Adaptive Strategies for a Semantically Driven Tree Optimizer to Control Code Growth

Bart Wyns, Luc Boullart
Dept. of Electrical, Systems and Automation, Ghent University
Technologiepark 913
Zwijnaarde, Belgium
Bart.Wyns@UGent.be, Luc.Boullart@UGent.be

ABSTRACT

In genetic programming many methods to fight growth exist. But most of these methods require one or multiple parameters to be set. Unfortunately performance strongly depends on a correct setting of each of those parameters. Recently a semantically driven tree optimizer has been developed. In this paper two adaptive strategies to choose a reasonable parameter setting for this growth limiter are presented.

Categories and Subject Descriptors: I.2.2 [Artificial Intelligence]: Automatic Programming Program synthesis; I.2.6 [Artificial Intelligence]: Learning Concept Learning and Induction; I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search

General Terms: Algorithms, Design, Experimentation

Keywords: Adaptive control, fuzzy logic, code growth, artificial ant, Boolean problems, genetic programming

1. INTRODUCTION

In previous research [1] a semantically driven code growth limiter was introduced (LOSE). Results on a variety of benchmark applications showed a substantial reduction of program size and depth and a significant increase in program fitness. But this growth limiter, as many others, has one disadvantage: it requires a setting of a breeding rate. A "good" setting is often very hard to choose and depends on the preferred size reduction and fitness.

2. AN ADAPTIVE PROBABILITY SETTING

In this section two new methods to adaptively control the breeding rate setting of LOSE are introduced. The first approach is based on fuzzy logic principles (FuLOSE) and uses size and fitness increases (Fig. 1). The second approach to control the breeding rate of LOSE focusses on the number of individuals with a suited subtree (ALOSE).

3. EXPERIMENTING

The artificial ant and the 11-bit multiplexer were used to validate the effect of adaptive LOSE on fitness and size.

In the ant FuLOSE produces similar results compared to static LOSE. Although program size increased, it is clear that FuLOSE is reducing size in the second part of the evolutionary run. FuLOSE consistently shows better performance compared to other growth limiters. In the multiplexer problem FuLOSE evolved small solutions but at the expense of fitness, dropping even below standard GP. The same experiment but using the ALOSE was also performed. We note that the number of suited subtrees will be very large in the beginning and then gradually decreases. So LOSE will rapidly reduce program size in the beginning but will not be used very often near the end of the run. In both the artificial ant and the multiplexer problem ALOSE really triumphs and outperforms any other kind of growth limiting and standard GP. Average size increases as explained above. But if we take a look at the program size of the best-of-generation individuals then only a minor increase can be noted.

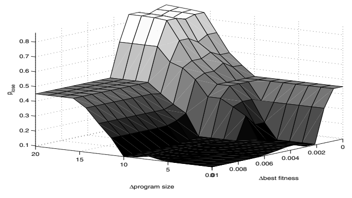

Figure 1: The fuzzy logic controller.

4. CONCLUSIONS

In summary, two new adaptive control strategies were proposed to set an optimal breeding rate for LOSE. The first method was based on fuzzy logic principles but had a few shortcomings. The model turned out to be quite complex (many other parameters were required such as the number of membership functions and their shape, etc.). To choose appropriate values for the FLC settings, the application programmer still needs to have some prior information on the problem at hand. As shown the resulting model was still dependent on the benchmark problem. A second much simpler algorithm was based on feedback given by the LOSE operator itself. We showed that this approach has some advantages over the FLC. Results of ALOSE on both benchmarks outperformed any other growth limiter as well as standard GP and the FLC (multiplexer).

5. REFERENCES

[1] B. Wyns, S. Sette, and L. Boullart. Self-improvement to control code growth in genetic programming. *Lecture Notes in Computer Science*, 2936, 256–266, 2004.