Fighting Bloat With Nonparametric Parsimony Pressure

Sean Luke and Liviu Panait

Department of Computer Science
George Mason University
Fairfax, VA 22030 USA
`http://www.cs.gmu.edu/~sean/`
`http://www.cs.gmu.edu/~lpanait/`

Abstract. Many forms of parsimony pressure are parametric, that is final fitness is a parametric model of the actual size and raw fitness values. The problem with parametric techniques is that they are hard to tune to prevent size from dominating fitness late in the evolutionary run, or to compensate for problem-dependent nonlinearities in the raw fitness function. In this paper we briefly discuss existing bloat-control techniques, then introduce two new kinds of non-parametric parsimony pressure, Direct and Proportional Tournament. As their names suggest, these techniques are based on simple modifications of tournament selection to consider both size and fitness, but not together as a combined parametric equation. We compare the techniques against, and in combination with, the most popular genetic programming bloat-control technique, Koza-style depth limiting, and show that they are effective in limiting size while still maintaining good best-fitness-of-run results.

1 Introduction

One of the foremost challenges to scaling genetic programming (GP) is *bloat*, the tendency for genetic programming individuals to grow in size as evolution progresses, relatively independent of any justifying improvements in fitness. It is not uncommon for the average size of an individual to grow over three orders of magnitude in just fifty generations. This is a serious stumbling block to genetic programming, as it slows the evolutionary search process, consumes memory, and can hamper effective breeding. Bloat produces a sort of Zeno's paradox, slowing successive generations by so much that it places a cap on GP's useful runtime.

Bloat is not a problem unique to genetic programming. It occurs in a wide variety of arbitrary-length representations, including neural networks, finite state automata, and rule sets. Indeed, the earliest known report of bloating (and of approaches to deal with it) involves evolving Pitt-approach rule systems [1]. However, because GP is the most popular arbitrary-length representation technique, the lion's share of papers on the subject have been in the GP literature.

As discussed in [2], bloating is a hotly debated topic in GP theory; and there is also presently no silver bullet to deal with it. The genetic programming literature uses a large number of different techniques to counter bloat, each with its own advantages and disadvantages. By far the most popular such technique is restricting breeding to only produce children less than some maximal tree depth. A distant second is *parsimony pressure*, where the size of an individual is a factor in its probability of being selected.

In this paper we examine two new approaches to parsimony pressure, and report on their success in managing population size while retaining reasonable best-fitness-of-run results as compared to Koza-style tree depth limitation. These techniques are based on modifications of the well-studied *tournament selection* method, which picks *N* random individuals with replacement, then selects the fittest individual from that pool. We modify tournament selection to consider parsimony as well as fitness, but retain the nonparametric features which make tournament selection popular.

2 Previous Bloat Control Techniques

Modification and Restriction Techniques. The most common approach to bloat control, at least in the GP literature, is *maximal depth restriction* [3]. Here, when a child is created by removing a subtree from a parent and replacing it with another subtree (as is done in subtree crossover or subtree mutation), and the child exceeds a maximal depth limit, then the child is rejected and a copy of the original parent takes its place in the new generation. The standard maximal depth limit for tree-based GP is 17. It is also possible, but much less common, to place size restrictions on the child rather than depth restrictions. *Pseudo-hillclimbing* [4] is a recent restriction approach: when a child is generated, its fitness is immediately assessed. If its fitness is not superior to its parent's fitness, it is rejected and a copy of its parent takes its stead in the new generation. Note that this approach is very similar to depth restriction in mechanism, except that oddly it does not compare size at all, yet is reasonably successful at limiting tree growth. This gives some insight into why depth restriction and pseudo-hillclimbing are successful: they limit growth not only by capping size but by injecting large numbers of parents directly into later generations. As parents are generally smaller than their children (hence the bloat), this has a stunting effect, but at a cost in diversity.

As GP parse trees may generally be viewed as computer functions, one obvious way to counter bloat is to perform *code editing* to optimize those functions. One paper [5] reports strong results with this approach, but there is evidence that editing may lead to premature convergence [6]. Finally, a number of papers (for example [7]) have investigated allowing GP parse trees to adapt, on a component-by-component basis, the probability that a given component will be chosen for crossover. This bloat-control technique is known as *explicitly defined introns*.

Parsimony Pressure Parsimony pressure is the popular bloat-control technique outside of GP, and is gaining popularity within GP as well. Most such parsimony pressure computes fitness as a linear function of an individual's raw fitness and its size (for example, [8]), though there are some nonlinear examples as well [9]. For a more complete survey of linear and other kinds of parametric parsimony pressure, see [10].

The trouble with parametric parsimony pressure is that it is *parametric*. We mean this in the statistical sense: it considers the actual *values* of size and fitness together in a parametric statistical model for selection: the experimenter must stipulate, in effect, that *N* units of size are worth *M* units of raw fitness. Stipulating this function is problematic when fitness is a nonlinear function of actual "worth", as is often the case: fitness functions are typically ad-hoc. It may well be that a difference between 0.9 and 0.91 in fitness is much more significant than a difference between 0.7 and 0.9. Parametric parsimony pressure can thus give size an unwanted advantage over fitness when the difference in fitness is only 0.01 as opposed to 0.2. This is also a problem because the relative significance of exact size and fitness parameters changes during the course of a run. For example, size-parameter dominance may arise late in evolution, when subtle differences in fitness become important. Notice that these issues are similar to those which gave rise to the preference of tournament selection and other nonparametric selection procedures over fitness-proportionate selection.

One approach to fixing this is to adapt the size parameter as the evolutionary run progresses [11], except that such techniques must usually rely on problem-specific analysis. Another recent approach, *pareto parsimony pressure*, eschews parametric techniques and instead treats size as a second objective in a pareto-optimization scheme. Pareto optimization is used when the evolutionary system must optimize for two or more objectives at once, and it is not clear which objective is "more important". An individual A is said to *pareto-dominate* another individual B if A is as good as B in all objectives, and better than B in at least one objective. One possible use of pareto dominance is to stipulate that an individual's fitness is the number of peers it dominates. Unfortunately, the technique has so far had mixed results in the literature. Some papers report smaller trees and the discovery of more ideal solutions [12, 13], but tellingly they omit best-fitness-of-run results. Another reports the mean best-fitness-of-run, but it is *worse* than when not using the technique [14].

3 Two New Parsimony Pressure Techniques

The two new techniques we propose here are modifications of the tournament selection operator. The techniques are *double tournament*, where individuals must pass two layers of tournaments (one by size, one by fitness) to be selected; and *proportional tournament*, where the tournament sometimes picks by size, and sometimes by fitness.

Double Tournament The double tournament algorithm selects an individual using tournament selection: however the tournament contestants are not chosen at random with replacement from the population. Instead, they were each the winners of *another* tournament selection. For example, imagine if the "final" tournament has a pool size of 7: then seven "qualifier" tournaments are held as normal in tournament selection, and the winners go on to compete in the "final" tournament. Double tournament has been previously used to select for both fitness and diversity [15]. We suggest it may be used for parsimony pressure by having the "final" tournament select based on parsimony while the qualifying tournaments select based on fitness (or vice versa). The algorithm has three parameters: a fitness tournament size S_f, a parsimony tournament size S_p, and a switch (*do-fitness-first*) which indicates whether the qualifiers select on fitness and the final selects on size, or (if false) the other way around.

Our initial experiments revealed that even S_p values as small as 2 put too much pressure on parsimony, and the fitnesses of the resulting individuals were statistically significantly worse than with no parsimony pressure at all. In order to rectify this matter, we permit S_p to hold real values between 1.0 and 2.0. In this value range, two individuals participate to the tournament; with probability $S_p/2$ the smaller individual wins, else the larger individual wins. Ties are broken at random. Thus $S_p = 1$ is random selection, while $S_p = 2$ is the same as a plain parsimony-based tournament selection of size 2.

Proportional Tournament This technique is even simpler. The proportional tournament algorithm selects an individual using tournament selection as usual, using some fixed tournament size S. However, a proportion of tournaments will select based on parsimony rather than on fitness. A fixed parameter R defines the proportion, where higher values of R imply more of an emphasis towards fitness: $R = 1$ implies that all tournaments will select based on fitness, while $R = 0.5$ implies that tournaments will select on fitness or size with equal probability.

4 Experiments

The bloat-control technique most used in the literature is Koza-style depth limiting, and we, like most of the literature, compare our technique against it. In future work, we will also compare against linear or pareto parsimony pressure. To this end, we performed two experiments. The first experiment compared depth limiting against Double and Proportional tournaments, while the second compared plain depth limiting against depth limiting in combination with the tournaments. We wish to emphasize that although these techniques are being used for GP in this paper, they are general techniques which are representation-independent.

The experiments used population sizes of 1000, with 50-generation runs. The runs did not stop when an ideal individual was found. Runs with plain depth limiting used plain tournament selection with a tournament size of 7. We chose four problem domains: Artificial Ant, 11-bit Boolean Multiplexer, Symbolic Regression, and Even-5 Parity. We followed the Koza-standard parameters specified in these four domains as set forth in [3], as well as its breeding, selection, and tree generation parameters. Artificial Ant used the Santa Fe food trail. Symbolic Regression used no ephemeral random constants. To compare means for statistical significance, we used ANOVAs with a 95% confidence. The evolutionary computation system used was ECJ 7 [16].

Our results are graphed as follows. For the Double Tournament, we set $S_f = 7$ and let S_p range from 1.0 to 2.0 by increments of 0.1. We experimented with setting *do-fitness-first* to false (leftmost methods in the graphs), and to true (the next set of methods). For the Proportional Tournament, we set $S = 7$, and let R range from 1.0 down to 0.5 by decrements of 0.05. In all graphs, lower fitnesses were better, and the rightmost bar represents plain depth-limiting alone.

4.1 First Experiment

The first experiment compared plain depth limiting against Double and Proportional Tournament, using the four problem domains listed above. We ran 50 runs per technique per problem domain, and plotted the mean best-fitness-of-run and the average tree size per run. The fitness results are shown in Figure 2, and the tree size results in Figure 1.

Results. For most problems, there existed non-extreme settings for both Double and Proportional Tournaments which maintained fitness with significantly smaller tree sizes than plain depth limiting, often by wide margins. In the Symbolic Regression domain, Double and Proportional Tournaments both improved on plain depth limiting in tree size and fitness, but never in a statistically significant manner. As we noted in a previous paper [10], plain depth limiting performs very well in Symbolic Regression. Overall, Double Tournament S_p values in the 1.4–1.6 range did reasonably well. The sweet spot for Proportional Tournament was around $R = 0.7$, which always performed nearly identically to plain depth limiting (other settings could do much better depending on the problem). The particular setting of *do-fitness-first* did not have a significant effect.

4.2 Second Experiment

Reasonable settings of Double and Proportional Tournament either equalled or outperformed plain depth limiting, but not by as wide a margin as we would have hoped. We wondered how well combining each of these two methods with depth limiting would perform against just plain depth limiting alone. In our second experiment we compared the combinations against depth limiting, once again doing 50 runs per technique, then plotted the best fitness per run and the average tree size per run. The fitness results are shown in Figure 4, and the tree size results are shown in Figure 3.

Results. This time judicious settings of S_p or R dramatically outperformed plain depth limiting in all four domains. Overall, Double Tournament S_p values in 1.2–1.6 had equal fitness to plain depth limiting, while significantly outperforming it in tree size, often halving the size. The sweet spot for Proportional Tournament was again around $R = 0.7$, which always halved tree size while maintaining statistically equivalent fitness. Again, the particular setting of *do-fitness-first* did not have a significant effect.

5 Conclusions and Future Work

When it comes to fitness, plain depth limiting is hard to beat. The techniques discussed in this paper all had statistically equivalent best-fitness-of-run results as depth limiting, but not better. However they were able to achieve these results while lowering the tree size. Double Tournament and Proportional Tournament by themselves could only lower total tree size slightly in comparison to plain depth limiting. However, when *combined* with depth limiting, they significantly outperformed depth limiting alone, yielding tree sizes at half the normal size while maintaining an equivalent best fitness of run. Given their simple implementation and general applicability, we think that non-parametric tournament-based parsimony pressure is worth consideration in a GP system in combination with depth limiting. As future work we hope to examine the applicability of these techniques to non-GP environments as well, and in comparison with other parsimony pressure methods.

References

1. Stephen F. Smith. *A Learning System Based on Genetic Adaptive Algorithms*. PhD thesis, Computer Science Department, University of Pittsburgh, 1980.
2. Sean Luke. *Issues in Scaling Genetic Programming: Breeding Strategies, Tree Generation, and Code Bloat*. PhD thesis, Department of Computer Science, University of Maryland, A. V. Williams Building, University of Maryland, College Park, MD 20742 USA, 2000.
3. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.
4. Terence Soule and James A. Foster. Removal bias: a new cause of code growth in tree based evolutionary programming. In *1998 IEEE International Conference on Evolutionary Computation*, pages 781–186, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press.
5. Terence Soule, James A. Foster, and John Dickinson. Code growth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 215–223, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
6. Thomas Haynes. Collective adaptation: The exchange of coding segments. *Evolutionary Computation*, 6(4):311–338, Winter 1998.
7. Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, pages 111–134. MIT Press, Cambridge, MA, USA, 1996.
8. Donald S. Burke, Kenneth A. De Jong, John J. Grefenstette, Connie Loggia Ramsey, and Annie S. Wu. Putting more genetics into genetic algorithms. *Evolutionary Computation*, 6(4):387–410, Winter 1998.
9. Jeffrey K. Bassett and Kenneth A. De Jong. Evolving behaviors for cooperating agents. In *International Syposium on Methodologies for Intelligent Systems*, pages 157–165, 2000.
10. Sean Luke and Liviu Panait. Lexicographic parsimony pressure. In W. B. Langdon *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002)*. Morgan Kaufmann, 2002.
11. Byoung-Tak Zhang and Heinz Mühlenbein. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, 3(1):17–38, 1995.
12. Stefan Bleuler, Martin Brack, Lothar Thiele, and Eckhart Zitzler. Multiobjective genetic programming: Reducing bloat using spea2. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 536–543, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.
13. Edwin D. DeJong, Richard A. Watson, and Jordan B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 11–18, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
14. Aniko Ekart and S. Z. Nemeth. Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, 2(1):61–73, March 2001.
15. Markus Brameier and Wolfgang Banzhaf. Explicit control of diversity and effective variation distance in linear genetic programming. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 37–49, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.
16. Sean Luke. ECJ 7: An EC system in Java. http://www.cs.umd.edu/projects/plus/ec/ecj/, 2001.

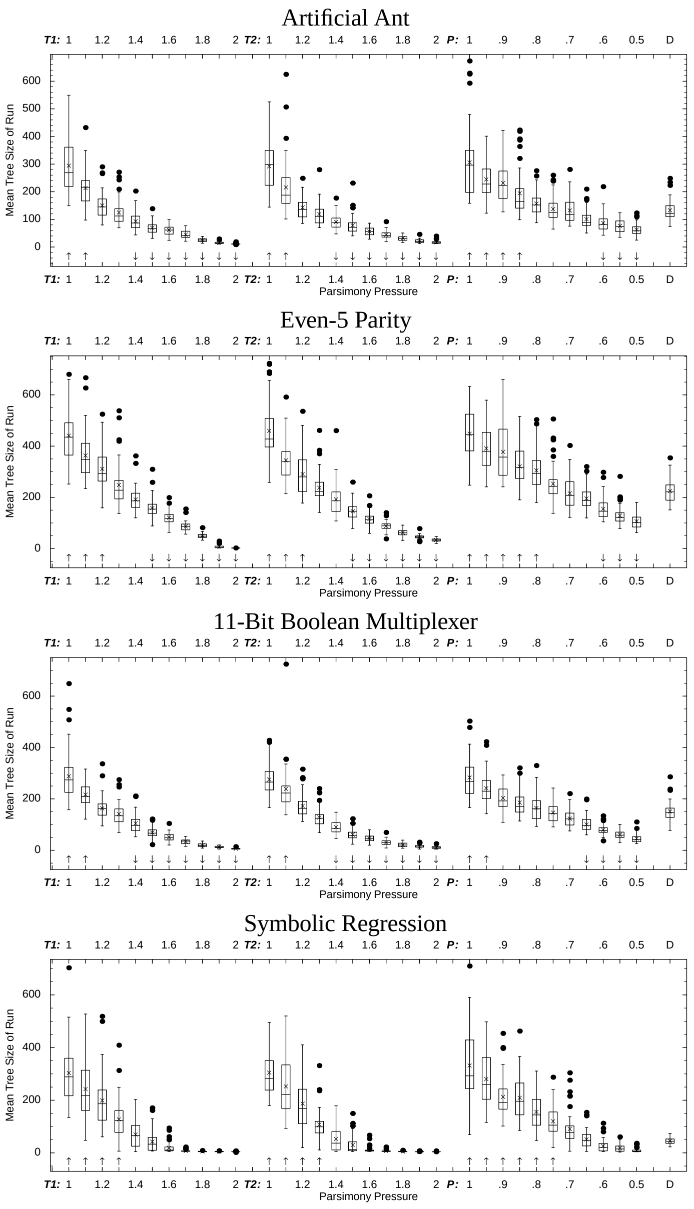

Fig. 1. Mean tree sizes for various parsimony pressure methods, as compared compared to plain depth limiting (labeled *D*). Distributions are plotted with boxplots. Proportional Tournament is labeled *P*, with the given ratio value *R*. Double Tournament is labeled *T1* (*do-fitness-first* false) or *T2* (*do-fitness-first* true), with the given tournament size S_p. The mean of each distribution is indicated with an $\times$. Lower values are better. Techniques statistically superior to plain depth limiting are marked with $\downarrow$; techniques statistically inferior are marked with $\uparrow$

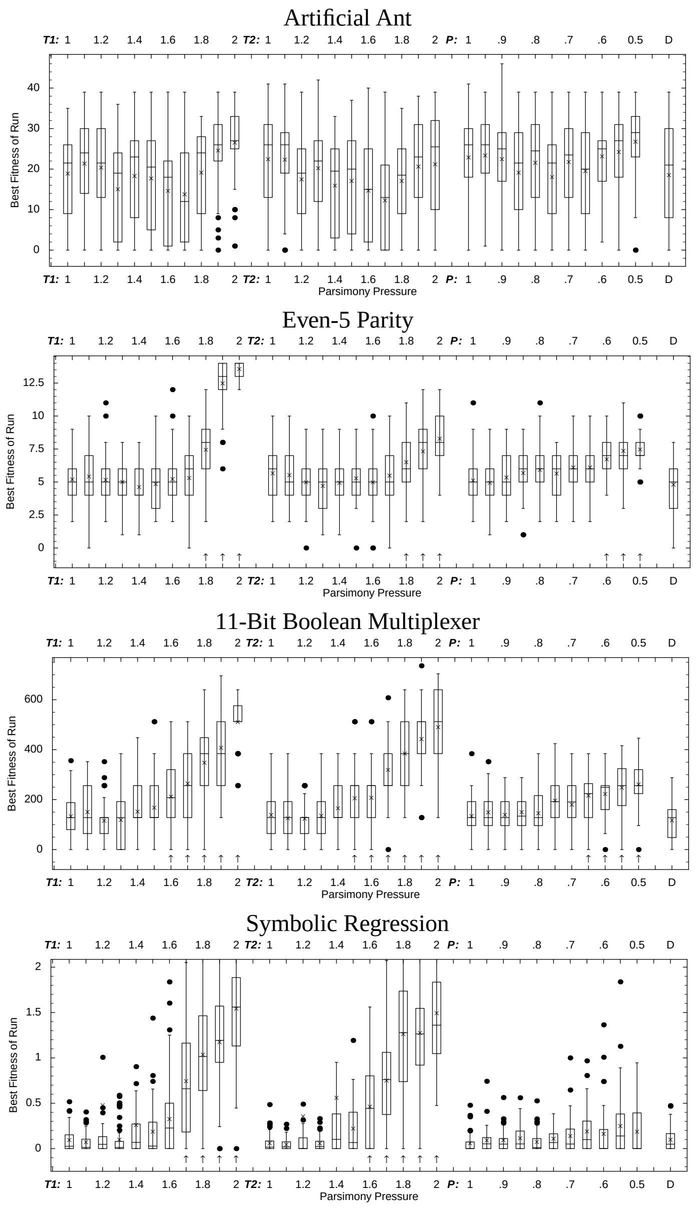

Fig. 2. Best fitness of run for various parsimony pressure methods, as compared compared to plain depth limiting (labeled *D*). Distributions are plotted with boxplots. Proportional Tournament is labeled *P*, with the given ratio value *R*. Double Tournament is labeled *T1* (*do-fitness-first* false) or *T2* (*do-fitness-first* true), with the given tournament size S_p. The mean of each distribution is indicated with an $\times$. Lower values are better. Techniques statistically superior to plain depth limiting are marked with $\downarrow$; techniques statistically inferior are marked with $\uparrow$

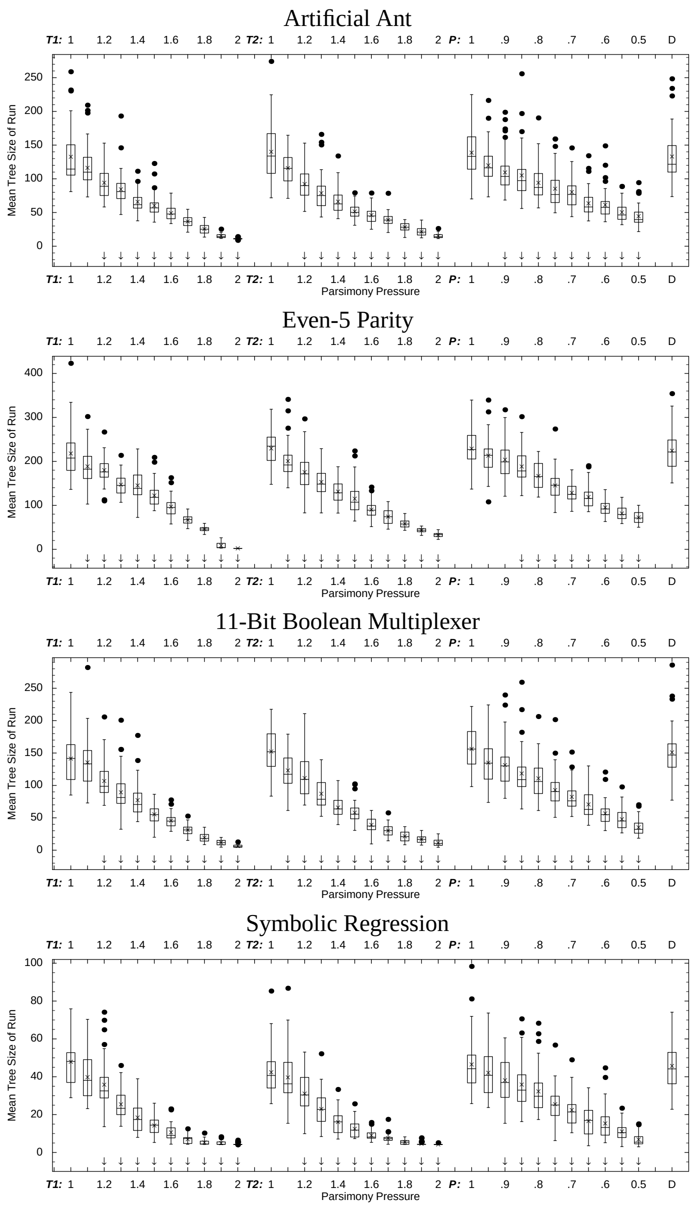

Fig. 3. Mean tree sizes for various parsimony pressure methods in combination with depth limiting, as compared compared to plain depth limiting alone (labeled *D*). Distributions are plotted with boxplots. Proportional Tournament is labeled *P*, with the given ratio value *R*. Double Tournament is labeled *T1* (*do-fitness-first* false) or *T2* (*do-fitness-first* true), with the given tournament size S_p. The mean of each distribution is indicated with an $\times$. Lower values are better. Techniques statistically superior to plain depth limiting are marked with $\downarrow$; techniques statistically inferior are marked with $\uparrow$

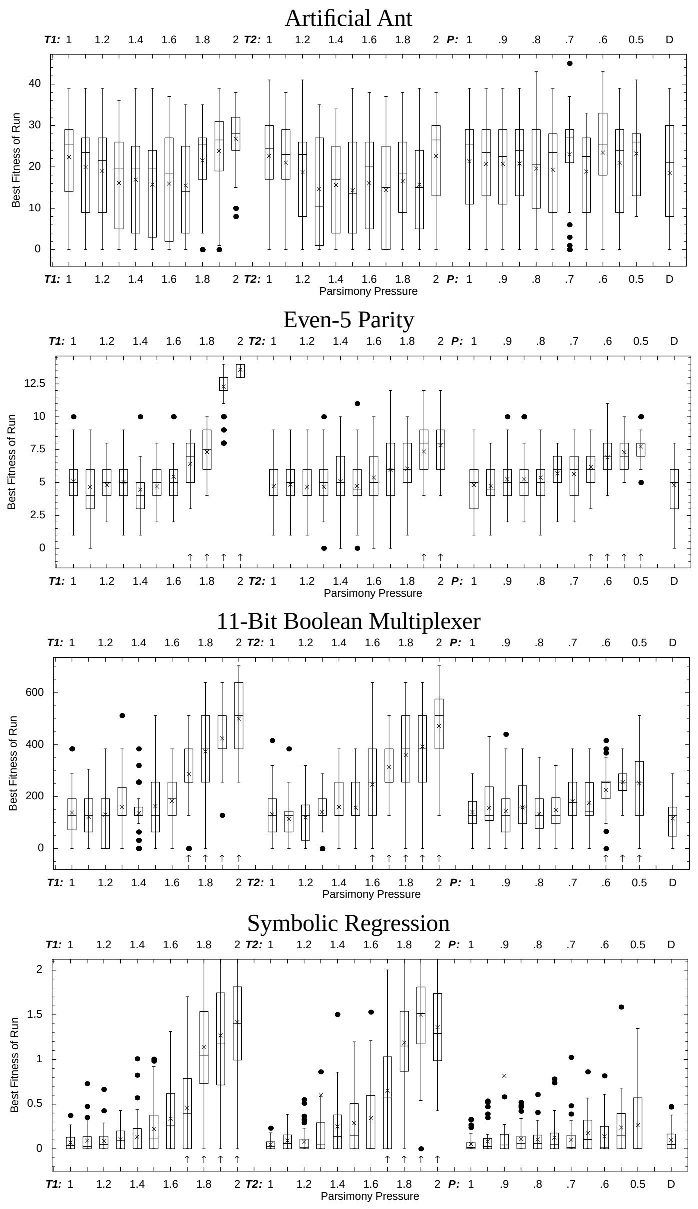

Fig. 4. Best fitness of run for various parsimony pressure methods in combination with depth limiting, as compared compared to plain depth limiting alone (labeled *D*). Distributions are plotted with boxplots. Proportional Tournament is labeled *P*, with the given ratio value *R*. Double Tournament is labeled *T1* (*do-fitness-first* false) or *T2* (*do-fitness-first* true), with the given tournament size S_p. The mean of each distribution is indicated with an $\times$. Lower values are better. Techniques statistically superior to plain depth limiting are marked with $\downarrow$; techniques statistically inferior are marked with $\uparrow$