Contributors
Preface
Acknowledgments
Introduction
A Perspective on the Work in this Book
What is Evolutionary Computation?
Why is Evolutionary Computation interesting?
Styles of Evolutionary Computation
What defines Genetic Programming?
Current activity in Genetic Programming
Part II: Increasing the Power of Genetic Programming
Part III: Innovative Applications of Genetic Programming
Practical Guidance
It isn't as easy as it looks -- but it does work.
The fitness function is exceptionally important.
Representation is important too.
It all comes together in the transmission function.
Population size and diversity are also important.
Don't generalize from one run.
Genetic programming is robust.
Know your problem, know your data.
Where to go for more information and inspiration.
Biology
Complex Adaptive Systems
Genetic Algorithms and other Evolutionary Computation paradigms
Conclusion
Bibliography
Introduction to Genetic Programming
Introduction to Genetic Algorithms
Program Trees and the LISP Programming Language
Genetic Programming
Automatic Function Definition in Genetic Programming
Sources of Additional Information about Genetic Programming
Sources of Additional Information about Genetic Algorithms
Bibliography
Increasing the Power of Genetic Programming
The Evolution of Evolvability in Genetic Programming
Introduction
Evolvability
Representations
Evolving evolvability
Local vs. Global Function Definitions
Evolution of Function Definitions
Structural Regularity
Self-Crossover
Modular Crossover
Discussion of Structural Regularity
Conclusions
Further Work
Acknowledgments
Bibliography

The Donut Problem: Scalability, Generalization and Breeding Policies in Genetic Programming

Introduction: Depth vs. Breadth
The Donut Problem
Purposely Introduced Imperfections
There is a Solution (Sort of)

Breeding Policies
"Demes" and Spatially Distributed Evolution
Implementation of Distributed Evolution
Elitism and the Steady State Model
Implementation of Steady-State Elitism

Experimental Method
Performance of GP as Class Overlap Increases
Generalization and Uniform Undersampling of the Training Set
Generalization and Nonuniform Sampling of the Training Set
Assessing the Effects of Demes and Elitism

Summary of Experimental Configurations

Results
Scalability With Respect to Class Overlap
Generalization With Respect to Class Overlap
Generalization and Uniform Undersampling of Training Data
Generalization and Nonuniformly Distributed Training Data

Comparative Performance and the Optimal Function Set
Comparative Performance of Breeding Policies
Performance Across All Experiments
Performance Using Uniformly Sparse Training Data
Performance Using Nonuniform Training Data
Performance Using Sparse and Nonuniform Training Data
Performance Using Sparse Data With High Degree of Class Overlap
Performance Using Non-Sparse Data Sets

Conclusions

Conclusions About Distributed Evolution
Conclusions Concerning Elitism
Comments on the Procedures Used
Need for Benchmark Test Functions
Big Deme Grids and Other Parameters
Gene Frequencies and Distributed Evolution
Acknowledgments
Bibliography
Effects of Locality in Individual and Population Evolution
Overview
Background
Domain
Terminal Set
Function Set
Fitness Evaluation
Method
Discussion of Results
Population Seeding
Statistical Analysis
Emergence of Demes
Structural Analysis of Individuals
Recommendations for Future Work
Conclusions
Appendix A: Seed Tank Code
Appendix B: Original Example Tank Code
Appendix C: Stimulus-Response Maps of Example Tanks
Acknowledgments
Bibliography
The Evolution of Mental Models
Introduction
The Method and The Model
The Environment
The Implementation
Experimental Results
Discussion of Indexed Memory
Discussion of Mental Models
Future Work
Conclusions
Acknowledgments
Bibliography
Evolution of Obstacle Avoidance Behavior: Using Noise to Promote Robust Solutions

Introduction
Previous Work
Obstacle Avoidance as Genetic Programming
The Vehicle
The Obstacle Course
Sensors
Noise
Measuring Fitness in the Presence of Noise
Results
Future Work
Acknowledgments
Bibliography
Pygmies and Civil Servants
Introduction
The Problem Space
Genetic Programming Or String GA?
Implementation Notes
The Benefits of Elitism
Traditional Methods
The Fitness Function - Punish or Reward?
Early Results
Maintaining Diversity In Artificial Evolution
Sharing And Crowding
Isolation by Distance
Steady State Genetic Algorithms
Restricted Mating
Breeding For Secondary Features
Pygmies And Civil Servants
Implementation
Extending the model
Generalising the model
Pygmies and Genetic Programming
Conclusion
Future Work
Appendix: The Pygmy Algorithm
Acknowledgements
Bibliography
Genetic Programming Using a Minimum Description Length Principle
Introduction
GP using an MDL principle
Decision Trees and Genetic Programming
MDL-based fitness functions
Evolving decision trees
Evolving trees with an MDL-based fitness function
Conclusion
Bibliography
Genetic Programming in C++: Implementation Issues
Introduction
Pointer Based Implementations
A Postfix, Stack-Based Approach
Memory Efficiency
Manipulating Postfix Programs
Postfix Initialization
Postfix Crossover
Postfix Mutation
The Flow Control Problem with Postfix
Mixfix
Prefix Ordering
Initialization, Crossover and Mutation with Prefix
Handling Program Flow with Prefix
The Node Representation
General Data Support
The Opcode Format
The Jump Table Mechanism
The Prefix, Jump-Table (PJT) Approach
Results
Advanced Topics (Looking for Roadblocks)
Beyond Closure: Handling Multiple Data Types
Module Implementation
Encapsulation
Module Execution
Handling Recursion
Simulated Multi-Tasking
Using Tables to Evaluate Diversity
Conclusion and Future Directions
Acknowledgments
Bibliography
A Compiling Genetic Programming System that Directly Manipulates the Machine Code
Prologue
Reading Guidance
Introduction
The Compiling Genetic Programming System (CGPS)
The Hardware Environment
The Language for the Genetic Algorithm Implementation
The Structure of a Machine Code Function Callable by a 'C'-function
The SPARC Architecture
The Instruction Set
The Genetic Algorithm
Comparison between CGPS and interpreting GP Systems.
A Genetic Programming System for Heuristic Classification
Comparison between the CGPS and a Neural Network
The Sample Problem
The Training Set
Coding of Words for the Genetic Programming System.
Coding of words for the Neural Network
The Neural Network
Training Method
Results of comparison
Population Size and Efficiency
Applicability
Future Work
Concluding Remarks
Biblography
Innovative Applications of Genetic Programming
Automatic Generation of Programs for Crawling and Walking
Introduction
The problem
The approach
Functions and terminals
Side-effecting functions and simulated memory
Constant perturbation
Fitness evaluation
Program structure
Implementation
Results
Experiment 1
Experiment 2
Experiment 3
Analysis of the results
Analysis of the method
Comparison with random search
Comparison with other methods
Practical considerations
Scalability
The Automatic Generation of Plans for a Mobile Robot via Genetic Programming with Automatically Defined Functions

Introduction

The Genetic Planner

An Example World: A Robot on a 2-D Grid.

A set of procedurally-defined operators
A set of predicates that describe the world
Fitness functions for each of the predicates.
A ground goal expression
A simulation of the world

A Demonstration of The Genetic Planner

Automatically Defined Functions

Analysis of Some Best-Of-Run ADFs

Discussion

Future Work

Conclusions

Acknowledgments

References

Competitively Evolving Decision Trees Against Fixed Training Cases for Natural Language Processing

Introduction

The Domain: Word Sense Disambiguation

The Training Cases

How Decision Trees Work

Crossover Operations on Decision Trees

How Fixed Training Data Participate in Competitive Adaptation

Averting Overlearning with Decision Trees: Fitness Penalty

Conclusions

Non-trivial Learning and Generalization Performance

Competition

Fitness Penalty

Linguistic Data

Further Work

Acknowledgments

Bibliography

Cracking and Co-Evolving Randomizers

Background

Motivation

Arguments for Success

Two Player Penny Matching Game

Uniform Distribution

Models
Two Player Multi-Penny Matching Game
Fitness Measure
Single Generator
Separate Generators and Guessers
Sexing Populations
New Techniques
Dynamic Sampling
Fitness Tournament
Experiments
Tested Randomizers
Tableau
Functions and Terminals
GP Shell Modifications
Results
Conclusion
Acknowledgements
Bibliography

Optimizing Confidence of Text Classification by Evolution of Symbolic Expressions

Introduction
The News Story Classification Problem
Automated keyword assignment using MBR
The Coding Algorithm
The Referral Problem
Referral with single keyword per document
Referral with multiple keywords
Brief Overview of Genetic Algorithms
Example formulae
example results
Representation of evolved formulae
The environment for Genetic evolution
Generation of the initial random populations
Evaluating fitness
Fitness proportionate reproduction
Cross-over
Mutation
Population size
Number of generations
Results
The test environment
Discussion of results
Conclusions