Towards Byte Code Genetic Programming

Brad Harvey
Compaq
14231 Tandem Bl.
Austin, Texas 78728-6699
brad.harvey@compaq.com

James Foster
Computer Science Dept.
University of Idaho
Moscow, Idaho 83844-1010
foster@cs.uidaho.edu

Deborah Frincke
Computer Science Dept.
University of Idaho
Moscow, Idaho 83844-1010
frincke@cs.uidaho.edu

Abstract

We investigate using the GP paradigm to evolve linear genotypes (individuals) that consist of Java byte code. Our prototype GP system (bcGP) is implemented in Java. The evolutionary process is done completely in memory and the fitness of individuals is determined by directly executing them in the Java Virtual Machine (JVM). Our scheme is an effective means for evolving native machine code for the JVM.

1 INTRODUCTION

Genetic Programming (GP) has proven to be a very powerful paradigm for solving diverse problems from a variety of different domains. The Java language and runtime environment is rapidly gaining acceptance as evidenced by numerous vendors including Java as strategic parts of their systems. Because of the significance of these two technologies, we explore using GP to directly evolve Java byte code in the context of a standard Java environment. Our work has been inspired by Nordin's [1994] use of GP to evolve RISC machine code. We, along with other researchers, are investigating the possibilities and issues of byte code GP [Harvey, et al., 1998][Klahold, et al., 1998][Lukschandl, et al., 1998]

2 BYTE CODE GP (bcGP)

The goals for our research prototype are: (1) use of a standard JDK, (2) implementation of bcGP in Java only, (3) direct execution of evolved individuals in the standard JVM, and (4) the evolutionary process occurring completely in memory.

A significant aspect of our work has been designing a genotype only representation scheme, which facilitates in memory evolution in the context of bcGP and the JVM. In our scheme, a generation is represented by an in-memory version of a Java class file with individuals represented as methods in the class file. A generation class file is organized in *index standard form*, which allows for a unique naming and indexing scheme for both generations and individuals. This scheme enables bcGP to directly manipulate the method byte code constituting an individual in a class file during evolution, and to dispatch it for execution during fitness evaluation. In the prototype, single-point crossover and mutation are used as well as k-tournament selection.

3 CONCLUSIONS

We validate our approach by solving a functional regression problem with a fourth degree polynomial, $f(x) = x^4+x^3+x^2+x$, and a classification problem diagnosing thyroid disease. For the classification problem, bcGP evolves individuals consisting of conditional byte codes. Each individual represents a rule used to classify instances of the database as either having or not having thyroid disease based upon an instance‘s features.

References

B. Harvey, J. Foster, D. Frincke (1998). Byte Code Genetic Programming. In *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin – Madison

S. Klanhlold, S. Frank, R. Keller, W. Banzhaf (1998). Exploring the Possibilities and Restrictions of Genetic Programming in Java Bytecode. In *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin – Madison

E. Lukschandl, M. Holmlund, E. Moden, M. Nordahl, P. Nordin (1998). Induction of Java Bytecode with Genetic Programming. In *Late Breaking Papers at the Genetic Programming 1998 Conference*, University of Wisconsin – Madison.

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