# Genetic Programming

**William H. Hsu**
*Kansas State University, USA*

## INTRODUCTION

Genetic programming (GP) is a subarea of evolutionary computation first explored by John Koza (1992) and independently developed by Nichael Lynn Cramer (1985). It is a method for producing computer programs through adaptation according to a user-defined fitness criterion, or objective function.

GP systems and genetic algorithms (GAs) are related but distinct approaches to problem solving by simulated evolution. As in the GA methodology, GP uses a representation related to some computational model, but in GP, fitness is tied to task performance by specific program semantics. Instead of strings or permutations, genetic programs most commonly are represented as variable-sized expression trees in imperative or functional programming languages, as grammars (O'Neill & Ryan, 2001) or as circuits (Koza et al., 1999). GP uses patterns from biological evolution to evolve programs:

- **Crossover:** Exchange of genetic material such as program subtrees or grammatical rules.
- **Selection:** The application of the fitness criterion in order to choose which individuals from a population will go on to reproduce.
- **Replication:** The propagation of individuals from one generation to the next.
- **Mutation:** The structural modification of individuals.

To work effectively, GP requires an appropriate set of program operators, variables, and constants. Fitness in GP typically is evaluated over fitness cases. In data mining, this usually means training and validation data, but cases also can be generated dynamically using a simulator or be directly sampled from a real-world problem-solving environment. GP uses evaluation over these cases to measure performance over the required task, according to the given fitness criterion.

This article begins with a survey of the design of GP systems and their applications to data-mining problems, such as pattern classification, optimization of representations for inputs and hypotheses in machine learning, grammar-based information extraction, and problem transformation by reinforcement learning. It concludes with a discussion of current issues in GP systems (i.e., scalability, human-comprehensibility, code growth and reuse, and incremental learning).

## BACKGROUND

Although Cramer (1985) first described the use of crossover, selection, mutation, and tree representations for using genetic algorithms to generate programs, Koza, et al. (1992) is indisputably the field's most prolific and influential author (Wikipedia, 2004). In four books, Koza, et al. (1992, 1994, 1999, 2003) have described GP-based solutions to numerous toy problems and several important real-world problems.

- **State of the Field:** To date, GPs have been applied successfully to a few significant problems in machine learning and data mining, most notably symbolic regression and feature construction. The method is very computationally intensive, however, and it is still an open question in current research whether simpler methods can be used instead. These include supervised inductive learning, deterministic optimization, randomized approximation using non-evolutionary algorithms (i.e., Markov chain Monte Carlo approaches), genetic algorithms, and evolutionary algorithms. It is postulated by GP researchers that the adaptability of GPs to structural, functional, and structure-generating solutions of unknown forms makes them more amenable to solving complex problems. Specifically, Koza, et al. (1999, 2003) demonstrate that, in many domains, GP is capable of human-competitive automated discovery of concepts deemed to be innovative through technical review such as patent evaluation.

## MAIN THRUST

The general strengths of genetic programming lie in its ability to produce solutions of variable functional form, reuse partial solutions, solve multi-criterion optimization problems, and explore a large search space of solutions in parallel. Modern GP systems also are able to produce structured, object-oriented, and functional programming solutions involving recursion or iteration, subtyping, and higher-order functions.

A more specific advantage of GPs is their ability to represent procedural, generative solutions to pattern recognition and machine-learning problems. Examples of

Copyright © 2006, Idea Group Inc., distributing in print or electronic forms without written permission of IGI is prohibited.

this include image compression and reconstruction (Koza, 1992) and several of the recent applications surveyed in the following.

## GP for Pattern Classification

GP in pattern classification departs from traditional supervised inductive learning in that it evolves solutions whose functional form is not determined in advance, and in some cases can be theoretically arbitrary. Koza (1992, 1994) developed GPs for several pattern reproduction problems, such as the multiplexer and symbolic regression problems.

Since then, there has been continuing work on inductive GP for pattern classification (Kishore et al., 2000), prediction (Brameier & Banzhaf, 2001), and numerical curve fitting (Nikolaev & Iba, 2001). GP has been used to boost performance in learning polynomial functions (Nikolaev & Iba, 2001). More recent work on tree-based multi-crossover schemes has produced positive results in GP-based design of classification functions (Muni et al., 2004).

## GP for Control of Inductive Bias, Feature Construction, and Feature Extraction

GP approaches to inductive learning face the general problem of optimizing inductive bias: the preference for groups of hypotheses over others on bases other than pure consistency with training data or other fitness cases. Krawiec (2002) approaches this problem by using GP to preserve useful components of representation (features) during an evolutionary run, validating them using the classification data and reusing them in subsequent generations. This technique is related to the wrapper approach to knowledge discovery in databases (KDD), where validation data is held out and used to select examples for supervised learning or to construct or select variables given as input to the learning system. Because GP is a generative problem-solving approach, feature construction in GP tends to involve production of new variable definitions rather than merely selecting a subset.

Evolving dimensionally-correct equations on the basis of data is another area where GP has been applied. Keijzer and Babovic (2002) provide a study of how GP formulates its declarative bias and preferential (search-based) bias. In this and related work, it is shown that a proper units of measurement (strong typing) approach can capture declarative bias toward correct equations, whereas type coercion can implement even better preferential bias.

## Grammar-Based GP for Data Mining

Not all GP-based approaches use expression tree-based representations or functional program interpretation as the computational model. Wong and Leung (2000) survey data mining using grammars and formal languages. This general approach has been shown to be effective for some natural language learning problems, and extension of the approach to procedural information extraction is a topic of current research in the GP community.

## GP Software Packages: Functionality and Research Features

A number of GP software packages are available publicly and commercially. General features common to most GP systems for research and development include a very high-period random number generator, such as the Mersenne Twister for random constant generation and GP operations; a variety of selection, crossover, and mutation operations; and trivial parallelism (e.g., through multithreading).

One of the most popular packages for experimentation with GP is Evolutionary Computation in Java, or ECJ (Luke et al., 2004). ECJ implements the previously discussed features as well as parsimony, strongly-typed GP, migration strategies for exchanging individual subpopulations in island mode GP (a type of GP featuring multiple demes—local populations or breeding groups), vector representations, and reconfigurability using parameter files.

## Other Applications: Optimization, Policy Learning

Like other genetic and evolutionary computation methodologies, GP is driven by fitness and suited to optimization approaches to machine learning and data mining. Its program-based representation makes it good for acquiring policies by reinforcement learning.1 Many GP problems are error-driven or payoff-driven (Koza, 1992), including the ant trail problems and foraging problems now explored more heavily by the swarm intelligence and ant colony optimization communities. A few problems use specific information-theoretic criteria, such as maximum entropy or sequence randomization.

3 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/genetic-programming/10654

## Related Content

Domain-Driven Data Mining: A Practical Methodology
Longbing Caoand Chengqi Zhang (2008). *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications (pp. 831-848).*
www.irma-international.org/chapter/domain-driven-data-mining/7677

Multidimensional Anlaysis of XML Document Contents with OLAP Dimensions
Franck Ravat, Olivier Testeand Ronan Tournier (2009). *Progressive Methods in Data Warehousing and Business Intelligence: Concepts and Competitive Analytics (pp. 155-171).*
www.irma-international.org/chapter/multidimensional-anlaysis-xml-document-contents/28166

Biological Data Mining
George Tzanis, Christos Berberidisand Ioannis Vlahavas (2008). *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications (pp. 1696-1705).*
www.irma-international.org/chapter/biological-data-mining/7725

Database Queries, Data Mining, and OLAP
Lutz Hamel (2005). *Encyclopedia of Data Warehousing and Mining (pp. 339-343).*
www.irma-international.org/chapter/database-queries-data-mining-olap/10619

Hybrid Query and Data Qodering for Fast and Progressive Range-Aggregate Query Answering
Cyrus Shahabi, Mehrdad Jahangiriand Dimitris Sacharidis (2008). *Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications (pp. 1250-1268).*
www.irma-international.org/chapter/hybrid-query-data-qodering-fast/7697