolver

Chapter 5 Matrix-based GA Representations in a Model of Evolving Animal Communication

5.1 Introduction
5.2 Implementation
5.3 Results
Chapter 6 Algorithms to Improve the Convergence of a Genetic Algorithm with a Finite State Machine Genome

6.1. Introduction to the genetic algorithm
   6.1.1 A simple genetic algorithm
   6.1.2 Variations on the simple genetic algorithm
      6.1.2.1 Variations on fitness
      6.1.2.2 Variations on the selection process
      6.1.2.3 Crossover and mutation variations, and other operators
   6.1.3 Schema: definition and theory
      6.1.3.1 Schemata at work: an example
      6.1.3.2 Fitness landscape
      6.1.3.3 One strategy to reduce the chance of premature convergence
      6.1.3.4 Variations on Termination
   6.1.4 Reflections on the genetic algorithm

6.2. The benchmark
   6.2.1 The finite state machine as a genome
   6.2.2 The benchmark: description and theory
   6.2.3 The coding

6.3. The SFS algorithm
   6.3.1 SFS: description and coding
   6.3.2 Examples of SFS reorganization

6.4. The MTF algorithm
   6.4.1 MTF: description and coding
   6.4.2 Example of an MTF reorganization

6.5. Competition

6.6. Summary

6.7. Interesting ants bred during experimentation
   6.7.1 Best observed time: 181 time steps
   6.7.2 Intermediate FSMs
   6.7.3 8-state perfect scoring FSMs

6.8. Summary

Chapter 7 A Synthesizable VHDL Coding of a Genetic Algorithm

7.1. Introduction

7.2. A VHDL Primer
   7.2.1. Entities, Architectures and Processes
7.2.2. Finite State Machines
7.2.3. Data Types
7.2.4. Miscellany
7.3. The Design
7.3.1. The User-Controlled (Run-Time) Parameters
7.3.2. The VHDL Compile-Time Parameters
7.3.3. The Architecture
7.3.4. Inter-Module Communication
7.3.5. Component Modules
  7.3.5.1. Pseudorandom Number Generator (RNG)
  7.3.5.2. Shared Memory
  7.3.5.3. Memory Interface and Control Module (MIC)
  7.3.5.4. Population Sequencer (PS)
  7.3.5.5. Selection Module (SM)
  7.3.5.6. Crossover/Mutation Module (CMM)
  7.3.5.7. Fitness Module (FM)
7.3.6. Combining the Modules
7.4. Example Applications
  7.4.1. A Mathematical Function
  7.4.2. Logic Partitioning
  7.4.3. Hypergraph Partitioning
  7.4.4. Test Vector Generation
  7.4.5. Traveling Salesman and Related Problems
  7.4.6. Other NP-Complete Problems
7.5. Extensions of the Design
7.6. Summary and Related Work

References

Chapter 8 Genetic Algorithm Model Fitting

8.1 Introduction
8.2 Program Description
  8.2.1 The Genetic Algorithm
8.3 Application
8.4 Application of Program to Other Problems
  8.4.1 Different Mathematical Models
  8.4.2 Different Objective Functions and Fit Criteria
8.5 Conclusion
8.6 Program Listings

©1999 by CRC Press LLC
Chapter 9  A Hybrid Genetic Algorithm, Simulated Annealing and Tabu Search Heuristic for Vehicle Routing Problems with Time Windows

9.1 Introduction
9.2 Push-Forward Insertion Heuristic (PFIH)
9.3 Structure of Local Search Methods
  9.3.1 λ-interchange Generation Mechanism
  9.3.2 Evaluation of a Cost Move
  9.3.3 A λ-Interchange Local Search Descent Method (LSD)
9.4 Meta-heuristic Methods
  9.4.1 Simulated Annealing
  9.4.2 Tabu Search
  9.4.3 Genetic Algorithms
9.5 Computational Results
9.6 Summary and Conclusion

Acknowledgments

References

Appendix A

Chapter 10 Doing GAs with GAGS

10.1 Introduction
10.2 Creating chromosomes
10.3 Genetic Operators
10.4 Looking at chromosomes
10.5 Computing fitness
10.6 Population
10.7 Extending GAGS
  10.7.1 Extending views
  10.7.2 Extending genetic operators
10.8 Solving real problems with GAGS
  10.8.1 Optimizing Neural Networks
  10.8.2 Optimizing ad placement
  10.8.3 Playing Mastermind
10.9 Conclusion and future work

©1999 by CRC Press LLC
Chapter 11 Memory efficiency and speed enhancement code for Gas

11.1 Introduction
11.2 Background to the Code
11.3 Computer Memory
   11.3.1 Memory Allocation
   11.3.2 Shift Operators in C
11.4 Initialisation of Chromosomes
   11.4.1 Mutation Operator
   11.4.2 Single-point Crossover Operator
   11.4.3 The Value Extraction Function ithruj2int()
   11.4.4 Loading Integer Values Into Chromosomes
   11.4.5 Reporting Chromosome Contents
   11.4.6 Memory Size and Efficiency Savings
11.5 Conclusion

References

Chapter 12 Adaptive Portfolio Trading Strategies Using Genetic Algorithms

Abstract
12.1 Introduction
12.2 Portfolio Trading Learning Process - Overview
12.3 Performance Measurement / Fitness Measurement
12.4 APT Object Oriented Distributed Parallel Processing
12.5 Learning Process Implementation
   12.5.1 Market Timing Decision
   12.5.2 Risk and Portfolio Allocation Decisions
   12.5.3 Price Risk Calculation
   12.5.4 Portfolio Allocation Decision
   12.5.5 Portfolio Risk Decision
12.6 Adaptive Process Design
   12.6.1 Dynamic Parameter Optimisation
   12.6.2 Cross-Validation
   12.6.3 Top Fitness / Closest Fitness Behaviour Patterns
   12.6.4 Continuos Strategy Evaluation
   12.6.5 GA Parameters
12.7 Foreign Exchange Trading Portfolio Simulation
   12.7.1 Portfolio Specification
   12.7.2 Performance Result - Overview
   12.7.3 Performance Result - Consistency across Training / Application Periods
12.8 Results of Other Portfolio Types
12.9 Results of Different Fitness Targets and Variation of GA Parameters
   12.9.1 Different Fitness Target Calculations
   12.9.2 Variation of GA Parameters

Conclusion
Chapter 13 Population size, building blocks, fitness landscape and genetic algorithm search efficiency in combinatorial optimization: An empirical study

13.1 Introduction
   13.1.1 Related work

13.2 Genetic algorithm

13.3 Theory
   13.3.1 Risk estimation
   13.3.2 Number of generations

13.4 Problem set
   13.4.1 All are ones
   13.4.2 Maximum sum
   13.4.3 Tiling polyominoes
   13.4.4 3-SAT problem
   13.4.5 Folding snake

13.5 Experiments
   13.5.1 Fitness distributions
   13.5.2 Fitness landscape
   13.5.3 Search speed
   13.5.4 Population size
   13.5.5 Search efficiency model

13.6 Discussion and recommendations
   13.6.1 Comparisons to other results
   13.6.2 Autocorrelation analysis
   13.6.3 Sensitivity
   13.6.4 Recommendations

13.7 Conclusions

Chapter 14 Experimental Results on the Effects of Multi-Parent Recombination: An Overview

14.1 Introduction

14.2 Multi-Parent Reproduction Operators

14.3 The Effects of Higher Operator Arities

14.4 Conclusions

Acknowledgments

References
Appendix 1 An Indexed Bibliography of Genetic Algorithm

A1.1 Preface
   A1.1.1 Your contributions erroneous or missing?
   A1.1.2 How to get this report via Internet
   A1.1.3 Acknowledgement

A1.2 Indexes
   A1.2.1 Authors

A1.3 Bibliography

LIST OF TABLES

Table 6.1 Initial Population
Table 6.2 First Generation after Selection
Table 6.3 First Generation after Crossover
Table 6.4 First Generation after Mutation
Table 6.5 Four state FSM with start state q13
Table 6.6 FSM of Table 6.5 after Step 1 of SFS
Table 6.7 FSM of Table 6.6 after Next States for q0 Reassigned
Table 6.8 FSM of Table 6.5 after SFS
Table 6.9 FSM before SFS
Table 6.29 FSM of Table 6.9 after SFS
Table 6.30 Correlation of States between the FSMs
Table 6.31 FSM of Table 6.5 after MTF
Table 6.32 Template for a family of ants
Table 6.33 Template for a few perfect scoring ants
Table 6.34 Template for a 12-state Perfect Scoring Ants
Table 6.35 Substitutions for ### and @ @ @
Table 7.1 Evaluating assignments for $x_1$ and $x_2$ in the SAT GA
Table 8.1 User Inputs to GA Model-Fitting Program
Table 9.1 Average number of vehicles and distance
Table 9.2 Average number of vehicles and distance
Table 9.3 Comparison of the best known solutions
Table 9.4 Comparison of the optimal, best known, and TSSA solution
Table 9.5 Comparison of the best known solutions
Table 11.1 XOR operator truth table
Table 11.3 Variable value calculations
Table 12.1 Setup and Training Evaluation Periods
Table 12.2 Main parameter set
Table 12.3 Currency Pairs Available for Trading
Table 12.4 Trading Parameter/Risk Parameter Inputs
Table 12.5 Portfolio Performance Measurements
Table 12.6 Results of Selection of Top-Fitness Rule Set
Table 12.7 Results of Selection of Closest-Fitness Rule Set
Table 12.8 Top Fitness Rule Set - Training/Application Result Ratio
Table 12.9 Closest Fitness Rule Set
Table 13.1 The parameters of the genetic algorithm
Table 13.2 Population size
Table 13.3 An example of a polyomino tiling
Table 13.4 Approximate parameter values
Table 13.5 The average number of generations $n_G$
Table 13.6 Problems and the success rate $P$
Table 14.1 Optimal number of parents and corresponding success rates
Table 14.2 Characterization of the test functions
Table 14.3 Diagonal crossover with 6 parents vs. 1-point crossover

©1999 by CRC Press LLC
LIST OF FIGURES

Figure 1.1 City grid, and one route with distance of 70.14
Figure 1.2 Lamarckian experiment parameters
Figure 1.3 Performance graph (avg. 35 runs)
Figure 1.4 Optimized individuals
Figure 1.5 Extended performance graph (avg. 35 runs)
Figure 1.6 Individuals processed and CPU processing time
Figure 2.1 Example of a Timetable
Figure 2.2 Time Complexity Function
Figure 3.1 A schematic of a master-slave GA
Figure 3.2 A schematic of a multiple-deme parallel GA
Figure 3.3 The code for one generation of a simple genetic algorithm
Figure 3.4 The main loop of a GlobalGA
Figure 3.5 Code for evaluating the population in a master-slave GA
Figure 3.6 The code for one generation in a deme
Figure 4.1 The only thirteen perfectly smooth gray patterns
Figure 4.2 Examples of symmetrical but unsmooth patterns
Figure 4.3 An example of a micromutation
Figure 4.4 Two examples of macromutations
Figure 4.5 The eight horizontal halves
Figure 4.6 A diagram for determining how many points are deducted
Figure 4.7. Randomly seeded generation 0
Figure 4.8 Parents for generations 1, 2, 3, and 4
Figure 4.9 Generation 5
Figure 4.10 Parents for generations 6, 7, 8, and 9
Figure 4.11 Generation 10
Figure 4.12 Parents for generations 11, 12, 13, and 14
Figure 4.13 Generation 15
Figure 4.14 Parent for generation 16
Figure 4.15 The smoothest gray pattern
Figure 4.16 Number of runs per 1000 that discovered the global peak
Figure 5.1 Functional diagram of a single agent
Figure 5.2 Flowchart of the algorithm used for the simulation
Figure 5.3 Schematic of evolution of base population runs
Figure 5.4 Course of evolution of a base population run
Figure 5.5 Course of evolution of a population of size 30
Figure 5.6 Course of evolution of a population of size 10
Figure 5.7 Course of evolution of agents with 6 internal states and 6 observables
Figure 6.1a. Table of ten generations of sample run
Figure 6.1b. Corresponding chart summary of sample run
Figure 6.2 Comparison of runs with different phenotypes
Figure 6.3 Fitness landscape for a four-bit
Figure 6.4 Fitness landscape for a four-bit genotype
Figure 6.5 Equivalent FSMs
Figure 6.6 The John Muir Trail

©1999 by CRC Press LLC
Figure 6.7 The John Muir Trail as it appears to the ant
Figure 6.8 Trail following strategy favoring right turns
Figure 6.9 32-state/453-bit FSM genome map
Figure 6.10 Calculate $2^n$
Figure 6.11 Bitstring to Bytestring Conversion
Figure 6.12 Bytestring to Bitstring Conversion
Figure 6.13 Binary to Decimal Conversion
Figure 6.14 Decimal to Binary Conversion
Figure 6.15 Genotype to Phenotype Conversion
Figure 6.16 Phenotype to Genotype Conversion
Figure 6.17 Trail related parameters
Figure 6.18 Determine an ant's fitness
Figure 6.19 Output Successful FSMs
Figure 6.20 The main Part of the GA
Figure 6.21 Check for Contradictory Constant Values
Figure 6.22 Prints Headings on Output Files
Figure 6.23 Close Output Files
Figure 6.24 Initialization
Figure 6.25 Calculating and Outputting the Statistics for Each Generation
Figure 6.26 Breeding the Next Generation
Figure 6.27 Mating
Figure 6.28 Mutation
Figure 6.29 Listing of file with $st$ extension from sample run
Figure 6.30 Listing of file with $ps$ extension from sample run
Figure 6.32 Interchange States Based on Name Change
Figure 6.33 Reorganize parent pool
Figure 6.34 Second step of the SFS algorithm
Figure 6.35 The SFS Algorithm
Figure 6.36 Prints Headings on Output Files-Modified to Include SFS
Figure 6.37 The MTF Algorithm
Figure 6.39 Prints Headings on Output Files-Modified to Include SFS and MTF
Figure 6.40 An example of premature convergence
Figure 6.41 Example of an unsuccessful run
Figure 6.42 Competition Procedure
Figure 6.43 The procedures to calculate and output the statistics
Figure 6.44 Prints Headings on Output Files
Figure 6.45 Seed $= 0.41974869$ for benchmark case
Figure 6.46a Seed $= 0.41974869$ for SFS
Figure 6.46b Perfect score machine for seed $= 0.41974869$ in SFS
Figure 6.47a Seed $= 0.41974869$ for MTF
Figure 6.47b Perfect score machine for seed $= 0.41974869$ in MTF
Figure 6.48a Seed $= 0.41974869$ for benchmark with competition
Figure 6.48b Perfect score machine for seed $= 0.41974869$ for benchmark with competition
Figure 6.49 Seed $= 0.41974869$ for SFS with competition
Figure 6.50a Seed $= 0.41974869$ for MTF with competition
Figure 6.50b Perfect score machine for seed = 0.41974869 in MTF with competition
Figure 6.51 The John Muir Trail with Marked and Unmarked Trail Steps Labeled
Figure 6.52 12-state Perfect Scoring Ant with Time = 181
Figure 6.53 11-state Perfect Scoring FSM with Time = 188
Figure 6.54 10-state Perfect Scoring FSM with Time = 185
Figure 6.55 9-state Perfect Scoring FSM with Time = 187
Figure 6.56 8-state Perfect Scoring FSM with Time = 193
Figure 6.57 8-state Perfect Scoring FSM with Time = 199
Figure 7.2 Plot of Equation 1, the function $f(x_1, x_2)$

Figure 8.1. Top-Level Structure of SSHGA Program
Figure 10.1 Genetic operators and modes
Figure 11.1 Linear model of computer memory
Figure 11.2 Vertical model of computer memory
Figure 11.3 Memory model of example chromosome population
Figure 11.4 Chromosome representation
Figure 11.5 Results of chromosome initialisation procedure
Figure 11.7 Building the mask for the mutation operation
Figure 11.8 Result of using the mutation function
Figure 11.9 Building the mask for the crossover operation
Figure 11.10 Construction of two offspring
Figure 11.11 Chromosome used in examples of function ithruj2int()
Figure 11.12 Processing steps for the first 5-bit group
Figure 11.13 Processing steps for the 7-bit group
Figure 11.14 Memory reduction v chromosome length curves
Figure 12.1 Buy/sell decision flowchart
Figure 13.1 The toy snake folding problem
Figure 13.4 Distribution of evaluations
Figure 13.5 The average number of generations $n_G$