Dynamic Maximum Tree Depth

A Simple Technique for Avoiding Bloat in Tree-Based GP

Sara Silva[1] and Jonas Almeida[1,2]

[1] Biomathematics Group, Instituto de Tecnologia Química e Biológica
Universidade Nova de Lisboa, PO Box 127, 2780-156 Oeiras, Portugal
sara@itqb.unl.pt
http://www.itqb.unl.pt:1111

[2] Dept Biometry & Epidemiology, Medical Univ South Carolina, 135 Cannon Street
Suite 303, PO Box 250835, Charleston SC 29425, USA
almeidaj@musc.edu
http://bioinformatics.musc.edu

Abstract. We present a technique, designated as dynamic maximum tree depth, for avoiding excessive growth of tree-based GP individuals during the evolutionary process. This technique introduces a dynamic tree depth limit, very similar to the Koza-style strict limit except in two aspects: it is initially set with a low value; it is increased when needed to accommodate an individual that is deeper than the limit but is better than any other individual found during the run. The results show that the dynamic maximum tree depth technique efficiently avoids the growth of trees beyond the necessary size to solve the problem, maintaining the ability to find individuals with good fitness values. When compared to lexicographic parsimony pressure, dynamic maximum tree depth proves to be significantly superior. When both techniques are coupled, the results are even better.

1 Introduction

Genetic Programming (GP) solves complex problems by evolving populations of computer programs, using Darwinian evolution and Mendelian genetics as inspiration. Its search space is potentially unlimited and programs may grow in size during the evolutionary process. Code growth is a healthy result of genetic operators in search of better solutions. Unfortunately, it also permits the appearance of pieces of redundant code, called introns, which increase the size of programs without improving their fitness. Besides consuming precious time in an already computationally intensive process, introns may start growing rapidly, a situation known as bloat. Several plausible explanations for this phenomenon have been proposed (reviews in [9,12]), and a combination of factors may be involved, but whatever the reasons may be, the fact remains that bloat is a serious problem that may stagnate the evolution, preventing the algorithm from finding better solutions.

Several techniques have been used in the attempt to control bloat (reviews in [7,11]), most of them based on parsimony pressure. This paper presents a

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1776–1787, 2003.

technique especially suited for tree representations in GP, based on tree depth limitations. The idea is to introduce a dynamic tree depth limit, very similar to the Koza-style strict limit [3] except in two aspects: it is initially set with a low value; it is increased when needed to accommodate an individual that is deeper than the limit but is better than any other individual found during the run. Unlike some of the most recent successful approaches, this technique does not require especially designed operators [6,14] and its performance is not reduced in the presence of populations where all individuals have different fitness values [7]. It is also simple to implement and can easily be coupled with other techniques to join forces in the battle against bloat.

2 Introns

In GP, the term *intron* usually refers to a piece of redundant code that can be removed from an individual without affecting its fitness [1]. The term *exon* refers to all non-intron code. However, on designing a procedure for detecting introns in GP trees, a more precise definition arises.

Our procedure for detecting introns works recursively in a tree from its root to its terminals, trying to maximize the number of introns detected. It can be described by the following pseudo-code, where *nintrons* designates the number of introns detected:

```
If tree is a terminal node:
    nintrons = 0
Otherwise:
    Evaluate tree and all its branches
    If none of the branches returns the same evaluation as tree:
        nintrons = sum of the number of introns in each branch
    Otherwise:
        Count nodes and introns in branches with same evaluation as tree
        Pick branch with lowest number of non intron nodes: ibranch
        nintrons = all nodes in tree minus non intron nodes in ibranch
```

This does not detect introns in code like `not(not(X))` or `-(-(X))`, although removing pieces of this code, leaving only the underlined parts, would not affect the fitness of the individual. We use the term *redundant complexity* to designate these and other cases where a tree could be replaced by a smaller tree (but not by one of its branches) without affecting the fitness of the individual. Redundant complexity is also a problem in GP [10].

We use the term *bloat* loosely to designate a rapid growth of the mean population size in terms of the number of nodes that make up the trees, whether it happens in the beginning or in the end of the run, and whether the growth is caused by introns or exons.

3 Dynamic Maximum Tree Depth

Dynamic maximum tree depth is a technique that introduces a dynamic limit to the maximum depth of the individuals allowed in the population. It is similar to the Koza-style depth limiting technique, but it does not replace it – both dynamic and strict limits are used in conjunction. There is also an initial depth limit imposed on the trees that form the initial population [3], which must not be mistaken for the initial value of the dynamic limit. The dynamic limit should be initially set with a low value at least as high as the initial tree depth. It can increase during the run, but it always lies somewhere between the initial tree depth and the strict Koza limit. Whenever a new individual is created by the genetic operators, these rules apply:

- if the new individual is deeper than the strict Koza depth limit, reject it and consider one of its parents for the new generation instead;
- if the new individual is no deeper than the dynamic depth limit, consider it acceptable to participate in the new generation;
- if the new individual is deeper than the dynamic limit (but no deeper than the strict Koza limit) measure its fitness and:
  - if the individual has better fitness than the current best individual of the run, increase the dynamic limit to match the depth of the individual and consider it acceptable to the new generation;
  - if the individual is not better than the current best of run, leave the dynamic level unchanged and reject the individual, considering one of its parents for the new generation instead.

Once increased, the dynamic level will not be lowered again. If and when the dynamic limit reaches the same value as the strict Koza limit, both limits become one and the same. By setting the dynamic limit to the same value as the strict Koza limit, one is in fact switching it off, and making the algorithm behave as if there were no dynamic limit.

The dynamic maximum depth technique is easy to implement and can be used with any set of parameters and/or in conjunction with other techniques for controlling bloat. Furthermore, it may be used for another purpose besides controlling bloat. In real world applications, one may not be interested or able to invest a large amount of time in achieving the best possible solution, particularly in approximation problems. Instead, one may only consider a solution to be acceptable if it is sufficiently simple to be comprehended, even if its accuracy is known to be worse than the accuracy of other more complex solutions. Choosing less stringent stop conditions to allow the algorithm to stop earlier is not enough to ensure that the resulting solution will be acceptable, as it cannot predict its complexity. By starting with a low dynamic limit for tree depth and repeatedly raising it as more complex solutions prove to be better than simpler ones, the dynamic maximum tree depth technique can in fact provide a series of solutions of increasing complexity and accuracy, from which the user may choose the most adequate one. Once again, it is important to choose a low value for the initial dynamic depth limit, to force the algorithm to look for simpler solutions before adopting more complex ones.

4 Experiments

To test the efficacy of the dynamic maximum tree depth technique we have decided to perform the same battery of tests using six different approaches: (1) Koza-style tree depth limiting alone [3]; (2) lexicographic parsimony pressure (coupled with the Koza-style limit, which yielded better results than lexicographic parsimony pressure alone, with no bucketing) [7]; (3) dynamic maximum tree depth (also coupled with the Koza-style limit, as described in the previous section) with initial dynamic limit 9; (4) dynamic maximum tree depth, with initial dynamic limit 9, coupled with lexicographic parsimony pressure; (5) dynamic maximum tree depth with initial dynamic limit 6 (the same as the initial tree depth); (6) dynamic maximum tree depth, with initial dynamic limit 6, coupled with lexicographic parsimony pressure.

Two different problems were chosen for the experiments: Symbolic Regression of the quartic polynomial ($x^4 + x^3 + x^2 + x$, with 21 equidistant points in the interval -1 to $+1$), and Even-3 Parity. These problems are very different in the number of possible fitness values of the evolved solutions – from a potentially infinite number in Symbolic Regression, to very few possible values in Even-3 Parity. This facilitates the comparison with lexicographic parsimony pressure because it covers the two domains where this technique, due to its characteristics, behaved most differently [7].

A total of 30 runs were performed with each technique for each problem. All the runs used populations of 500 individuals evolved during 50 generations, even when an optimal solution was found earlier. The parameters adopted for the experiments were essentially the same as in [3] and [7], to facilitate the comparison between the techniques, but a few differences must be noted. Although some effort was put into promoting the diversity of the initial population, the tree initialization procedure used (Ramped Half-and-Half [3]) did not guarantee that all individuals were different. For 500 individuals and initial trees of maximum depth 6, the mean diversity of the initial population was roughly 75% for Symbolic Regression and 80% for Even-3 Parity, where diversity is the percentage of individuals in the population that account for the total number of different individuals (based on variety in [5]). Standard tree crossover was used, but with total random choice of the crossover points, independently of being internal or terminal nodes.

As stated in the previous section, the dynamic maximum tree limit can be switched on and off without affecting the rest of the algorithm, and switching it off is setting it to the same value as the Koza-style limit, which was always set to depth 17. Likewise, lexicographic parsimony pressure can be switched on by using a modified tournament selection that prefers smaller trees when their fitness values are the same, or switched off by using the standard tournament selection. Table 1 summarizes the combinations of tournament and initial dynamic limit that lead to each technique.

All the experiments were performed with GPLAB [8], a GP Toolbox for MATLAB [13]. Statistical significance of the null hypothesis of no difference was determined with ANOVAs at $p = 0.01$.

Table 1. Tournament and initial dynamic limit settings for each technique used

Technique	Tournament	Initial dynamic limit
(1) Koza-style tree depth limiting (K)	standard	17
(2) Lexicographic parsimony pressure (L)	modified	17
(3) Dynamic maximum tree depth 9 (S9)	standard	9
(4) L+S9	modified	9
(5) Dynamic maximum tree depth 6 (S6)	standard	6
(6) L+S6	modified	6

5 Results

The results of the experiments are presented as boxplots and evolution curves concerning the mean size of the trees (Fig. 1), where size is the number of nodes, the mean percentage of introns in the trees (Fig. 2), the population diversity as defined in the last section (Fig. 3), and best (lowest) fitness achieved (Fig. 4). The boxplots are based on the mean values (except Fig. 4, based on the best value) of all generations of each run, and each technique is represented by a box and pair of whiskers. Each box has lines at the lower quartile, median, and upper quartile values, and the whiskers mark the furthest value within 1.5 of the quartile ranges. Outliers are represented by + and × marks the mean. The evolution curves represent each generation separately (averaged over all runs), one line per technique. Throughout the text, we use the acronyms introduced in Table 1 to designate the different techniques.

5.1 Mean Size of Trees

Figure 1 shows the results concerning the mean size of trees. In the Symbolic Regression problem, techniques S9 and S6 performed equally well and were able to significantly lower mean tree sizes of run when compared to K and L. None of the compound techniques (L+S9, L+S6) proved to be significantly different from its counterpart (S9 and S6, respectively). The evolution curves show a clear difference in growth rate between the four techniques that use dynamic maximum tree depth (S9, L+S9, S6, L+S6) and the two that do not (K, L).

In the Even-3 Parity problem, techniques S9 and S6 were significantly different from each other. Both outperformed K, but only S6 performed better than L (no significant difference between S9 and L). However, both compound techniques (L+S9, L+S6) were able to outperform L, as well as their respective counterparts S6 and S9. The evolution curves show a tendency for stabilization of growth in all three techniques that use lexicographic parsimony pressure (L, L+S9, L+S6).

5.2 Mean Percentage of Introns

Figure 2 shows the results concerning the mean percentage of introns. Starting with Symbolic Regression, the striking fact about this problem is that it prac-

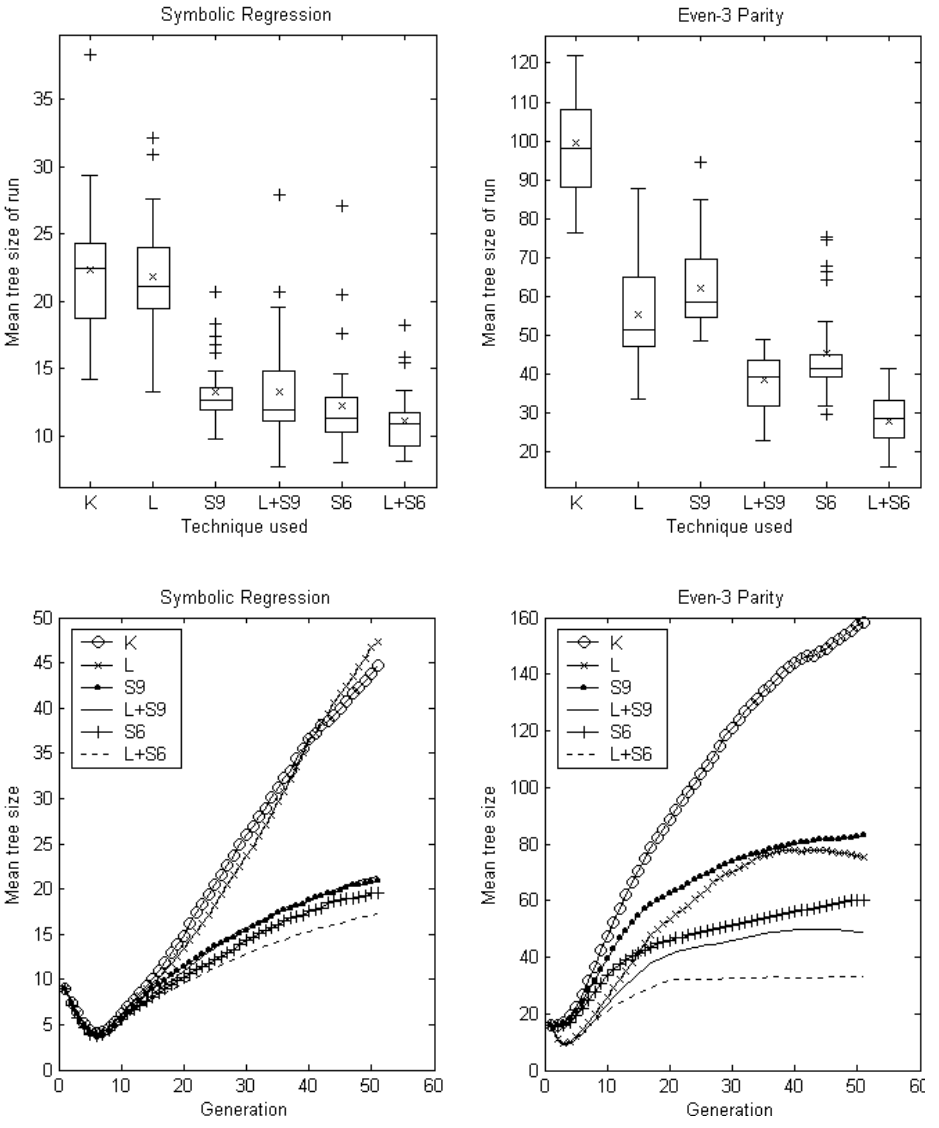

Fig. 1. Boxplots and evolution curves of mean tree size. See Table 1 for the names of the techniques

tically does not produce introns, redundant complexity being the only apparent cause of bloat. The differences between techniques in mean intron percentage of run were not significant. The evolution curves show that the percentage of introns tends to slowly increase after an initial reduction.

In the Even-3 Parity problem the percentage of introns is generally high. Techniques S9 and S6 outperformed K, and S6 also outperformed L, with no

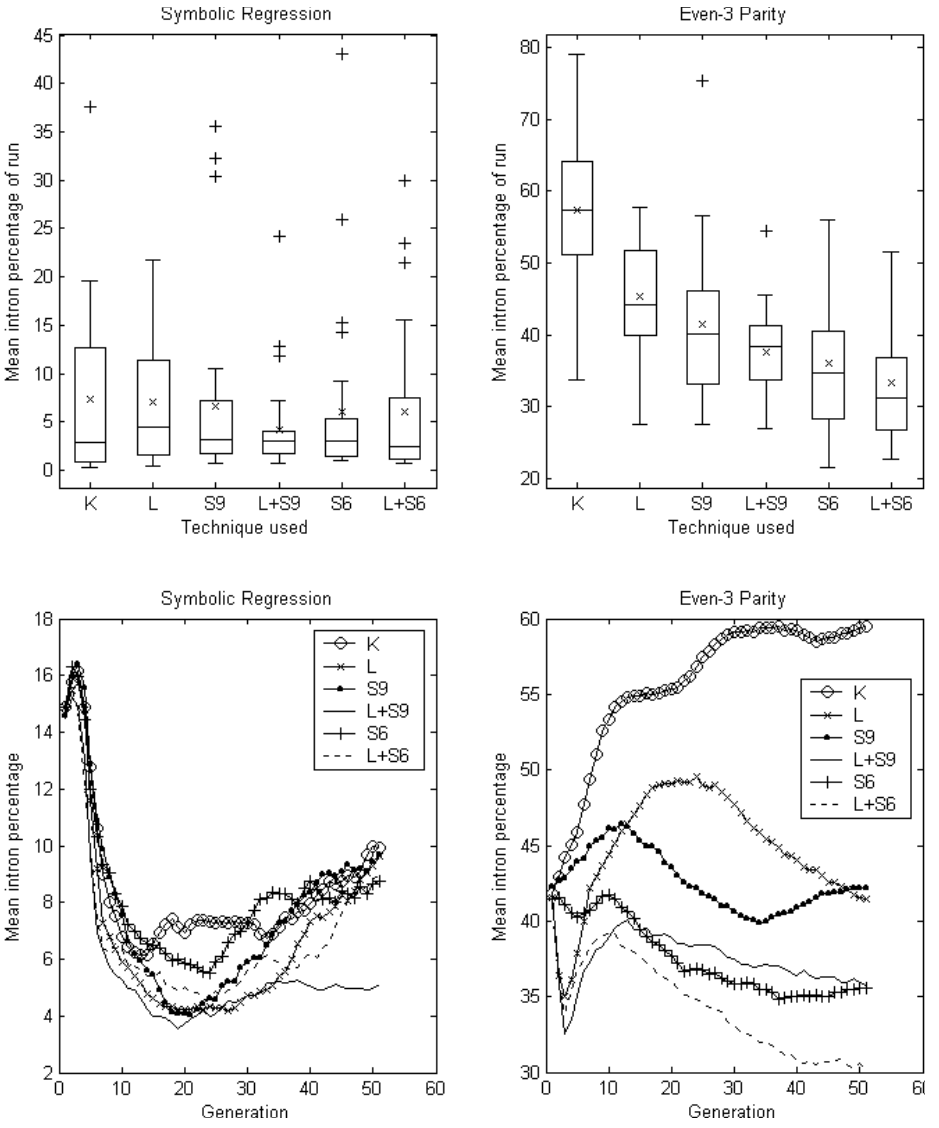

Fig. 2. Boxplots and evolution curves of mean intron percentage. See Table 1 for the names of the techniques

significant difference between S9 and L (similarly to what had been observed in mean tree size). The compound techniques (L+S9, L+S6) showed no significant differences from their counterparts (respectively S9 and S6), but both were able to significantly outperform L. The evolution curves show that, by the end of the run, the mean intron percentage has not increased from its initial value, in all techniques except K.

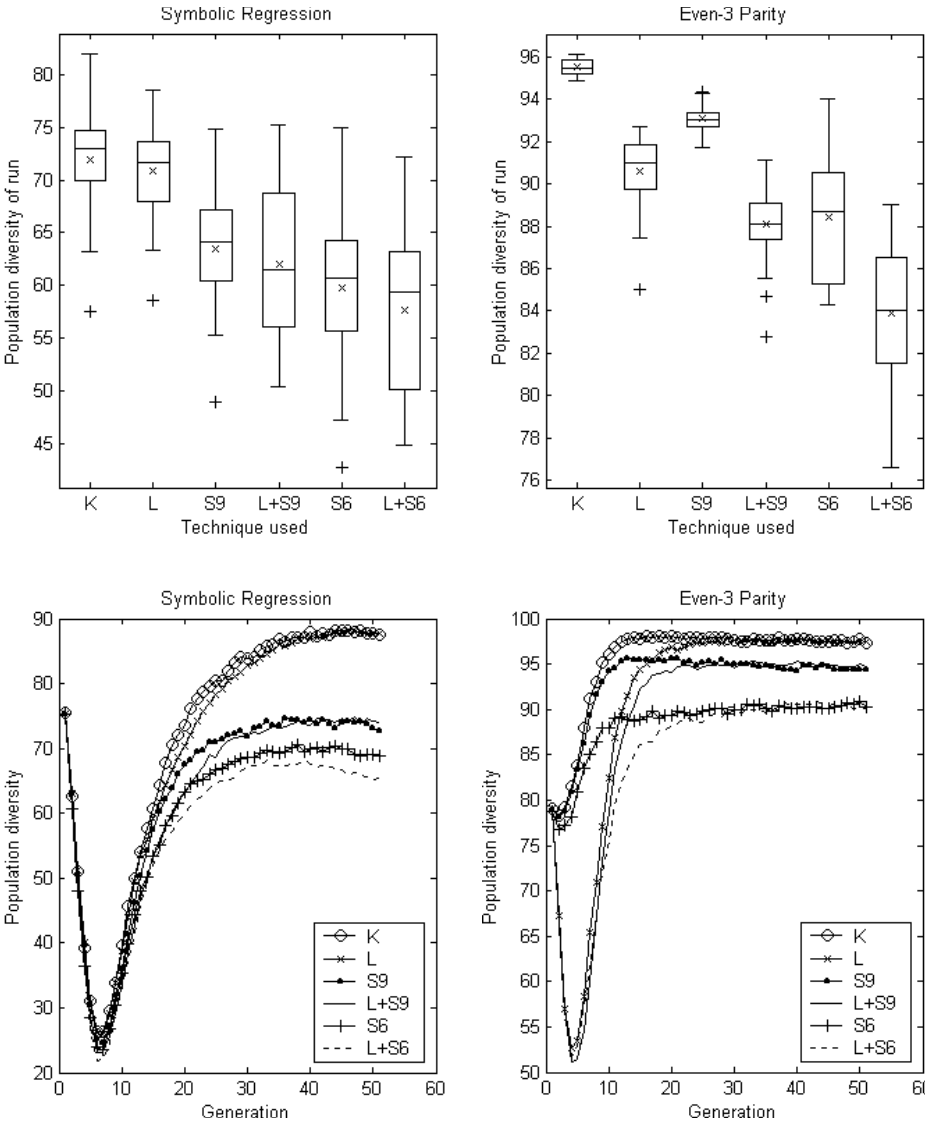

Fig. 3. Boxplots and evolution curves of population diversity. See Table 1 for the names of the techniques

5.3 Population Diversity

Figure 3 shows the results concerning the population diversity. In Symbolic Regression, techniques S9 and S6 caused a significant decrease in population diversity when compared to K and L. None of the compound techniques (L+S9, L+S6) was significantly different from its counterpart (S9 and S6, respectively). The evolution curves show a conspicuous decrease in population diversity in the

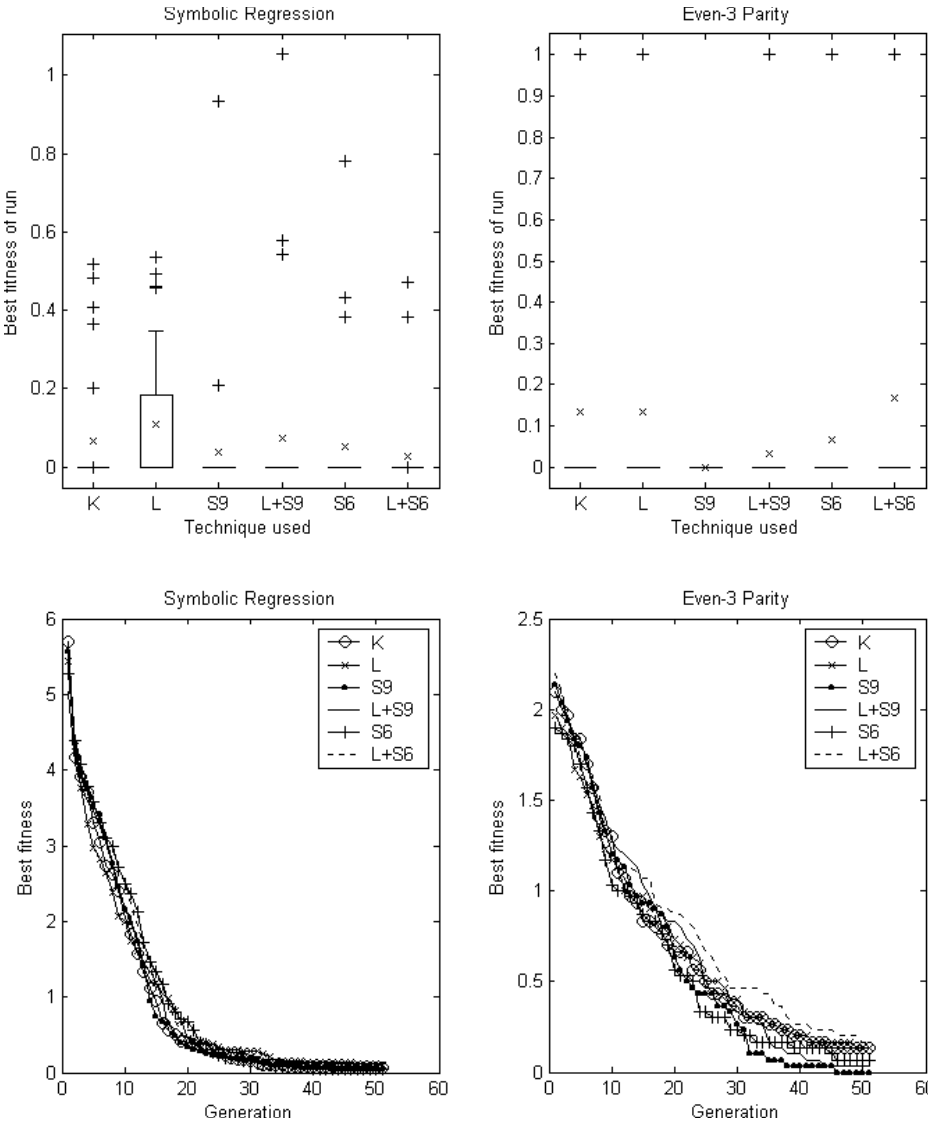

Fig. 4. Boxplots and evolution curves of best fitness. See Table 1 for the names of the techniques

beginning of the run, promptly recovered afterwards. The two techniques that do not use dynamic maximum tree depth (K, L) are able to sustain much higher population diversity in later generations.

In the Even-3 Parity problem, once again S9 and S6 caused a significant decrease in population diversity when compared to K, but showed significantly different performances when compared to each other. While S9 outperformed L,

maintaining significantly higher population diversity, S6 was outperformed by L. Both compound techniques (L+S9, L+S6) significantly lowered population diversity when compared to L and to their counterparts S9 and S6. The evolution curves for this problem also show an abrupt loss of diversity in the beginning of the run, but only in the techniques that use lexicographic parsimony pressure (L, L+S9, L+S6). In this problem, techniques using dynamic maximum tree depth (S9, L+S9, S6, L+S6) also result in lower population diversity in later generations, with both S6 and L+S6 yielding the worst results.

An initial decrease in population diversity is expected in problems where the number of possible fitness values is unlimited, as in Symbolic Regression. Since most of the random initial trees (particularly the larger ones) have very poor fitness, selection quickly eliminates them from the population in favor of a more homogeneous group of better (and usually smaller) individuals. It can be observed that the abrupt loss of diversity shown in Fig. 3 occurs at the same time in the run as the decrease in mean tree size shown in Fig. 1 (evolution curve on the left). In the Even-3 Parity problem, the number of possible fitness values is very limited, so the initial trees never have very poor fitness. However, several different trees have the same fitness value, and lexicographic parsimony pressure quickly eliminates the larger trees in favor of the smaller ones, thus producing the same effect (which also explains the initial decrease of the intron percentage that can be observed in Fig. 2, evolution curve on the right). This behavior has not been reported before.

A higher number of clonings resulting from unsuccessful crossovers, or simply the reduction of the search space, caused by the introduction of the dynamic limit, may account for the persistent lower diversity observed in the later stages of the run, with all techniques using dynamic maximum tree depth.

5.4 Best Fitness

Figure 4 shows the results concerning the best fitness. Although there has been some concern regarding whether depth restrictions may affect GP performance negatively when using crossover [2,4], in both Symbolic Regression and Even-3 Parity none of the techniques was significantly different from the others. The evolution curves indicate that convergence to good solutions was not a problem.

6 Conclusions and Future Work

The dynamic maximum tree depth technique was able to effectively control code growth in both Symbolic Regression and Even-3 Parity problems. In spite of a noticeable decrease in population diversity, the ability to find individuals with good fitness values was not compromised in these two problems. Dynamic maximum tree depth was clearly superior to lexicographic parsimony pressure in the Symbolic Regression problem. In the Even-3 Parity problem, where the differences between both techniques were not always significant, the combination of both outperformed the use of either one separately.

There seems to be no disadvantage in using the lower, and most limitative, value for the initial depth limit. The size of the trees was kept low without losing convergence into good solutions. However, the persistent lower population diversity observed when using the dynamic level, particularly with the more limitative setting, may become an impediment to good convergence in smaller populations, which may require a higher value for the initial depth limit in order to maintain the convergence ability. Further work must be carried out in order to confirm or reject this hypothesis. The performance of the dynamic maximum tree depth technique must also be checked in harder problems.

Acknowledgements. This work was partially supported by grants QLK2-CT-2000-01020 (EURIS) from the European Commission and POCTI/1999/BSE/34794 (SAPIENS) from Fundação para a Ciência e a Tecnologia, Portugal. We would like to thank Ernesto Costa (Universidade de Coimbra), Pedro J.N. Silva (Universidade de Lisboa), and all the anonymous reviewers for carefully reading the manuscript and providing many valuable suggestions.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic programming – an introduction, San Francisco, CA. Morgan Kaufmann (1998)
2. Gathercole, C., Ross, P.: An adverse interaction between crossover and restricted tree depth in genetic programming. In Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., editors, Proceedings of GP'96, Cambridge, MA. MIT Press (1996) 291–296
3. Koza, J.R.: Genetic programming – on the programming of computers by means of natural selection, Cambridge, MA. MIT Press (1992)
4. Langdon, W.B., Poli, R.: An analysis of the MAX problem in genetic programming. In Koza, J.R., Deb, K., Dorigo, M., Fogel, D.B., Garzon, M., Iba, H., Riolo, R.L., editors, Proceedings of GP'97, San Francisco, CA. Morgan Kaufman (1997) 222–230
5. Langdon, W.B.: Genetic Programming + Data Structures = Automatic Programming!, Boston, MA. Kluwer (1998)
6. Langdon, W.B.: Size fair and homologous tree crossovers for tree genetic programming. Genetic Programming and Evolvable Machines, 1 (2000) 95–119
7. Luke, S., Panait, L.: Lexicographic parsimony pressure. In Langdon, W.B., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R. Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M.A., Schultz, A.C., Miller, J.F., Burke, E., Jonoska, N., editors, Proceedings of GECCO-2002, San Francisco, CA. Morgan Kaufmann (2002) 829–836
8. Silva, S.: GPLAB – a genetic programming toolbox for MATLAB. (2003) `http://www.itqb.unl.pt:1111/gplab/`
9. Soule, T.: Code growth in genetic programming. PhD thesis, University of Idaho (1998)
10. Soule, T.: Exons and code growth in genetic programming. In Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B., editors, Proceedings of EuroGP-2002, Berlin. Springer (2002) 142–151

11. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation, 6(4) (1999) 293–309
12. Soule, T., Heckendorn, R.B.: An analysis of the causes of code growth in genetic programming. Genetic Programming and Evolvable Machines, 3 (2002) 283–309
13. The MathWorks. (2003) `http://www.mathworks.com`
14. Van Belle, T., Ackley, D.H.: Uniform subtree mutation. In Foster, J.A., Lutton, E., Miller, J., Ryan, C., Tettamanzi, A.G.B., editors, Proceedings of EuroGP-2002, Berlin. Springer (2002) 152–161