Genetic Programming With Gene Dominance

Kanta Vekaria and Chris Clack
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom
Email: K.Vekaria@cs.ucl.ac.uk, C.Clack@cs.ucl.ac.uk

ABSTRACT
This paper proposes the use of haploid gene dominance in genetic programming.

1. Introduction

Dawkins's model of evolution is based on the gene. He presents his theory of the gene as the fundamental unit of natural selection [Dawk89]. Genetic material in complex organisms is represented using a diploid chromosome. In the diploid form a genotype carries one or more *pairs* of chromosomes, each containing information for the same functions. The genes contained in one set can be regarded as a direct alternative to the genes in the other set. When building the body the genes from one set compete with those in the other set. Genes which are expressed in the phenotype of an organism are dominant and those that don't are recessive. Some genes that have been known to be dominant have become recessive in successive generations and vice versa. These dominance characteristics have evolved over generations.

Genetic programming [Koza92] traditionally uses a haploid chromosome. The haploid form contains all the information relevant to the problem. The genes do not have associated dominance values.

2. GP with Dominance and Haploidy

Assuming a correlation between program fitness and subtree fitness, we propose an alternative method of crossover which uses a dominance operator.

The parse tree contains the normal defined function and terminal sets. Each use of a function or terminal will have an associated dominance value. This dominance value will reflect how good each one is with respect to the fitness of the entire program.

During crossover two parent trees are selected and the position for crossover is selected at random. Once the subtrees are chosen, the nodes from each subtree are compared, from top to bottom and left to right. The node with greater dominance is used to create a new subtree. This is a recursive process. In the case where one tree is greater than the other the remaining component in the larger tree is simply copied to the new subtree. This new subtree is then attached to the trees of the parent where crossover occurred. Figure 1 shows an example of crossover.

```
Parent 1      Parent 2         Child 1        Child 2
  a              z               a               z
 / \            /\              / \             / \
 b  c          y    x           b  y            y   x
     /\       /\   /\              /\          /\   /\
    d  e      w v  u t            d  e        d  e  u t
   /\         /\                  /\          /\
   f g        s r                 s  r        s r
```

(c) and (y) are chosen for crossover
(y) has a greater dominance value than (c)
(d) has a greater dominance than (w)
(e) has a greater dominance than (v)
(s), (r) are dominant over (f), (g)

Figure 1 Crossover using dominance

The fitness is then evaluated and the dominance value of the new subtree that was attached is increased proportionately to the increase in fitness.

On initialisation of the population, random dominance values are assigned.

Preliminary experiments will use simple crossover not the merge crossover shown in figure 1. Here dominance is compared only at the top nodes of the chosen subtree. The node which is recessive is replaced by the dominant one.

3. Prospects

We will investigate whether, as in nature, this method enhances the GP evolutionary process. We will further investigate:

- the best way to increase dominance values relative to program fitness.
- implications for evolutionary convergence.
- the application of dominance to ADFs and other modular techniques.

Bibliography

[Dawk89] Dawkins, R. (1989). The Selfish Gene - New Ed. Oxford University Press, Great Britain.

[Koza92] Koza, J.R. (1992). Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA:MIT Press.