Alternative Bloat Control Methods

Liviu Panait and Sean Luke

George Mason University, Fairfax, VA 22030
lpanait@cs.gmu.edu, sean@cs.gmu.edu

Abstract. Bloat control is an important aspect of evolutionary computation methods, such as genetic programming, which must deal with genomes of arbitrary size. We introduce three new methods for bloat control: Biased Multi-Objective Parsimony Pressure (BMOPP), the Waiting Room, and Death by Size. These methods are unusual approaches to bloat control, and are not only useful in various circumstances, but two of them suggest novel approaches to attack the problem. BMOPP is a more traditional parsimony-pressure style bloat control method, while the other two methods do not consider parsimony as part of the selection process at all, but instead penalize for parsimony at other stages in the evolutionary process. We find parameter settings for BMOPP and the Waiting Room which are effective across all tested problem domains. Death by Size does not appear to have this consistency, but we find it a useful tool as it has particular applicability to steady-state evolution.

1 Introduction

When evolutionary computation uses arbitrary-sized representations, often the evolutionary process drives not only towards fitter individuals, but often dramatically larger individuals. This rapid increase in size, known as *bloat* (or as Bill Langdon calls it, "survival of the fattest") can hinder the evolutionary mechanism itself and can slow successive generations to the point that further progress is not feasible. Bloat occurs across the evolutionary computation landscape (for example, [1, 2]), but most attention paid to it has been in the context of genetic programming (GP).

In this paper we will introduce three new methods for controlling bloat, *Biased Multi-Objective Parsimony Pressure, the Waiting Room,* and *Death by Size* respectively. These methods are effective, and some (such as Death by Size) are applicable to special evolutionary methods such as steady-state evolution. Because it is nearly always helpful, we combine the methods with the most common method of GP bloat-control, namely establishing fixed limits on tree depth, and compare them against using tree depth limits alone.

The dynamics of bloat are complex [3] and in the absence of a widely-accepted generalized theory of bloat, most effective methods for dealing with the problem are justified empirically. However empirical comparisons must take into consideration two simultaneous, and often contradictory, objectives: increasing fitness and reducing size. We will attempt to ascertain settings for these methods which seem to find a good middle-ground between these objectives: specifically, we view one method as better than another if its fitness is not statistically significantly worse, but its tree size is significantly smaller.

We begin with a discussion of previous approaches to bloat control. We then introduce the new methods and argue for their effectiveness in a series of experiments using various tree-style genetic programming test problems. We then conclude with a brief discussion.

1.1 Previous Approaches to Bloat Control

The most common approach to bloat control is establishing hard limits on size or depth, primarily because this approach was popularized by early work in genetic programming [4]. In such tree-based genetic programming, if a newly-created child is deeper than 17 nodes, it is rejected and a parent is introduced into the population in its stead[1]. Depth limiting has been criticized in the past for its effect on breeding [7], but it has proven a surprisingly successful method [8, 9]. In the experiments in this paper we will augment our new methods with depth limiting (because in all cases it is beneficial to them), and compare this combination against plain depth limiting alone.

Outside of genetic programming, the common alternative is *parsimony pressure*, which includes the size of an individual as a factor in the selection procedure. Parsimony pressure has often been *parametric*, meaning that it considers the actual values of size and fitness together in a parametric statistical model for selection. For example, one might compute fitness as a linear function of raw fitness and size (for example, [10]) or use them together in a nonlinear fashion [2]. The problem with parametric parsimony pressure is that it requires the experimenter to state that N units of size are worth M units of raw fitness, but fitness assessment procedures are usually ad-hoc. One of several undesirable effects is that late in the evolutionary run, when all individuals have approximately the same fitness, size begins to overwhelm the selection procedure.

Recent work has instead focused on so-called “nonparametric” parsimony pressure methods, where the size and raw fitness values are not combined during the selection process. Instead, selection methods compare sizes with sizes and fitnesses with fitnesses. One example of nonparametric parsimony pressure is *lexicographic parsimony pressure* [8]. Lexicographic parsimony pressure methods compare individuals first by fitness, and then by size only if there are ties in fitness. Lexicographic parsimony pressure fails in some problem domains (notably Symbolic Regression) where large swaths of individuals have completely unique fitness values, and so size is rarely considered. The method can be improved by *bucketing* the individuals by fitness. Here individuals are sorted by fitness into N buckets, and if two individuals being compared are from the same bucket, then size is used as the comparison. This has the effect of discretizing

[1] Some depth limiting variants: [5] retries crossover some N times until it generates a valid-depth child; and [6] dynamically expands the tree depth as fitter individuals are discovered.

the fitness into classes. Related nonparametric parsimony pressure methods were discussed in [9]. Specifically, in *double tournament,* selection is done by holding several tournaments by size, and the winners of those tournaments then go on to enter a final tournament by fitness. The alternative approach is *proportional tournament*, where an ordinary tournament selection occurs, but first a coin toss of probability p is held to determine whether the tournament will select based on size or on fitness. A related technique is the *Tarpeian* method, whereby a random subset of above-average-sized individuals are simply eliminated from the population [11].

There is also some literature on treating size as an alternative objective in a pareto-optimization scheme [12–14]. In a pareto scheme, individual A is said to *dominate* individual B if A is as good as B in all objectives (fitness, size) and is better than B in at least one objective. This family of methods use one of several multi-objective optimization algorithms to discover the "front" of solutions which are dominated by no one else. The literature has had mixed results, primarily because nondominated individuals tend to cluster near the front extremes of all-fitness or all-size, and a highly fit but poorly sized individual is undesirable, as is a tiny but highly unfit individual. [13] has tackled this problem successfully by introducing diversity preferences into the pareto system, thereby discouraging these clusters.

2 Three New Bloat Control Methods

In this paper we introduce three new bloat control methods and examine their performance. The first method, *Biased Multi-Objective Parsimony Pressure* (BMOPP) is another variation on the pareto-optimization theme which combines lexicographic ordering, pareto dominance, and a proportional tournament. This admittedly convoluted-sounding method is in fact quite easy to implement, is effective, and (unlike many multi-objective approaches) may easily be combined with common evolutionary computation algorithms.

The remaining two methods are unusual in that size is not considered in the selection process at all, but rather it plays a factor in punishing the individual in other parts of the evolutionary cycle. In *The Waiting Room*, newly-created individuals are not permitted to enter the population until they have sat in a queue (the "waiting room") for a period of time proportional to their size. This gives small individuals an advantage in that they may spread more rapidly through the population[2]. Finally, *Death by Size* works with those evolutionary methods

[2] We mention as an aside one bloat-control approach which is related to the waiting room but which we have *not* found to be effective: asynchronous island models. Island models connect parallel evolutionary processes via a network; every so often an evolutionary processes will "migrate" individuals to other processes over the network. In an asynchronous island model the processes are not synchronized by generation, but may run at their own pace. In theory processes with smaller individuals should run faster, and hence migrate more individuals to other processes, and those migrants will tend to be small. This gives small individuals a chance to spread throughout the

which must choose individuals to die and be replaced — in our experiments, we used steady state evolution. While individuals are selected for breeding using fitness, they are selected for death and replacement using size.

2.1 Biased Multi-Objective Parsimony Pressure

One of the problems with pareto-based multiobjective methods is that they consider any point along the front to be a valid candidate. Thus it is easy to generate individuals at one extreme of the Pareto front, namely individuals with small tree sizes but terrible fitness values. Usually, however, the experimenter is more interested in the other extreme: highly fit individuals. The idea behind the *Biased Multi-Objective Parsimony Pressure* (BMOPP) is to bias the search along the front towards the fitness end of the gamut.

BMOPP, like other pareto methods, uses fitness and size as its two objectives. At each generation, BMOPP places each individual into a *Pareto layer* as follows. First, individuals along the nondominated front of the two objectives are assigned to layer 1. Those individuals are then removed from consideration, and a new front is computed from among the remaining individuals. Individuals in that new front are assigned to layer 2 and removed from consideration, another new front is then computed, and so on.

After Pareto layers have been established, individuals are selected using a form of tournament selection which, with probability P, compares individuals based solely on their fitnesses, and with probability $1 - P$ compares them based on their respective Pareto layers (lower layers being preferred). Ties are broken using the alternative comparison (fitness or Pareto layer). If both individuals are identical in fitness and in layer, one individual is chosen at random. The particular value of P is of interest to us: as P increases, selection is more likely to be based on tree size and less likely to be based on fitness. If $P = 1$, BMOPP is exactly lexicographic tournament. If $P = 0$, BMOPP is entirely pareto-based.

2.2 The Waiting Room

In *The Waiting Room* newly-created individuals are punished for being large by being placed in a queue (the "waiting room") prior to entry into the population. Let the population size be N. At each generation, some RN newly-created individuals are evaluated, then added to the waiting room. $R > 1$, and so the waiting room will be larger than the final population size. Each individual in the waiting room is assigned a *queue value* equal to the individual's size. Next, the N children with the smallest queue values are removed from the waiting room and form the next generation. The remaining individuals have their queue values multiplied by a cut-down value A between 0 and 1. A prevents queue stagnation: eventually even giant individuals may have a chance to be introduced to the population.

network more rapidly. While we have not been able to get an asynchronous island model on its own to produce sufficiently parsimonious results, nonetheless we find the notion intriguing enough to be worth mentioning.

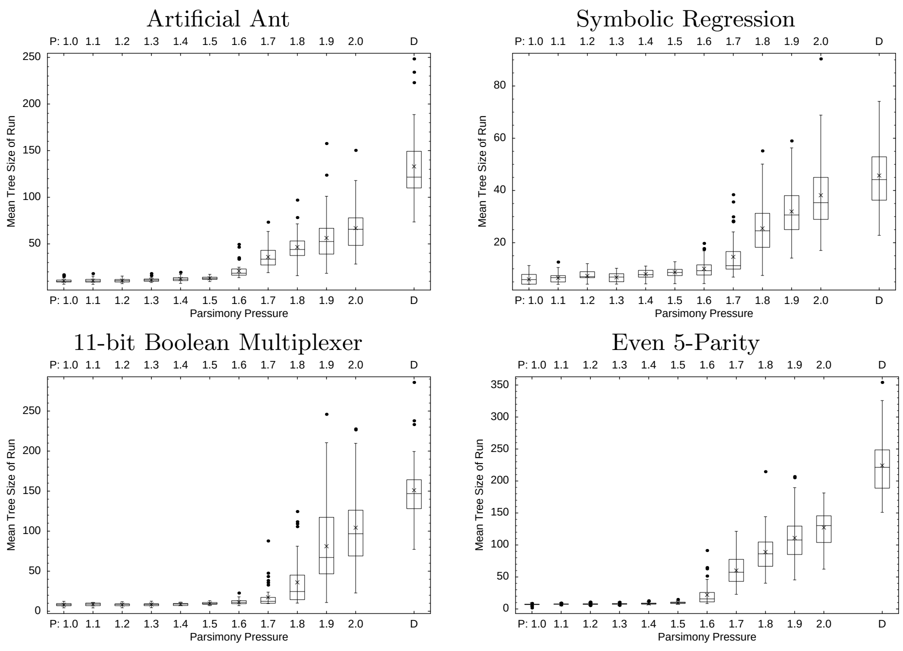

Fig. 1. Mean tree sizes for BMOPP in combination with depth limiting, as compared to depth limiting alone (labeled D). The size of the tournament selection which selects between lexicographic tournaments (layer, fitness) and (fitness, layer) is labeled P. The mean of each distribution is indicated with an $\times$.

The idea of the waiting room is that smaller individuals get to compete in the evolutionary process sooner, and thus spread (smaller) genetic material more rapidly. Our implementation of the waiting room allows it to grow without bound; but we imagine that real-world implementations would require some maximum size. Besides memory concerns, naive implementations of the waiting room impose significant computational complexity by cutting down individuals by A each time. We suggest using a binomial heap, and introducing new individuals to the queue by multiplying them by some increasingly large number (the inverse of A) rather than decreasing A for the existing individuals in the heap. This imposes at most an $O(\lg |\text{Waiting Room}|)$ cost per selection.

2.3 Death by Size

The two methods presented so far, and indeed all of our previously discussed methods, are designed for generational evolutionary computation. In contrast, *Death by Size* is intended for methods such as steady-state evolution which require a procedure for marking individuals for death. Death by Size is very simple: use fitness to select individuals for breeding, as usual, but when selecting an individual for death, select by size (preferring to kill larger individuals). In our

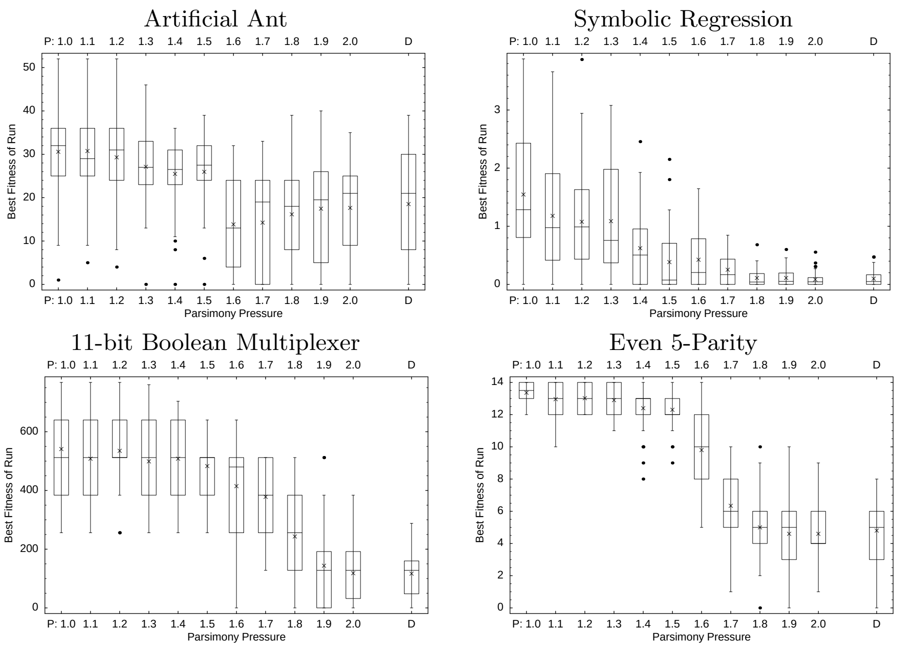

Fig. 2. Best fitnesses of run for BMOPP in combination with depth limiting, as compared compared to depth limiting alone (labeled *D*). The size of the tournament selection which selects between lexicographic tournaments (layer, fitness) and (fitness, layer) is labeled *P*. The mean of each distribution is indicated with an $\times$.

experiments we used steady-state evolution, with tournament selection for both selection procedures.

3 Experiments

The bloat-control technique most used in the literature is Koza-style depth limiting. Like most of the literature, we compare our technique against it, and maintain the traditional depth of 17 to stay relevant for comparison. As it turns out, depth limiting can be easily added to the parsimony techniques presented in this paper. As our focus is in creating efficient methods to control bloat, we decided to augment our parsimony techniques with depth limiting.

The experiments used population sizes of 1000, run for 50 generations. Similarly, the waiting room method was stopped after evaluating 50000 individuals. The runs did not stop when an ideal individual was found. Unless stated otherwise, generational GP runs used plain tournament selection of size 7. We chose four problem domains: Artificial Ant, 11-bit Boolean Multiplexer, Symbolic Regression, and Even-5 Parity. We followed the parameters specified in these four domains as set forth in [4], as well as its breeding, selection, and tree generation

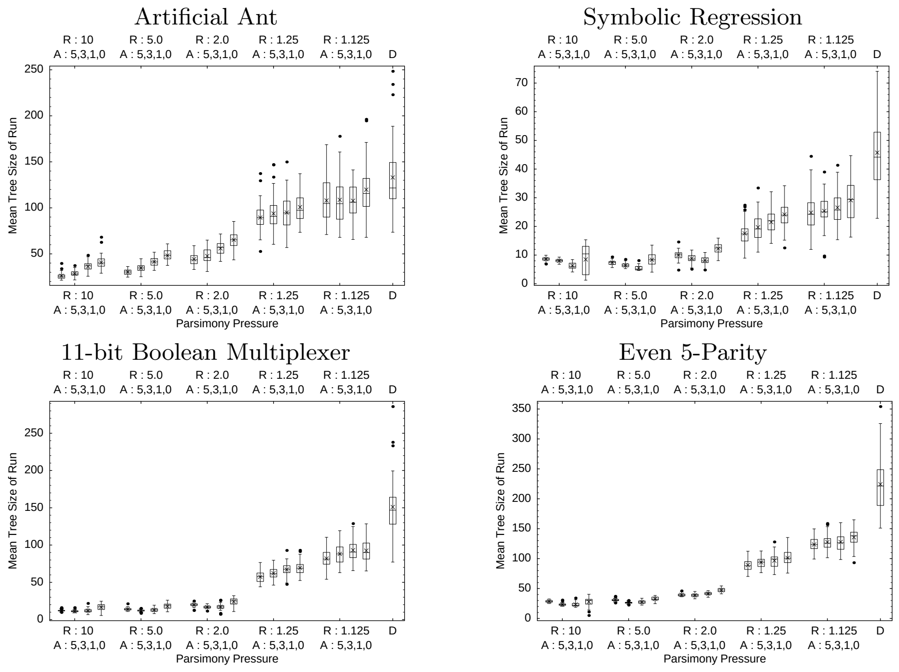

Fig. 3. Mean tree sizes for the waiting room parsimony pressure method in combination with depth limiting, as compared compared to depth limiting alone (labeled D). The ratio of waiting room to the population size is labeled R, and the aging rate is labeled A. The mean of each distribution is indicated with an $\times$.

parameters. Artificial Ant used the Santa Fe food trail. Symbolic Regression used the quartic polynomial fitness function and no ephemeral random constants. The evolutionary computation system used was ECJ [15]. ECJ augments standard tournament selection with a the addition of legal real-valued "tournament sizes" ranging from 1.0 to 2.0. For a tournament size P in this range, two individuals are selected, and the better is returned with probability $P/2$; with probability $1 - P/2$, a random individual is returned.

Our statistical analysis of the results used the Welch's two-sample test. In the Artificial Ant, 5-Bit Parity and 11-Bit Multiplexer domains, we compared results for each setting against the results of depth limiting alone. In the Symbolic Regression domain, we first computed ranks for the results of the two methods, and then used Welch's test directly on the ranks, in order to compensate for Symbolic Regression's non-normality in fitness. We used 95% confidence level for tests on best fitnesses of run, but used 99.98864% confidence level (derived from Bonferroni's inequality) for all tests on mean tree size. The reason for this was to be conservative when reporting better tree size at the same fitness level.

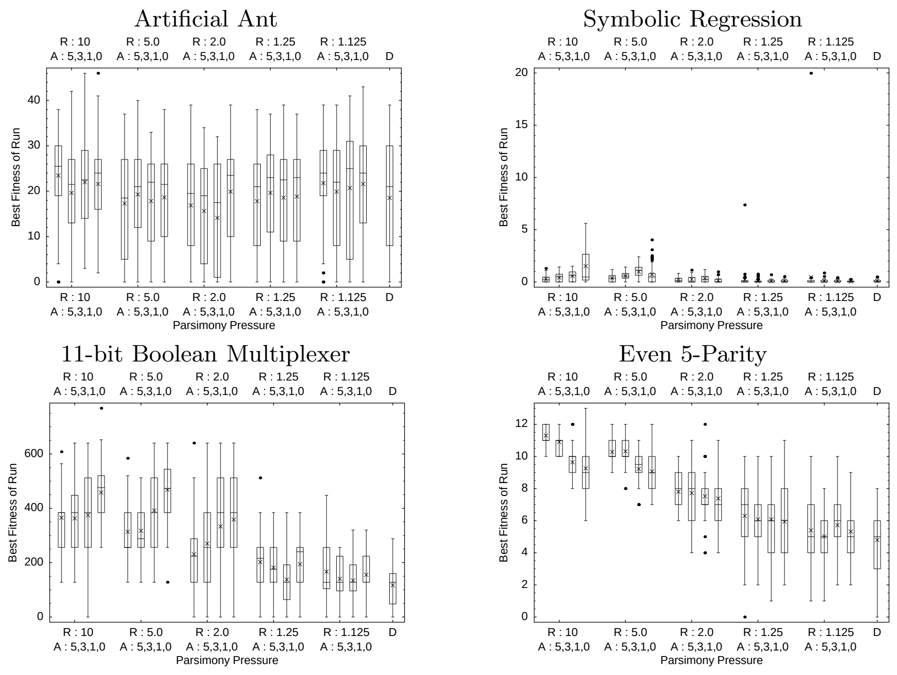

Fig. 4. Best fitnesses of run for the waiting room parsimony pressure method in combination with depth limiting, as compared compared to depth limiting alone (labeled *D*). The ratio of waiting room to the population size is labeled *R*, and the aging rate is labeled *A*. The mean of each distribution is indicated with an $\times$.

In all graphs, lower fitness is better, and the rightmost bar (labeled *D*) represents plain depth-limiting alone.

3.1 Biased Multi-Objective Parsimony Pressure

Figures 1 and 2 show the mean tree size and best fitness of run for the experiments on the BMOPP method in combination with depth limiting. The extremes of tournament size ($P = 1$ and $P = 2.0$) represent pure pareto multiobjective optimization and pure lexicographic ordering, respectively.

The aim is to achieve lower tree sizes while at least maintaining equivalent fitness to depth limiting alone. In the Artificial Ant domain, this occurred when P ranged from 1.7 to 2.0. For the Multiplexer domain: 1.9 and 2.0. For Parity: 1.8 to 2.0. For Symbolic Regression: 1.5, 1.8, and 1.9.

As expected, smaller values of P add more parsimony pressure to the search process: this significantly reduces the tree size, but usually this comes at the expense of fitness. Overall, the mild value of 1.9 for P significantly reduced bloat without affecting fitness in all problem domains in our experiments.

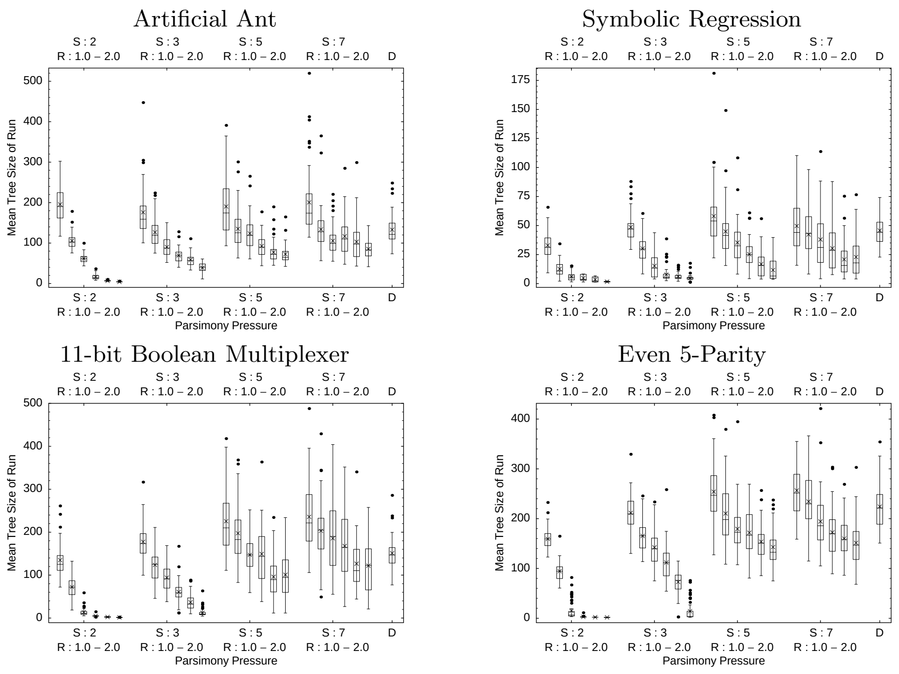

Fig. 5. Mean tree sizes for the death by size parsimony pressure method in combination with depth limiting, as compared compared to depth limiting alone (labeled D). The size of the selection tournament is labeled S, and the size of the de-selection tournament is labeled R. The mean of each distribution is indicated with an $\times$.

3.2 The Waiting Room

Figures 3 and 4 show the mean tree size and best fitness of run for the experiments on the waiting room method in combination with depth limiting. There are two parameters: the ratio R of the size of the waiting room relative to the population size; and the cut-down parameter A. Generally smaller values of R and larger values of A resulted in smaller tree sizes.

The results in the Artificial Ant domain show no degradation in the fitness of the individuals, but the the tree size is significantly reduced in all cases with the exception of (R=10, A=5) and (R=1.125, A=5, 3, 0). Interestingly, the mean tree size of run is reduced by as much as 5 times for (R=10, A=3). The Multiplexer domain is difficult for the waiting room algorithm. Only a few settings have similar fitness and better tree sizes than depth limiting: (R=1.25, A=1) and (R=1.125, A=3, 1). The results in the 5 Bit Parity domain are similar to the ones obtained in the Multiplexer domain: only (R=1.125, A=5, 3, 0) had similar fitness and smaller trees than depth limiting alone. Symbolic Regression yielded better results: (R=1.25, 1.125, A=5, 3, 1, 0) and (R=1, A=5, 3, 0).

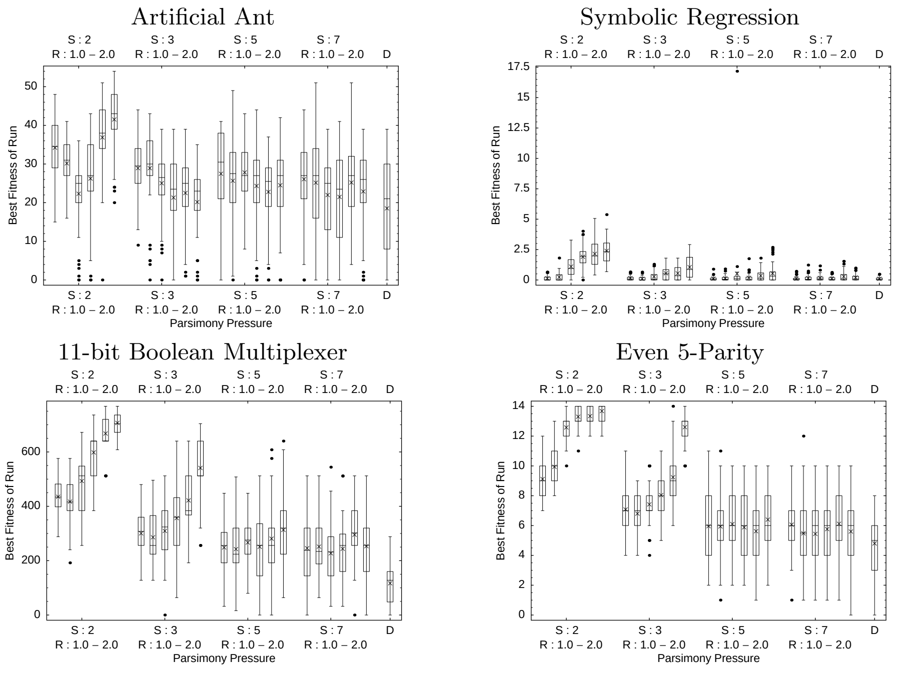

Fig. 6. Best fitnesses of run for the death by size parsimony pressure method in combination with depth limiting, as compared compared to depth limiting alone (labeled *D*). The size of the selection tournament is labeled *S*, and the size of the de-selection tournament is labeled *R*. The mean of each distribution is indicated with an $\times$.

Overall, it appears that a small pressure for parsimony significantly reduces the mean tree size of individuals, but does not significantly hinder the performance of the search process. This implies that the size of the waiting room will not increase substantially, which is also good news. The setting (R=1.125, A=3) appears to perform well across all four domains.

3.3 Death By Size

Figures 5 and 6 show the mean tree size and best fitness of run for the experiments on the death by size method in combination with depth limiting. We varied two parameters: S is the size of the tournament selection (based on fitness) to pick individuals to breed. R is the size of the tournament selection (based on size) for individuals to die and be replaced.

In the Artificial Ant domain, the following (S, R) tournament size settings resulted in similar fitnesses to depth limiting: (2, 1.4), (3, 1.6), (3, 1.8), (3, 2.0), (5, 1.8), (7, 1.4), (7, 1.6), and (7, 2.0). Of them, (7, 1.4) and (7, 1.6) resulted in similar tree sizes, while all others settings had a smaller mean tree size. In particular, tree size was halved by the (2, 1.4), (3, 1.6), (3, 1.8) and (5, 1.8) settings, and it was reduced by a factor of three when using (3, 2.0). In the 5-Bit Parity domain, the following settings had similar best fitness of run: (7, 1.2), (7, 1.4) and (7, 2.0). Of them, only (7, 2.0) yielded significantly smaller mean tree sizes. In the Symbolic Regression domain, the following settings had both similar best fitness of run and lower mean tree size: (2, 1.0), (2, 1.2), (3, 1.2), (3, 1.4), (5, 1.6), (5, 1.8), (7, 1.6), (7, 1.8) and (7, 2.0).

Unfortunately in the Multiplexer domain, *no* setting yielded both smaller tree sizes *and* equivalent fitness values as plain depth limiting. We believe that the Multiplexer problem domain provides a difficult testbed for steady-state approaches, and we plan to investigate this issue further.

As expected, higher values of R bias the search process more towards smaller individuals, while higher values of S bias towards higher fitness at the expense of parsimony. (7, 2.0) appears to be a good combination across all domains except for Multiplexer.

4 Conclusion

This paper introduced three new techniques for controlling bloat, each with different approaches to penalizing bloated individuals: Biased Multi-Objective Parsimony Pressure (BMOPP), the Waiting Room, and Death by Size. BMOPP uses fitness and tree size together to compute Pareto nondominated layers, which are later used in combination with fitness to compare individuals. The Waiting Room penalizes new, large children by delaying their entry into the evolutionary process. Death by Size uses a steady-state setting where bigger individuals are more likely to be replaced by newly created children.

We tested these three bloat control methods on four traditional GP benchmarks, and found that they were able to reduce the mean tree size of individuals evaluated without degradation in best fitness of run. Biased Multi-Objective Parsimony Pressure and the Waiting Room allow for settings that perform well across all four domains. The results of Death by Size suggest that the 11-Bit Multiplexer domain may be a particularly difficult testbed for steady-state approaches. However, we found settings that performed well in all other three domains, and we believe this method may be a useful tool due to its unusual application to steady-state evolution.

Our future work will provide comparative examinations of the many bloat control methods proposed in the literature. We also plan to investigate the applicability of these techniques to non-GP environments as well.

References

1. Smith, S.F.: A Learning System Based on Genetic Adaptive Algorithms. PhD thesis, Computer Science Department, University of Pittsburgh (1980)
2. Bassett, J.K., De Jong, K.A.: Evolving behaviors for cooperating agents. In: International Syposium on Methodologies for Intelligent Systems. (2000) 157–165
3. Luke, S.: 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)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
5. Martin, P., Poli, R.: Crossover operators for A hardware implementation of GP using FPGAs and Handel-C. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 845–852
6. Silva, S., Almeida, J.: Dynamic maximum tree depth. In: Genetic and Evolutionary Computation – GECCO-2003, Chicago, Springer-Verlag (2003) 1776–1787
7. Haynes, T.: Collective adaptation: The exchange of coding segments. Evolutionary Computation **6** (1998) 311–338
8. Luke, S., Panait, L.: Lexicographic parsimony pressure. In: GECCO-2002: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kauffman (2002) 829–836
9. Luke, S., Panait, L.: Fighting bloat with nonparametric parsimony pressure. In Merelo Guervós, J.J., Adamidis, P., Beyer, H.G., Fernández-Villacañas, J.L., Schwefel, H.P., eds.: Proceedings of the International Conference on Parallel Problem Solving from Nature (PPSN VII), Springer-Verlag (2002) 411–421
10. Burke, D.S., De Jong, K.A., Grefenstette, J.J., Ramsey, C.L., Wu, A.S.: Putting more genetics into genetic algorithms. Evolutionary Computation **6** (1998) 387–410
11. Poli, R.: A simple but theoretically-motivated method to control bloat in genetic programming. In: Genetic Programming, Proceedings of EuroGP'2003, Springer-Verlag (2003) 204–217
12. Bleuler, S., Brack, M., Thiele, L., Zitzler, E.: Multiobjective genetic programming: Reducing bloat using SPEA2. In: Proceedings of the 2001 Congress on Evolutionary Computation, IEEE Press (2001) 536–543
13. de Jong, E.D., Pollack, J.B.: Multi-objective methods for tree size control. Genetic Programming and Evolvable Machines **4** (2003) 211–233
14. Ekart, A., Nemeth, S.Z.: Selection based on the pareto nondomination criterion for controlling code growth in genetic programming. Genetic Programming and Evolvable Machines **2** (2001) 61–73
15. Luke, S. ECJ 10 : An Evolutionary Computation research system in Java. Available at http://www.cs.umd.edu/projects/plus/ec/ecj/ (2003)