Version 3 – June 25, 1996 for *Handbook of Evolutionary Computation*.

Future Work and Practical Applications of Genetic Programming

John R. Koza
Computer Science Department
Stanford University
258 Gates Building
Stanford, California 94305 USA
PHONE: 415-941-0336
FAX: 415-941-9430
E-MAIL: Koza@CS.Stanford.Edu
WWW ADDRESS: http://www-cs-faculty.stanford.edu/~koza/

ABSTRACT

Genetic programming is a relatively new domain-independent method for evolving computer programs to solve problems. This chapter suggests avenues for possible future research on genetic programming, opportunities to extend the technique, and areas for possible practical applications.

1. Introduction

The goal of the field of automatic programming is to create, in an automated way, a computer program that enables a computer to solve a problem. Genetic programming (Koza 1992, 1994) is a domain-independent approach to automatic programming in which computer programs are evolved to solve, or approximately solve, problems. The field of genetic programming has grown rapidly in the past few years. Between 1992 and 1996, over 600 papers on genetic programming have been published.

This paper discusses the many opportunities to apply genetic programming to realistic and practical problems, numerous possible avenues to extend the technique of genetic programming, and avenues for research on theoretical aspects of genetic programming.

2. Promising Application Areas

I believe the single most important area for future work in genetic programming (as well as for all other techniques of automated machine learning) is to demonstrate the applicability of the technique to realistic problems.

The presence of some or all of the following characteristics make an area especially suitable for the application of genetic programming:

- an area where conventional mathematical analysis does not, or cannot, provide analytic solutions,
- an area where the interrelationships among the relevant variables are poorly understood (or where it is suspected that the current understanding may well be wrong),
- an area where finding the size and shape of the ultimate solution to the problem is a major part of the problem,
- an area where an approximate solution is acceptable (or is the only result that is ever likely to be obtained),

- an area where there is a large amount of data, in computer readable form, that requires examination, classification, and integration, or
- an area where small improvements in performance are routinely measured (or easily measurable) and highly prized.

For example, problems in automated control are especially well suited for genetic programming because of the inability of conventional mathematical analysis to provide analytic solutions to many problems of practical interest, the willingness of control engineers to accept approximate solutions, and the high value placed on small incremental improvements in performance.

Problems in fields where large amounts of data are accumulating in machine readable form (e.g., biological sequence data, astronomical observations, geological and petroleum data, financial time series data, satellite observation data, weather data, news stories, marketing databases) also constitute especially interesting areas for potential practical applications of genetic programming.

3. The Threshold of Practicality

Evidence is accumulating that genetic programming is now reaching the threshold of delivering results that are competitive with human performance on non-trivial problems. There have been several recent examples of problems – from fields as diverse as cellular automata, space satellite control, molecular biology, and design of electrical circuits – in which genetic programming has evolved a computer program whose results were, under some reasonable interpretation, competitive with human performance on the specific problem. For example, genetic programming with automatically defined functions has evolved a rule for the majority classification task for one-dimensional two-state cellular automata with an accuracy that exceeds that of the original human-written Gacs-Kurdyumov-Levin (GKL) rule, all other known subsequent human-written rules, and all other known rules produced by automated approaches for this problem (Andre, Bennett, and Koza 1996). Another example involves the near-minimum-time control of a spacecraft's attitude maneuvers using genetic programming (Howley 1996). A third example involves the discovery by genetic programming of a computer program to classify a given protein segment as being a transmembrane domain without using biochemical knowledge concerning hydrophobicity (Koza 1994a; Koza and Andre 1996a, 1996b). A fourth example illustrated how automated methods may prove to be useful in discovering biologically meaningful information hidden in the rapidly growing databases of DNA sequences and protein sequences. Genetic programming successfully evolved motifs for detecting the D-E-A-D box family of proteins and for detecting the manganese superoxide dismutase family that detected the two families either as well as, or slightly better than, the comparable human-written motifs found in the database created by an international committee of experts on molecular biology (Koza and Andre 1996c). A fifth example involves the design of difficult-to-design electrical circuits using genetic programming (Koza, Bennett, Andre, and Keane 1996). A sixth example is recent work on facility layouts (Garces-Perez, Schoenefeld, and Wainwright 1996).

4. Handling Complex Data Structures

Ordinary computer programs use numerous well-known techniques for handling vectors of data, arrays, and more complex data structures. One important area for work on technique extensions for genetic programming involves developing workable and efficient ways to handle vectors, arrays, trees, graphs, and more complex data structures. Such new techniques would have immediate application to a number of problems in such fields as computer vision, biological sequence analysis, economic time series analysis, and pattern recognition where a solution to the

problem involves analyzing the character of an entire data structure. Recent work in this area includes that of Langdon (1996) in handling more complex data structures, Teller (1996) in understanding images represented by large arrays of pixels, and Handley (1996) in applying statistical computing zones to biological sequence data.

5. Evolution of Mental Models

Complex adaptive systems usually possess a mechanism for modeling their environment. A mental model of the environment enables a system to contemplate the effects of future actions and to choose an action that best fulfills its goal. Brave (1996b) has developed a special form of memory that is capable of creating relations among objects and then using these relations to guide the decisions of a system.

6. Evolution of Assembly Code

The innovative work by Nordin (1994) in developing a version of genetic programming in which the programs are composed of sequence of low-level machine code offers numerous possibilities for extending the techniques of genetic programming (especially for programs with loops) as well as enormous savings in computer time. These savings can then be used to increase the scale of problems being considered.

7. Automatically Defined Functions and Macros

Computer programs gain leverage in solving complex problems by means of reusable and parametrizable subprograms. Automated machine learning can become scalable (and truly useful) only if there are techniques for creating large and complex problem-solving programs from smaller building blocks. Rosca (1995) has analyzed the workings of hierarchical arrangements of subprograms in genetic programming. Spector (1996) has developed the notion of automatically defined macros (ADMs) for use in evolving control structures. Considerable future work can be anticipated in this area.

8. Cellular Encoding

Gruau (1994) described an innovative technique, called *cellular encoding* or *developmental genetic programming* in which genetic programming is used to concurrently evolve the architecture of a neural network, along with the weights, thresholds, and biases of the individual neurons in the neural network. In this technique, each individual program tree in the population is a specification for developing a complete neural network from a starting point consisting of a very simple embryonic neural network containing a single neuron. Genetic programming is applied to populations of these network-constructing program trees in order to evolve a neural network to solve various problems. Brave (1996a) has extended and applied this technique to the evolution of finite automata. This technique has also been applied to other complex structures, such as electrical circuits (Koza, Bennett, Andre, and Keane 1996).

9. Automatic Programming of Multi-Agent Systems

The cooperative behavior of multiple independent agents can potentially be harnessed to solve a wide variety of practical problems. However, programming of multi-agent systems is particularly vexatious. Bennett's recent work (1996) in evolving the number of independent agents while concurrently evolving the specific behaviors of each agent and the recent work by Luke and Spector (1996) in evolving teamwork are opening this area to the application of genetic programming.

10. Autoparallelization of Algorithms

The problem of mapping a given sequential algorithm onto a parallel machine is usually more difficult than writing a parallel algorithm from scratch. The recent work of Walsh and Ryan (1996) is advancing the autoparallelization of algorithms using genetic programming. Considerable future work can be anticipated in this important area.

11. Co-Evolution

In nature, individuals do not evolve in a vacuum. Instead, there is co-evolution that involves interactions between agents and other agents as well as between agents and their physical environment. The important area of co-evolution, as illustrated by the work of Pollack and Blair (1996), can be expected to attract considerable future work.

12. Complex Adaptive Systems

Genetic programming has proven useful in evolving complex systems, such as Lindenmayer systems (Jacob 1996) and cellular automata (Andre, Bennett, and Koza 1996) and can be expected to continue to be useful in this area.

13. Evolution of Structure

One of the most vexatious aspects of automated machine learning from the earliest times has been the requirement that the human user predetermine the size and shape of the ultimate solution to his problem (Samuel 1959). There can be expected to be continuing research on ways by which the size and shape of the solution can be made part of the *answer* provided by the automated machine learning technique, rather than part of the *question* supplied by the human user. For example, architecture-altering operations (Koza 1995) enable genetic programming to introduce (or delete) function-defining branches, to adjust the number of arguments of each function-defining branch, and to alter the hierarchical references among function-defining branches. Brave (1995) showed that recursion could be implemented within genetic programming. Future work can be expected on operations that enable genetic programming to dynamically introduce iteration and recursion and nested occurrences of iteration and recursion.

14. Foundations of Genetic Programming

Genetic programming inherits many of the mathematical and theoretical underpinnings from John Holland's pioneering work (1975) in the field, including the near-optimality of Darwinian search. However, the genetic algorithm is a dynamical system of extremely high dimensionality. Many of the most basic questions about the operation of the algorithm and the domain of its applicability are only partially understood. The transition from the fixed-length character strings of the genetic algorithm to the variable-sized Turing-complete program trees (and even program graphs) of genetic programming further compounds the difficulty of the theoretical issues involved. There is increasing word on the grammatical structure of genetic programming (Whigham 1996).

15. Optimization

The fundamental importance of optimization problems guarantees that there will be considerable future work on applying genetic programming to optimization. Recent examples include work (Soule, Foster, and Dickinson 1996) from the University of Idaho, the site of much early work on genetic programming techniques and the work of Garces-Perez, Schoenefeld, and Wainwright (1996).

16. Novel Methods of Fitness Evaluation

In a novel experiment, Floreano and Mondada (1994) ran the genetic algorithm on a fast workstation to evolve a control strategy for an obstacle-avoiding robot. The fitness of an individual strategy in the population within a particular generation of the run was determined by executing a physical robot tethered to the workstation for 30 seconds in real time. The robot behavior is thus highly realistic and avoids the pitfalls of computer simulated behavior. This technique can be expected to find future application in genetic programming.

17. Techniques that Exploit Parallel Hardware

Evolutionary algorithms offer the ability of solve problems in a domain-independent way that requires little domain-specific knowledge. However, the price of this domain-independence and knowledge-independence is paid in execution time. Application of genetic programming to realistic problems inevitably requires considerable horse power. The long-term trend toward ever faster microprocessors is likely to continue to provide ever increasing amounts of computational power. However, for those using algorithms that can beneficially exploit parallelization (such as genetic programming), the trend toward decreasing prices of hardware will be even more important in terms of providing the large amounts of computational power necessary to solve realistic problems. In most genetic programming applications, the vast majority of computer resources are used on the fitness evaluations. The calculation of fitness for the individuals in the population is usually entirely decoupled. Thus, parallel computing techniques can be beneficially applied to genetic programming and genetic algorithms with almost 100% efficiency (Andre and Koza 1996). In fact, the use of semi-isolated subpopulations often accelerates the finding of a solution to a problem using genetic programming and produces super-linear speed-up. Parallelization of genetic programming will be of central importance to the growth of the field.

18. Evolvable Hardware

One of the exciting new areas of evolutionary programming involves the use of evolvable hardware (Sanchez and Tomassini 1996). Evolvable hardware includes devices such as field programmable gate arrays (FPGA) and field programmable analog arrays (FPAA). These devices are reconfigurable with very short configuration times and download times. Thompson (1996) has pioneered the use of field-programmable gates arrays to evolve a frequency discriminator circuit and a robot controller using the recently developed Xilinix 6216 chip. I anticipate an explosive growth in the use of genetic programming to evolve hardware and the use of reconfigurable hardware to accelerate genetic programming runs.

Bibliography

Andre, David, Bennett III, Forrest H, and Koza, John R. 1996. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Andre, David and Koza, John R. 1996. Parallel genetic programming: A scalable implementation using the transputer network architecture. In Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge, MA: The MIT Press. Chapter 18.

Bennett, Forrest H III. 1996. Automatic creation of an efficient multi-agent architecture using genetic programming with architecture-altering operations. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming*

1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University. Cambridge, MA: The MIT Press.

Brave, Scott. 1995. Using genetic programming to evolve recursive programs for tree search. *Proceedings of the Fourth Golden West Conference on intelligent System*s. Raleigh, NC: International Society for Computers and Their Applications. Pages 60 – 65.

Brave, Scott. 1996a. Evolving deterministic finite automata using cellular encoding. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Brave, Scott. 1996b. The evolution of memory and mental models using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

de Garis, Hugo. CAM-BRAIN: The evolutionary engineering of a billion neuron artificial brain by 2001 which grows / evolves at electronic speeds inside a cellular automata machine (CAM). In Sanchez, Eduardo and Tomassini, Marco (editors). *Towards Evolvable Hardware*. Lecture Notes in Computer Science, Volume 1062. Berlin: Springer-Verlag. Pages 76 – 98.

Floreano, Dario and Mondada, Francesco. 1994. Automatic creation of an autonomous agent: Evolution of a neural-network drive robot. In Cliff, Dave, Husbands, Philip, Meyer, Jean-Arcady, and Wilson, Stewart W. (editors). 1994. *From Animals to Animats 3 Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. Pages 421–430.

Garces-Perez, Jaime, Schoenefeld, Dale A., and Wainwright, Roger L. 1996. Solving facility layout problems using genetic programming**.** In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Gruau, Frederic. 1994. Genetic micro programming of neural networks. In Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming*. Cambridge, MA: The MIT Press. Pages 495–518.

Handley, Simon. 1996. A new class of function sets for solving sequence problems. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: The MIT Press.

Holland, John H. 1975. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press. The 1992 second edition was published by The MIT Press.

Howley, Brian. 1996. Genetic programming of near-minimum-time spacecraft attitude maneuvers. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Koza, John R. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

Koza, John R. 1994. *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: The MIT Press.

Koza, John R. 1995. Gene duplication to enable genetic programming to concurrently evolve both the architecture and work-performing steps of a computer program. *Proceedings of*

14th International Joint Conference on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann.

Koza, John R. and Andre, David. 1996a. Classifying protein segments as transmembrane domains using architecture-altering operations in genetic programming. In Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming II*. Cambridge, MA: MIT Press.

Koza, John R. and Andre, David. 1996b. Evolution of iteration in genetic programming. In *Evolutionary Programming V*: *Proceedings of the Fifth Annual Conference on Evolutionary Programming*. Cambridge, MA: MIT Press.

Koza, John R. and Andre, David. 1996c. Automatic discovery of protein motifs using genetic programming. In Yao, Xin (editor). 1996. *Evolutionary Computation: Theory and Applications*. Singapore: World Scientific.

Koza, John R., Bennett III, Forrest H, Andre, David, and Keane, Martin A. 1996. Automated WYWIWYG design of both the topology and component values of analog electrical circuits using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Langdon, W. B. 1996. Using data structures within genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Nordin, Peter. 1994. A compiling genetic programming system that directly manipulates the machine code. In Kinnear, Kenneth E. Jr. (editor). 1994. *Advances in Genetic Programming*. Cambridge, MA: The MIT Press.

Pollack, Jordan B. and Blair, Alan D. 1996. Coevolution of a backgammon player. In *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems*. Cambridge, MA: The MIT Press.

Rosca, Justinian P. 1995. Genetic programming exploratory power and the discovery of functions. In McDonnell, John R., Reynolds, Robert G., and Fogel, David B. (editors). 1995. *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming*. Cambridge, MA: The MIT Press.

Samuel, Arthur L. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development.* 3(3): 210–229.

Sanchez, Eduardo and Tomassini, Marco (editors). *Towards Evolvable Hardware*. Lecture Notes in Computer Science, Volume 1062. Berlin: Springer-Verlag.

Soule, Terence, Foster, James A., and Dickinson, John. 1996. Code growth in genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Spector, Lee. 1996. Simultaneous evolution of programs and their control structures. In Angeline, Peter J. and Kinnear, Kenneth E. Jr. (editors). 1996. *Advances in Genetic Programming 2*. Cambridge, MA: The MIT Press.

Teller, Astro and Veloso Manuela. 1996. PADO: A new learning architecture for object recognition. In Ikeuchi, Katsushi and Veloso Manuela (editors). *Symbolic Visual Learning*. Oxford University Press.

Thompson, Adrian. Silicon evolution. 1996. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings*

of the First Annual Conference, July 28-31, 1996, Stanford University. Cambridge, MA: MIT Press.

Walsh, Paul and Ryan, Conor. 1996. Paragen: A novel technique for the autoparallelisation of sequential programs using genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.

Whigham, Peter A. Search bias, language bias, and genetic programming. In Koza, John R., Goldberg, David E., Fogel, David B., and Riolo, Rick L. (editors). 1996. *Genetic Programming 1996: Proceedings of the First Annual Conference, July 28-31, 1996, Stanford University*. Cambridge, MA: MIT Press.