Dynamic Limits for Bloat Control
Variations on Size and Depth

Sara Silva and Ernesto Costa

Evolutionary and Complex Systems Group
Centre for Informatics and Systems of the University of Coimbra
Polo II – Pinhal de Marrocos, 3030 Coimbra, Portugal
{sara,ernesto}@dei.uc.pt
http://cisuc.dei.uc.pt/ecos/

Abstract. We present two important variations on a recently successful bloat control technique, Dynamic Maximum Tree Depth, intended at further improving the results and extending the idea to non tree-based GP. Dynamic Maximum Tree Depth introduces a dynamic limit on the depth of the trees allowed into the population, initially set with a low value but increased whenever needed to accommodate a new best individual that would otherwise break the limit. The first variation to this idea is the Heavy Dynamic Limit that, unlike the original one, may fall again to a lower value after it has been raised, in case the new best individual allows it. The second variation is the Dynamic Size Limit, where size is the number of nodes, instead and regardless of depth. The variations were tested in two problems, Symbolic Regression and Parity, and the results show that the heavy limit performs generally better than the original technique, but the dynamic limit on size fails in the Parity problem. The possible reasons for success and failure are discussed.

1 Introduction

Genetic Programming (GP) solves complex problems by evolving populations of computer programs, using Darwinian evolution and Mendelian genetics as inspiration. Bloat is an excess of code growth caused by the genetic operators in search of better solutions, without the corresponding improvement in fitness, and it may be caused by both introns and exons [1,14]. Several explanations for bloat have been proposed (reviews in [2,13,16,17]), and many different bloat control techniques have been tried (reviews in [8,10,15]). Most of them are based on parsimony pressure, some of the latest using modified tournaments [8,9].

Recently, a new bloat control technique, Dynamic Maximum Tree Depth, was shown to be very successful in two simple problems [12], particularly when coupled with Lexicographic Parsimony Pressure [8]. Especially suited for tree-based GP, the dynamic depth technique is based on a dynamic limit on the depth of the trees allowed in the population, initially set with a low value and only raised when needed to accommodate a new best individual that would otherwise break the limit. Once this limit is raised, it will not be lowered again.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 666–677, 2004.

This paper elaborates on the dynamic depth idea, introducing two important variations aiming at further improving the results and extending the concept to non tree-based GP. The first variation is the use of a *heavy* dynamic limit that, unlike the original dynamic limit, may fall again to a lower value after it has been raised, in case the new best individual allows it. The second is the use of a dynamic limit on the tree size (number of *nodes*), instead and regardless of depth. The heavy factor is used with both depth and size limits, alone and in combination with Lexicographic Parsimony Pressure, and the results are compared with the published work [12].

2 Previous Work

Dynamic Maximum Tree Depth [12] introduces a dynamic limit to the maximum depth of the individuals allowed in the population. It is similar to the traditional static limit [4], but it does not replace it – both dynamic and strict limits are used in conjunction. The dynamic limit should be initially set with a low value at least as high as the maximum depth of the initial random trees. Any new individual who breaks this limit is rejected and replaced by one of its parents instead, as it does when it breaks the static limit, unless it is the best individual found so far. In this case, the dynamic limit increases to match the depth of the new best and allow it into the population. The dynamic limit is never decreased, and it never surpasses the static limit in any circumstance.

Lexicographic Parsimony Pressure is based on a modified tournament that always selects smaller trees when their fitness is the same. Individuals are first compared in terms of fitness, and the best one wins, regardless of its size. But if their fitness is the same, they are then compared in terms of size, defined as the number of tree nodes, and the smaller one wins.

Lexicographic Parsimony Pressure performed well in three problems where it is usual to find two different individuals with the same fitness (Parity, Multiplexer, Artificial Ant), but failed in a problem where two different individuals seldom have the same fitness (Symbolic Regression), and the bucketing variation was not able to produce much better results either [8]. Dynamic Maximum Tree Depth performed equally well in two problems of different nature (Parity, Symbolic Regression), and its combination with Lexicographic Parsimony Pressure was beneficial in most cases [12].

3 Variations on Size and Depth

We now introduce two important variations on the original work on Dynamic Maximum Tree Depth, explaining the reasons for the options taken.

3.1 Heavy Dynamic Limit

In the previous section we have learned that the results of Dynamic Maximum Tree Depth were not harmed by the use of the most restrictive initial dynamic

limit 6, instead of 9. This suggests that the technique is able to withstand a high amount of parsimony pressure without losing performance. We have also learned that the dynamic limit is never allowed to decrease during the run, even if the new best individual is less deep than the current dynamic limit (something that we have often observed), and there seems to be no good reason for this. Lowering the dynamic limit whenever the new best individual allows it is our first variation on the original technique - we have called it *heavy dynamic limit*.

As expected, whenever the limit falls back to a lower value, some individuals already in the population immediately break the new limit, becoming 'illegals'. There was a vast range of options to deal with them, the more drastic being their immediate removal from the population, possibly replacing them by new random individuals. However, when considering that these new 'illegals' could be the ones who managed to produce the new best individual, this did not seem like a very good idea. We have decided to adopt a much softer option: the 'illegals' are allowed to remain in the population as if they were not breaking the limit, but when engaging in crossover their children cannot be larger (in depth or size, depending on the case - see Sect. 3.2) than the largest parent. This naturally and slowly places the population within limits again.

3.2 Dynamic Size Limit

Dynamic Maximum Tree Depth is only suited for tree-based GP, but it is known that bloat is prone to affect any other variable-length representation [6]. The extension of the dynamic limit idea to other domains must begin with the removal of the concept of depth, replacing it with the concept of size. In tree-based GP, this means ignoring tree depth and looking only at the number of tree nodes. Our second variation on the original work is the replacement of the dynamic depth limit with what we have called the *dynamic size limit*, where size is defined as the number of nodes. Because a static inviolable limit is useful to avoid the possible exhaustion of computer resources, the dynamic size variation also uses a static limit on the number of nodes, equivalent to the static depth limit always used with the original technique (see Sect. 2).

When using the dynamic size limit, it makes no sense to keep using depth as a restriction on tree initialization. So we have created a modified version of the Ramped Half-and-Half initialization method [4], where an equal number of individuals are initialized with sizes ranging between 2 and the initial value of the dynamic size limit. For each size, an equal number of individuals are initialized with the Grow method and with the Full method [4], that we have also modified to fit the size constraints only. In the modified Grow method, the individual grows by addition of random nodes (internal or terminal) without exceeding the maximum size specified; the modified Full method chooses only internal nodes until the size is close to the specified, and only then chooses terminals. Unlike the original Full version, it may not be able to create individuals with the exact size specified, but only close (never exceeding).

4 Experiments

To test the efficacy of the variations introduced in the original idea, we have used the same simple problems studied in [12]: 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. We have performed the same battery of tests using 10 different approaches, the first four techniques being previous work (see Sect. 2):

(1) K → static depth limit [4];
(2) L → lexicographic parsimony pressure [8];
(3) D → dynamic depth limit [12];
(4) DL → D coupled with L [12];
(5) HD → heavy dynamic depth limit;
(6) HDL → HD coupled with L;
(7) N → dynamic size limit;
(8) NL → N coupled with L;
(9) HN → heavy dynamic size limit;
(10) HNL → HN coupled with L.

Table 1 summarizes the combinations of dynamic limit and tournament settings that lead to each technique. The first two techniques (K, L) only use a static limit on tree depth; the following four 'D' techniques (D, DL, HD, HDL) use dynamic limit on tree depth, initially set at 6, the value that yielded better results [12]; the last four 'N' techniques (N, NL, HN, HNL) use dynamic limit on tree size (number of nodes). Regardless of what kind of dynamic limit they use, all techniques use a static limit on depth (17) or number of nodes (512), whichever is the case. 512 seems much less than the allowed number of nodes on a 17-depth tree, but in fact it almost doubles the maximum number of nodes actually found in 17-depth trees of the experiments performed with dynamic depth. The initial values for the dynamic size limit are 21 for Symbolic Regression and 40 for Even-3 Parity. These were the values that produced distributions of tree sizes more similar to the ones obtained with dynamic depth limit 6, as determined by the Kolmogorov-Smirnov goodness-of-fit criterium. All the 'H' techniques use the heavy limit described in Sect. 3.1. We have decided to combine it with both depth and size limits. All the 'L' techniques use the modified tournament described in Sect. 2 (with no bucketing, for the sake of simplicity).

A total of 30 runs were performed with each technique for each problem. The parameters adopted for the experiments were essentially the same as in [4,8,12] to facilitate the comparison between the techniques. All the runs used populations of 500 individuals evolved during 50 generations, even when an optimal solution was found earlier. Tree initialization was performed with the Ramped Half-and-Half method [4], modified in the techniques using size limit (see Sect. 3.2). Reproduction was set at 10%, no mutation was used, and the crossover points were totally random, independently of being internal or terminal nodes. Tournament size was always 2. The (protected) function sets for the Regression and Parity problems were $\{+, -, \times, \div, sin, cos, log, exp\}$ and $\{nand, nor, and, or\}$, respectively, with no random constants used.

Table 1. Dynamic limit and tournament settings for each technique used

Technique	Dynamic limit	Heavy factor	Tournament
(1) K	-	-	standard
(2) L	-	-	modified
(3) D	depth	no	standard
(4) DL	depth	no	modified
(5) HD	depth	yes	stardard
(6) HDL	depth	yes	modified
(7) N	size	no	standard
(8) NL	size	no	modified
(9) HN	size	yes	stardard
(10) HNL	size	yes	modified

All the experiments were performed using the GPLAB toolbox [11]. Statistical significance of the null hypothesis of no difference was determined with Kruskal-Wallis non-parametric ANOVAs at $p = 0.01$.

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), calculated like in [12]; and the population diversity (Fig. 3), defined as the percentage of individuals in the population that account for the total number of (genotypically) different individuals (based on variety in [5]). The p-values concerning the best (lowest) fitness of run are also presented, in Table 2. The boxplots are based on the mean values, 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. A dot on the bottom of the lower whisker indicates there are no outliers. 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 both problems, all the techniques using dynamic limits achieved significantly lower tree sizes than K and L, confirming and strengthening the results reported in [12]. In the Regression problem, the heavy depth techniques (HD, HDL) caused a significant decrease in tree sizes when compared to the original depth techniques (D, DL) and to any of the dynamic size techniques (the last four). The differences

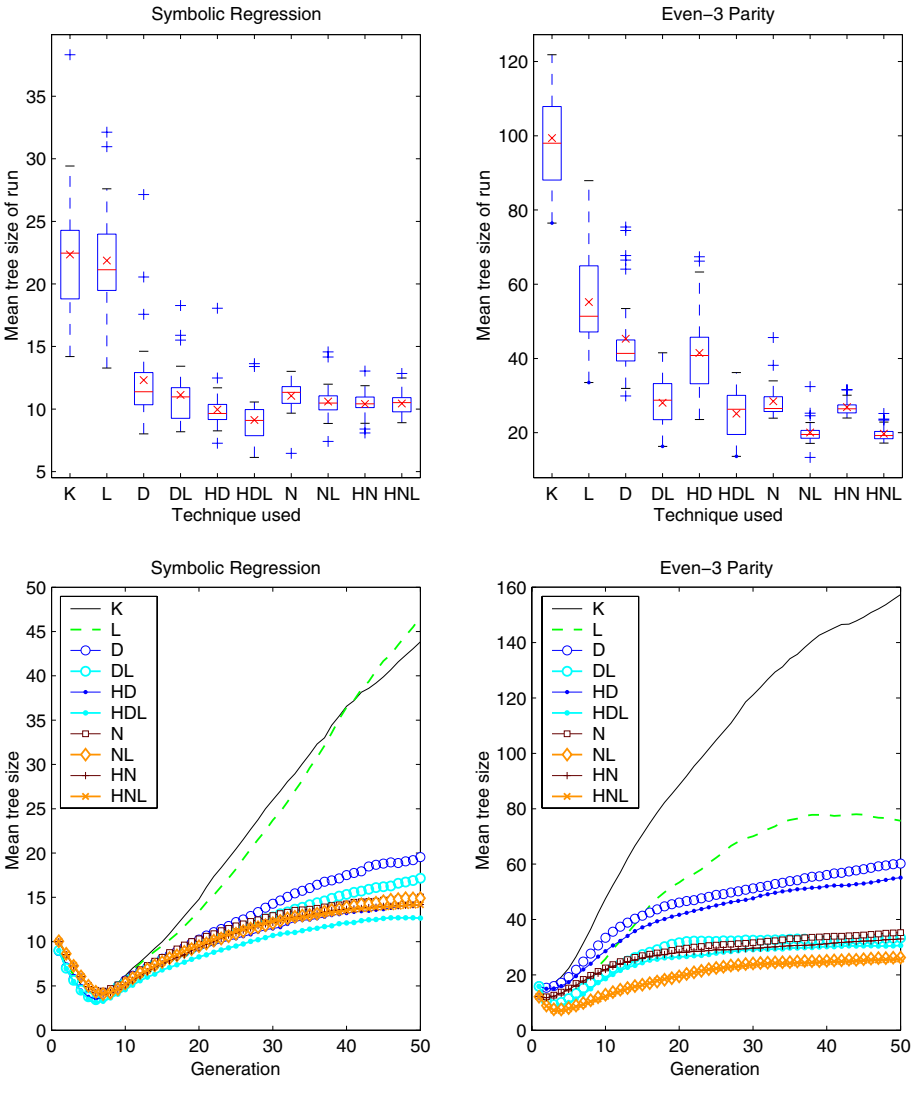

Fig. 1. Boxplots and evolution curves of mean tree size

introduced by the L tournament are not considered significant in any case. In general, the techniques based on size were not significantly better than the ones based on depth. Nevertheless, the evolution curves show that, by the end of the run, the lower tree sizes are achieved by both the heavy depth techniques and by all the size techniques.

In the Parity problem, the heavy factor never caused any significant difference. On the other hand, the introduction of the L tournament caused a signif-

icant decrease in tree sizes, in all cases. When comparing the techniques based on size with the ones based on depth, only the size techniques coupled with the L tournament (NL, HNL) were always significantly better than any of the depth techniques. The evolution curves show no surprises. The fact that the difference in mean tree size between the best and the worst techniques surpasses 130 nodes is noteworthy.

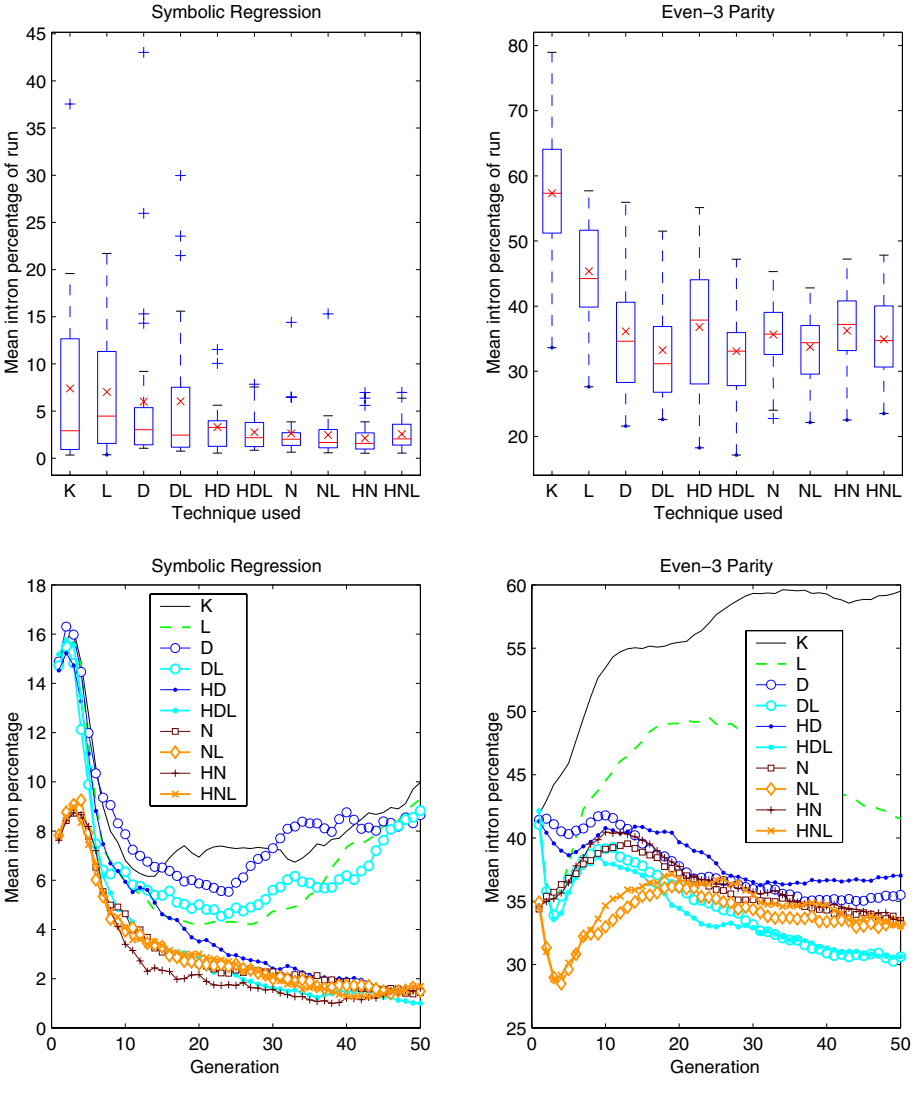

Fig. 2. Boxplots and evolution curves of mean intron percentage

5.2 Mean Percentage of Introns

Figure 2 shows the results concerning the mean percentage of introns. The tree initialization procedure used with the dynamic size techniques produces comparatively lower percentages of introns in both problems, although tree sizes are very similar (Fig. 1). In the Regression problem we realize that, as in [12], code growth is not caused by introns. The differences between the several techniques using dynamic limits are not considered significant, but the evolution curves show that, by the end of the run, only the new variations (heavy factor and dynamic size) continue lowering the percentage of introns, unlike in [12]. In the Parity problem, there are also no significant differences between any of the techniques using dynamic limits, and no marked tendency is apparent in the evolution curves either.

5.3 Population Diversity

Figure 3 shows the results concerning the population diversity. In the Regression problem, the techniques based on size are all capable of sustaining significantly higher diversity than the ones based on depth, but still perform worse than K and L. The influence of the heavy factor and the L tournament is considered significant in only half of the cases. The evolution curves show a marked decrease in diversity when the run starts (like in [12]), and by the end of the run the heavy depth techniques (HD, HDL) distinguish themselves by the lower diversity, particularly HDL that ends with a value almost as low as 50%.

In the Parity problem, once again the techniques based on size can sustain significantly higher diversity than the ones based on depth. The L tournament is responsible for significant differences in population diversity, but only because of a sharp initial decrease observed in the evolution curves, similar to what happened in the Regression problem. This phenomenon was also apparent in [12], where a plausible explanation has been proposed.

5.4 Best Fitness

Table 2 shows the p-values concerning the best fitness of run. Although this table contains more detailed information than the plots would, these values should not be interpreted as anything more than mere indicators of performance and suitability of the new techniques under study.

In the Regression problem, it is worth noting that both DL and HD yield significantly *better* results than L or most of the dynamic nodes techniques. The heavy dynamic depth did not decrease the ability to find individuals with good fitness. Although the dynamic limit on nodes only fails when compared to the winners DL and HD, we also notice substantially lower p-values when comparing to K and D.

In the Parity problem, once again the heavy dynamic depth performed as well as any of the previously published techniques (the first four). On the contrary, the dynamic limit on nodes performed significantly *worse* than the rest, except N (although with substantially lower p-values).

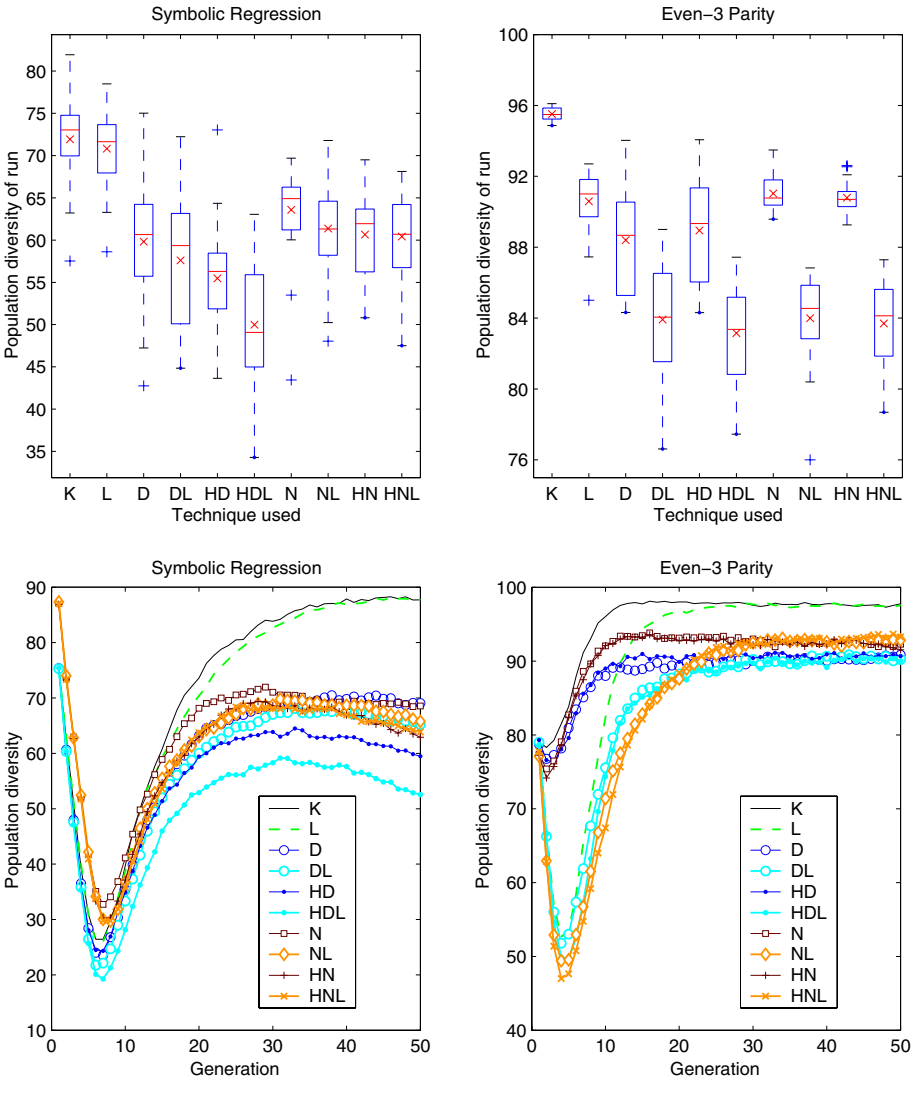

Fig. 3. Boxplots and evolution curves of population diversity

6 Discussion and Conclusions

The introduction of the heavy factor into the dynamic limit adds parsimony pressure during the run. Even without taking drastic measures towards the individuals that suddenly break the new limit, the results show that the heavy limit is capable of achieving even lower mean tree sizes than the original limit, particularly when applied to depth in the Regression problem. This decrease in

Table 2. p-values concerning the best fitness of run, using non-parametric ANOVAs. Statistical significance is considered where $p<0.01$ (in bold). See Sect. 5.4 for details

Even-3 Parity

	K	L	D	DL	HD	HDL	N	NL	HN	HNL	
K		1.000	0.393	0.720	0.690	0.492	0.120	**0.000**	**0.001**	**0.000**	K
L	0.144		0.393	0.720	0.690	0.492	0.120	**0.000**	**0.001**	**0.000**	L
D	0.719	0.057		0.232	0.643	0.132	0.021	**0.000**	**0.000**	**0.000**	D
DL	0.057	**0.000**	0.063		0.451	0.741	0.226	**0.000**	**0.003**	**0.000**	DL
HD	0.271	**0.001**	0.332	0.165		0.282	0.055	**0.000**	**0.000**	**0.000**	HD
HDL	0.599	0.306	0.443	0.014	0.110		0.375	**0.000**	**0.008**	**0.000**	HDL
N	0.126	0.682	0.089	**0.002**	0.011	0.369		**0.002**	0.069	**0.005**	N
NL	0.033	0.467	0.012	**0.000**	**0.000**	0.101	0.669		0.188	0.783	NL
HN	0.084	0.700	0.021	**0.000**	**0.001**	0.216	0.844	0.752		0.296	HN
HNL	0.052	0.431	0.031	**0.001**	**0.002**	0.179	0.935	0.976	0.724		HNL
	K	L	D	DL	HD	HDL	N	NL	HN	HNL	

Symbolic Regression

mean tree size is, however, accompanied by a proportional decrease in population diversity, although this makes no difference on the ability to converge to the optimal solution. The concerns regarding how depth limitations may adversely affect GP performance [3,7] do not seem to apply here.

On the other hand, the dynamic size limit did not perform so well. In spite of achieving similarly low mean tree sizes, and without losing so much diversity, it did not perform better than the best depth techniques on the Regression problem. On the Parity problem it clearly failed, with the last three techniques achieving significantly worse results, as if the amount of pressure that the problem can withstand was finally surpassed.

It is arguable whether the dynamic limit on size instead of depth increases the amount of pressure. It is clear that, when dealing with depth, there is usually the opportunity to add more nodes without breaking the limit, because trees are always far from full (see Fig. 4). On the other hand, looking at size regardless of depth removes a very important restriction from the search: the shape of the trees. It is true that when using the size limit instead of the depth limit, trees are comparatively less full. Could this account for the higher diversity achieved? A comparison with Fig. 3 reveals that mean tree fill rate seems to be in fact inversely related to population diversity – even the sudden loss of diversity observed in the beginning of the run is accompanied by a small increase in fill rate. So it may not be the case that a higher number of parent clones, resulting from their offspring being rejected, is the sole reason for diversity loss. The relationship between the two is not even clear (plots not shown), although this is still work in progress. Quite surprisingly, diversity loss (or higher tree fill rate) does not seem to jeopardize performance. The general poor fitness results achieved by the dynamic size techniques have a different cause, and we speculate that it

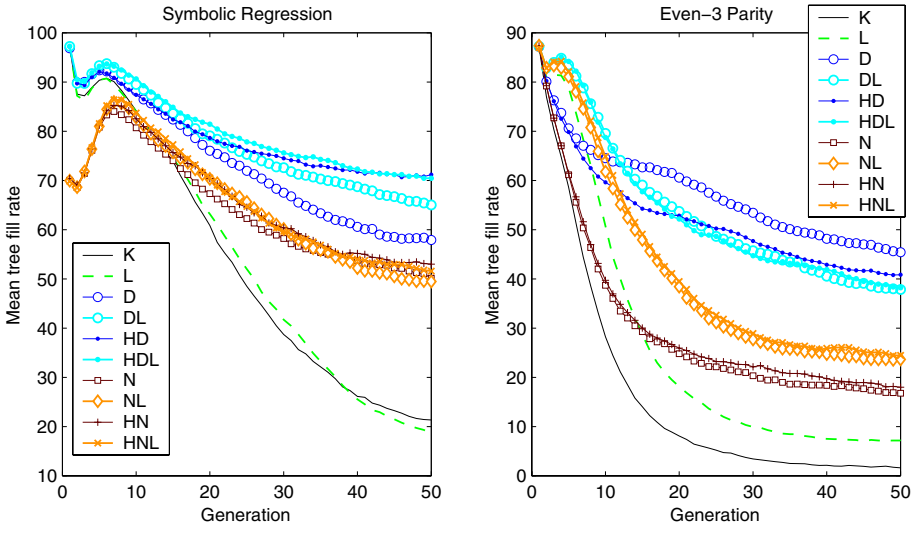

Fig. 4. Evolution curves of mean tree fill rate. Tree fill rate is the percentage of tree nodes in relation to the expected number of nodes of a full tree, given its depth and the function and terminal sets being used

might be related to the difficulty in slightly altering the number of nodes without conflicting with the current limit. This small window of freedom, present in all dynamic depth techniques (see previous paragraph), may be the crucial factor that the dynamic size lacks. If this supposition is true, a simple modification to the dynamic size techniques may easily solve the problem, for example by allowing individuals to always grow a little beyond the limit, regardless of their fitness.

7 Future Work

The whole idea of using dynamic limits should be tested in different, harder problems, and different sets of parameters should also be used to ensure the robustness of the technique. Considering the high applicability that the dynamic size limit may have in other domains, some effort should be made in order to fully understand and possibly eliminate the reasons behind its apparent failure, by introducing the elements responsible for the excellent results achieved by the best dynamic depth techniques. Some insight may be obtained by further studying how the number of parent clones relates to population diversity (where a different, possibly phenotypical, diversity measure may be more informative), and how do tree shape restrictions affect this relationship.

Acknowledgements. We acknowledge grants SFRH/BD/14167/2003 and POCTI/1999/BSE/34794 from Fundação para a Ciência e a Tecnologia, Portugal. We would also like to thank the members of the ECOS – Evolutionary and Complex Systems Group (Universidade de Coimbra) and Pedro J.N. Silva (Universidade de Lisboa) for the valuable suggestions provided throughout this research.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., Francone, F.D.: Genetic programming – an introduction, San Francisco, CA. Morgan Kaufmann (1998)
2. Brameier, M., Bankhaf, W.: Neutral variations cause bloat in linear GP. In C. Ryan *et al.*, editors, Proceedings of EuroGP-2003, Berlin. Springer (2003) 286–296
3. Gathercole, C., Ross, P.: An adverse interaction between crossover and restricted tree depth in genetic programming. In J.R. Koza *et al.*, editors, Proceedings of GP'96, Cambridge, MA. MIT Press (1996) 291–296
4. Koza, J.R.: Genetic programming – on the programming of computers by means of natural selection, Cambridge, MA. MIT Press (1992)
5. Langdon, W.B.: Genetic Programming + Data Structures = Automatic Programming!, Boston, MA. Kluwer (1998)
6. Langdon, W.B.: The evolution of size in variable length representations. In D. Fogel, editor, Proceedings of ICEC'98. IEEE Press (1998) 633-638
7. Langdon, W.B., Poli, R.: An analysis of the MAX problem in genetic programming. In J.R. Koza *et al.*, editors, Proceedings of GP'97, San Francisco, CA. Morgan Kaufman (1997) 222–230
8. Luke, S., Panait, L.: Lexicographic parsimony pressure. In W.B. Langdon *et al.*, editors, Proceedings of GECCO-2002, San Francisco, CA. Morgan Kaufmann (2002) 829–836
9. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In J.J. Merelo Guervós *et al.* , editors, Proceedings of PPSN-2002, Berlin. Springer Verlag (2002) 411–421
10. Poli, R.: A simple but theoretically-motivated method to control bloat in genetic programming. In C. Ryan *et al.*, editors, Proceedings of EuroGP-2003, Berlin. Springer (2003) 204–217
11. Silva, S.: GPLAB – a genetic programming toolbox for MATLAB. (2004) http://gplab.sourceforge.net
12. Silva, S., Almeida, J.: Dynamic maximum tree depth – a simple technique for avoiding bloat in tree-based GP. In E. Cantú-Paz *et al.*, editors, Proceedings of GECCO-2003, Berlin. Springer Verlag (2003) 1776–1787
13. Soule, T.: Code growth in genetic programming. PhD thesis, University of Idaho (1998)
14. Soule, T.: Exons and code growth in genetic programming. In J.A. Foster *et al.*, editors, Proceedings of EuroGP-2002, Berlin. Springer (2002) 142–151
15. Soule, T., Foster, J.A.: Effects of code growth and parsimony pressure on populations in genetic programming. Evolutionary Computation, 6(4) (1999) 293–309
16. 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
17. Streeter, M.J.: The root causes of code growth in genetic programming. In C. Ryan *et al.*, editors, Proceedings of EuroGP-2003, Berlin. Springer (2003) 449–458