Evolving Readable Perl

Mark S. Withall
Department of Computer Science
Loughborough University
Leics. LE11 3TU, UK
m.s.withall2@lboro.ac.uk

Chris J. Hinde
Department of Computer Science
Loughborough University
Leics. LE11 3TU, UK
c.j.hinde@lboro.ac.uk

Roger G. Stone
Department of Computer Science
Loughborough University
Leics. LE11 3TU, UK
r.g.stone@lboro.ac.uk

1 INTRODUCTION

A program is informally deemed readable, for the purpose of this experiment, if it is easy for a person to follow the steps that the program takes to solve the problem. In this experiment, readability is achieved by constraining the available syntax for generating solutions.

The Genetic Programming (GP) system created uses the target language Perl because it is an interpreted, untyped, robust procedural language which has good error handling and recovery.

2 GENETIC PROGRAM

The *genotype* and *phenotype* have been separated to make genetic manipulation simpler. Each program is represented as a fixed-length integer array and then mapped onto Backus-Naur Form (BNF). The program statements used are shown in Figure 1a. The BNF is designed to minimise the size of the genome that describes a program. The mapping, between the genotype and phenotype, is similar to *Grammatical Evolution*[2].

The GP was tested using the symbolic regression problem $X^4 + X^3 + X^2 + X$[1]. A population size of 500 and mutation rate of 1 gene in 5000 were used for the test problem. The population was initialised randomly and each test run was of 100 generations. The fitness values for the programs were given as the absolute difference between the target value and the actual value.

3 RESULTS AND CONCLUSIONS

All results were of the correct order (X^4) and 5 out of the 8 test runs produced entirely correct solutions. An example of an optimal program is given in Figure 1b.

The results of the experiment were encouraging. As a comparison the solution evolved by Koza[1] is given in Figure 1c, which is only really understandable by LISP users.

STMT	FORMAT
Assign	$X = Y$
Add	$X = Y + Z$
Sub	$X = Y - Z$
Mul	$X = Y \times Z$
If	if(X cmp Y){
For	for $X(0..Y)${
End	}

(a)

```
# Header
$x = $ARGV[0];
$res = 0;

# Evolved Code
$res = $x * $x;
$x = $x + $res;
$res = $x * $res;
$res = $res + $x;

# Footer
print "$res";
```

(b)

```
(+X(*(+X(*(*(+X(-(COS(-XX))(-XX)))X)X))X))
```

(c)

Figure 1: (a) List of program statements used. (b) Example of code produced to solve the problem. (c) LISP result from Koza[1].

Acknowledgements

Thanks to everybody and all their friends.

References

[1] Koza J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press.

[2] Ryan C. O'Neill M. & Collins J.J. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. Lecture Notes in Computer Science 1391. First European Workshop on Genetic Programming 1998.