Multi-Objective Methods for Tree Size Control

Edwin D. de Jong* (`edwin@cs.brandeis.edu`) and Jordan B. Pollack (`pollack@cs.brandeis.edu`)
Brandeis University
Department of Computer Science
Waltham, Massachusetts 02454 USA

Received September 18 2002; Revised February 6, 2003

Abstract. Variable length methods for evolutionary computation can lead to a progressive and mainly unnecessary growth of individuals, known as bloat. First, we propose to measure performance in genetic programming as a function of the number of nodes, rather than trees, that have been evaluated.

Evolutionary Multi-Objective Optimization (EMOO) constitutes a principled way to optimize both size and fitness and may provide parameterless size control. Reportedly, its use can also lead to minimization of size at the expense of fitness. We replicate this problem, and an empirical analysis suggests that multi-objective size control particularly requires diversity maintenance. Experiments support this explanation.

The multi-objective approach is compared to genetic programming without size control on the 11-multiplexer, 6-parity, and a symbolic regression problem. On all three test problems, the method greatly reduces bloat and significantly improves fitness as a function of computational expense. Using the FOCUS algorithm, multi-objective size control is combined with active pursuit of diversity, and hypothesized minimum-size solutions to 3-, 4- and 5-parity are found. The solutions thus found are furthermore easily interpretable. When combined with diversity maintenance, EMOO can provide an adequate and parameterless approach to size control in variable length evolution.

Keywords: Genetic programming, variable size representations, bloat, code growth, multi-objective optimization, Pareto optimality, interpretability.

1. Introduction

Evolutionary search with variable size representations often leads to unnecessary growth of individuals (Smith, 1980; Koza, 1992; Blickle & Thiele, 1994; McPhee & Miller, 1995; Langdon & Poli, 1998; Langdon, Soule, Poli, & Foster, 1999; Soule & Foster, 1999; Soule & Heckendorn, 2002; Banzhaf & Langdon, 2002). We consider this phenomenon, known as *bloat* or *code growth*, in the context of genetic programming (Cramer, 1985; Koza, 1992, 1994; Banzhaf, Nordin, Keller, & Francone, 1998; Langdon & Poli, 2002), and will assume that individuals are represented as trees.

---

*Current address: Vrije Universiteit Amsterdam, AI Dept. E-mail: edj@cs.vu.nl

The accumulative increases in size that bloat produces are problematic for several reasons. First, larger trees are undesirable as *solutions*, not merely because they are less convenient to handle and take more time to execute, but more so because they have been reported to generalize less well than compact trees (Kinnear, Jr., 1993; Rosca, 1996). Second, bloat may affect the ability of the search to find individuals of increasing fitness; if trees largely consist of non-functional parts, then crossover is more likely to select and combine elements from degraded genetic material[1]. Third, and most importantly, when no measures are taken against bloat, it can increasingly slow down the search process and thereby effectively limit the total number of individuals that can be processed (Banzhaf & Langdon, 2002). This means that problems requiring extensive search processes cannot be addressed. Thus, adequate methods for tree size control are essential in extending the range of problems that variable size evolution may address.

The most common technique to prevent unlimited growth is simply to disallow large size individuals by choosing a size threshold. This measure avoids producing trees of unlimited size, but until this size threshold is reached, bloat will occur unsuppressed. The genetic material that will form the basis of further search may therefore largely consist of material that was produced by bloat. Furthermore, the method requires a parameter; this introduces inductive bias, as it codes human expectation for the code size appropriate to a particular problem.

Another way to approach bloat is to start with small individuals and consider larger individuals systematically, so that a small solution will be found if it exists. An example of this approach is ADATE (Olsson, 1995), which combines an iterative deepening search method with the use of heuristics. For genetic programming however, smaller solutions cannot be assumed to have been considered. Thus, to reduce the incorporation of excessive genetic material in the search, a trade-off between fitness and size must be found.

A possible approach to trade off fitness and size is to use a function that combines fitness and size to yield a single measure of quality. However, the space of weighting functions is very large. To reduce the problem of choosing an appropriate function, the weighting is often restricted to linear functions. The difficulty of finding appropriate weights for such functions (Soule & Foster, 1999) raises the question of whether appropriate linear weighting functions exist for this purpose.

An alternative to choosing a fixed weighting is to base the weighting on actual properties of the evolutionary process. This idea is used

---

[1] Nonetheless, such material could also play a useful role in evolution (Angeline, 1994; Nordin, Francone, & Banzhaf, 1996).

by Zhang and Mühlenbein (1995) in Adaptive Parsimony Pressure, where the size penalty is based among others on the sizes of the best individuals in recent generations. Luke and Panait (2002) avoid using a weighting of fitness and size by separating the two objectives, and selecting for size or fitness in different tournaments using methods called proportional and double tournament. Interestingly, these methods point in the direction of multi-objective optimization, as they resemble early approaches to multi-objective optimization by Schaffer (1985) and Fourman (1985); see also discussion in (Fonseca & Fleming, 1995). While not using a weighting, the approach does require parameters that trade off the relative importance of the two objectives. Our goal here will be to avoid such parameters too, and strive towards parameterless size control.

Rather than compressing size and fitness into a single measure of quality as done in weighting approaches, methods for *Evolutionary Multi-Objective Optimization* (EMOO) treat all objectives separately, and avoid comparing different objectives (Fonseca & Fleming, 1995; Coello, 2000; Deb, 2001). This approach to bloat has been taken by several authors (Langdon, 1996; Rodríguez-Vázquez, Fonseca, & Fleming, 1997; Zitzler & Thiele, 1999; Ekárt & Németh, 2001; De Jong, Watson, & Pollack, 2001).

The promise of multi-objective methods for size control is illustrated by several successful applications (Langdon, 1996; Zitzler & Thiele, 1999; De Jong et al., 2001). However, the direct use of size and fitness as objectives can result in a harmful convergence to small size individuals, as found by several authors; see e.g. (Langdon & Nordin, 2000; Ekárt & Németh, 2001). Our aim here is to understand why EMOO algorithms with this particular choice of objectives are prone to such problematic convergence. Modifications to the multi-objective method were suggested in (Ekárt & Németh, 2001), but these methods again required a parameter that regulates the trade-off between fitness and size, thus introducing problem-dependent parameters. Our goal here is to understand the cause of the failure, as a step towards the desirable aim of parameterless size control.

To determine why an evolutionary multi-objective approach to bloat can be problematic, we investigate the behavior of a basic method for evolutionary multi-objective optimization. A straightforward application of this method to the objectives of fitness and size is found to strongly focus on the size objective and neglect the fitness objective, confirming earlier reports. The problem is analyzed, leading to the conclusion that diversity maintenance is required. To test this hypothesis, we make a minimal modification to the algorithm to maintain some existing diversity, and find that it avoids the over-representation of small individuals. This approach is then applied to three test problems. Finally, we describe the FOCUS algorithm, which actively promotes diversity by making it an additional objective of the search. Results with using this algorithm to find compact trees are presented.

The structure of this article is as follows. In section 2, causes for bloat that have been put forward are discussed. Section 3 shows how bloat can rapidly bring down the number of individuals that can be processed within a given amount of node evaluations. Section 4 verifies experimentally that the use of a basic multi-objective method can lead to a premature convergence to small trees, and an analysis of this phenomenon suggests that diversity maintenance is required. This hypothesis is tested in section 5, and the resulting method is applied to test problems in section 6. Finally, results obtained with active promotion of diversity using the FOCUS method are reported in section 7, and section 8 concludes.

2. Explanations of Bloat

Several causes of bloat have been suggested in the literature. First, the offspring of large trees can be favored by selection because nonfunctional code can play a protective role against crossover (Angeline, 1994; Blickle & Thiele, 1994; McPhee & Miller, 1995; Nordin & Banzhaf, 1995). This phenomenon may occur when trees are disarranged (for example by mutation or crossover) and subsequently selected based on fitness; whenever a tree is created that is as fit as other population members but less likely to be disrupted due to larger size[2], it will be more likely to be selected in the absence of other factors, and thus increase the average size of individuals. Considering this form of bloat, Langdon (1998) also found that a size neutral mutation operator is more likely to produce children of equal fitness for larger trees.

Second, trees may grow larger due to crossover and subsequent selection due to *removal bias* (Soule, 1998). Soule defines inviable code as code that cannot affect the fitness or function of the tree, whatever code it is replaced with.

Briefly, removal of a subtree is expected to have less impact on fitness if the subtree is small, while for the replacing subtree no such relation exists. Thus, trees are expected to grow larger. Since the increase is proportional to the current size of the trees, this mechanism can potentially produce exponential growth.

---

[2] Note that for mutation, this is not the case when using a fixed per node probability of mutation.

Third, code growth is expected to occur in general for progressive search techniques if fitness based selection is used (Langdon & Poli, 1998; Banzhaf & Langdon, 2002), since variable length search problems typically have more long representations for a given solution than short ones. Langdon and Poli observed that crossover with fitness-based selection led to bloat, whereas crossover with random selection did not. This also depends on the connectivity of the fitness landscape.

Nordin and Banzhaf (1995) observed that the length of the *effective* part of programs decreases over time. However, the total length of the programs in the experiments also increased rapidly, hence it may be concluded that bloat in these experiments was mainly due to the growth of ineffective code.

3. Bloat and the Speed of Search

In the following, the influence of bloat on the performance of the search process is considered in a simple genetic programming experiment. The algorithm in this experiment is a basic genetic programming method, described in section 6. The problem is the even 6-parity problem, described in section 4.1.

Figure 1 shows that tree size increases progressively as a function of the number of generations. The computational complexity of evaluation is typically (at least) linear in the number nodes of a tree. Together, this suggests that without any measure against bloat, sustained evolution is infeasible because the cost of evaluation increases dramatically with the number of generations. Depending on implementation, the time required for generation and selection will typically also increase with tree size, and thus further slow down the search.

The above point has important implications for the way performance can be measured in genetic programming. In general, the purpose of measuring an algorithm's performance is to determine its efficiency in finding solutions or approximations thereof. To avoid the influence of machine-specific details, the amount of computational effort is typically measured as a function of the number of generations or, equivalently, the number of fitness evaluations. However, as figure 1 showed, bloat can greatly influence the amount of function or node evaluations per fitness evaluation. Therefore, in variable-size evolution, the number of fitness evaluations spent in achieving a certain performance is no adequate measure of computational effort spent.

To address this issue, we measure computational effort as a function of node evaluations instead of tree evaluations. While this method has been used previously by several authors (Poli, 1997; Banzhaf, Ban-

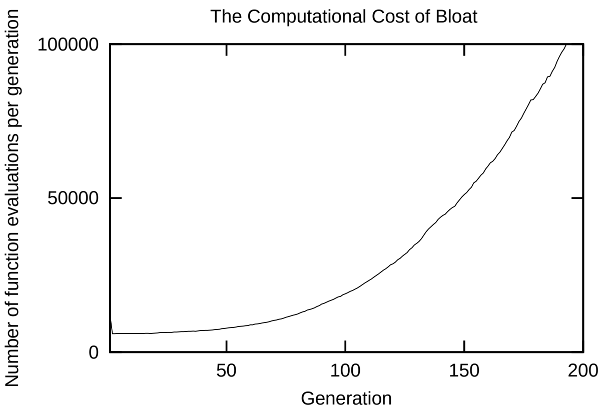

Figure 1. The number of nodes evaluated per generation as a function of the generation number when no measures are taken against bloat, averaged over 100 runs (see text). The rapid increase in computational costs per generation imply that an adequate solution to bloat is required to enable sustained progress.

scherus, & Dittrich, 1998), its use has not yet become widespread. We propose that it should be used as standard practice when evaluating methods for variable length evolution. By measuring the amount of computation in this more fine-grained manner, a far better approximation of the computational effort is obtained. The proposed method is easy to implement, and does not cost much computation time when the number of node evaluations for a tree is simply approximated by its number of nodes. If it is important to obtain yet higher precision, the number of nodes actually evaluated can be counted at evaluation time; if each node is evaluated at least once, costing at least $O(1)$ per node, even this does not affect the computational complexity of evaluation. Since the number of nodes only has to be computed once per tree and will often be a sufficiently accurate indicator of the cost of evaluation, we will use this measure as the indicator of the computational effort spent.

4. Premature Convergence of the Multi-Objective Approach to Small Trees

Multi-objective methods can converge to populations of small individuals of low fitness, meaning convergence took place before high fitness individuals could be found. In this section, we investigate how such premature convergence to small trees can occur.

Current methods for Evolutionary Multi-Objective Optimization (EMOO) are based on Pareto-dominance. For an introduction, see e.g. (Fonseca & Fleming, 1995; Coello, 2000; Deb, 2001). EMOO assumes that objectives are essentially incomparable, and that comparison of different objectives is to be avoided. Thus, an individual is only preferred over another if it is at least as good considering all objectives. By further requiring that an individual cannot be preferred over one with identical objective values, we arrive at the principle of Pareto-dominance.

Let individual x have values x_i for the n objectives, and let individual y have objective values y_i. Then x dominates y if and only if:

$$\forall i \in [1..n] : x_i \geq y_i \quad \wedge \quad \exists i \in [1..n] : x_i > y_i$$

Methods for evolutionary multi-objective optimization strive to approximate the Pareto-optimal set, containing all non-dominated solutions, i.e. individuals that are not dominated by any other individuals. Thus, when functioning properly, a multi-objective method using fitness and size as objectives produces a tradeoff front of individuals ranging from small but unfit individuals to highly fit but large individuals, with many intermediate combinations in between. For each of the individuals on this front, neither fitness nor size can improve without decreasing the other objective.

4.1. ALGORITHM DESCRIPTION AND TEST PROBLEM

We aim to investigate the behavior of multi-objective methods when using fitness and size as objectives. Several sophisticated methods for evolutionary multi-objective optimization exist, e.g. (Zitzler & Thiele, 1999; Corne, Jerram, Knowles, & Oates, 2001; Deb, Agrawal, Pratab, & Meyarivan, 2000). Our aim here however is to understand what can go wrong when applying multi-objective methods to bloat. We will therefore use a basic multi-objective algorithm, based on the algorithm described by Fonseca and Fleming (1993). We follow Soule and Foster (1999) in other properties of the experiment, as described below; see also table I.

The initial population consists of random trees. The size of a random trees is selected by drawing its number of internal nodes randomly from a uniform distribution between 1 and 10. The tree is generated by starting with a single root node and changing randomly chosen leaf nodes into nodes with children until the specified number of internal nodes is reached.

Table I. Properties of the experiment.

Problem	Even 6-parity
Terminal set	The 6 input variables $x0..x5$
Function set	AND, OR, NAND, and XOR
Fitness	Number input cases handled correctly (out of $2^6 = 64$)
Selection	Stochastic remainder without replacement
Cycle	Generational
Population size	500
Initial Population	Random trees (see text)
Generation of new individuals	66.6% by crossover, 33.4% by copying
Number of runs per experiment	100
Termination	1,000,000 node evaluations

Starting from an initial population, we create a new generation using crossover and copying on randomly selected parents, and combine all individuals into a single set. Crossover randomly selects a node in each of two trees and exchanges the subtrees rooted at these nodes. The number of individuals by which each individual x_i is dominated, the domination number $no_{dom}(x_i)$, is determined. Individuals are then sorted according to their no_{dom} value. Next, the relative probabilities of selection are assigned to individuals according to their index in the sorted list, scaled between zero and one. For groups of individuals that have equal no_{dom}, the resulting values are then averaged, so that they will have equal probability of selection. To obtain the next population, we use stochastic remainder selection without replacement (Brindle, 1981). This calculates the desired number of copies for each individual based on its normalized rank, assigns the whole parts of these numbers, and uses the fractional parts as the probability that an additional copy of the individual is added to fill up the next population to the required size, where each individual is selected at most once.

As a test problem in this experiment, we use the even 6-parity problem with the operators `AND`, `OR`, `NAND`, and `XOR`. Later experiments will employ a more difficult version in which `XOR` is replaced by `NOR`.

Solutions to an n-parity problem receive a sequence of n bits as input and return *true* (1) if and only if the number of ones in the sequence is even. The problem is called *even* parity to distinguish it from the complementary odd n-parity problem, which returns the inverse answers. The fitness objective is the number of correct cases; the size objective is minus the number of nodes.

4.2. EXPERIMENTAL ANALYSIS OF CONVERGENCE TO SMALL TREES

We investigate the plain multi-objective algorithm described in the previous section, using size and fitness as objectives. The average evolution of populations is shown in figure 2. Starting from a randomly initialized population, the method is ineffective at improving fitness. It drives the population towards trees of minimum size, i.e. single nodes, and consequently the average fitness converges to the baseline fitness of 32.

The left graph shows the distribution of tree sizes in the population for the first ten generations, averaged over 100 runs. At generation zero, the tree sizes of the initial population are uniformly distributed over the odd numbers between 3 and 21 as a result of the initialization of the population[3]. Within the first few generations, this distribution rapidly shifts toward smaller trees. From generation 8 on, the vast majority of the trees contain just a single node. The graph on the right shows the distribution of fitness for the same experiment. Although initially some individuals with higher than baseline fitness exist, and there is even a slight increase in their frequency during the first three generations, they are then driven out of the population by the smaller individuals, causing fitness to converge to the baseline fitness of 32. Since two single-node parents can only produce offspring of single nodes, this convergence is permanent.

These findings show that the plain multi-objective method with fitness and size objectives is not functioning properly, as it does not lead to different combinations of fitness and size. In the following, the problematic convergence to small individuals is analyzed to find the underlying cause, so that it may be addressed.

4.3. EXPLANATION OF THE SHIFT TOWARDS SMALL INDIVIDUALS

We will now examine the shift towards small individuals found for the plain application of the multi-objective approach to tree size control. First, selection in multi-objective methods favors non-dominated individuals over dominated ones. For a discrete fitness measure such as that used here, this implies that for each fitness represented in the

---

[3] Since all operators are binary, trees cannot have an even number of nodes.

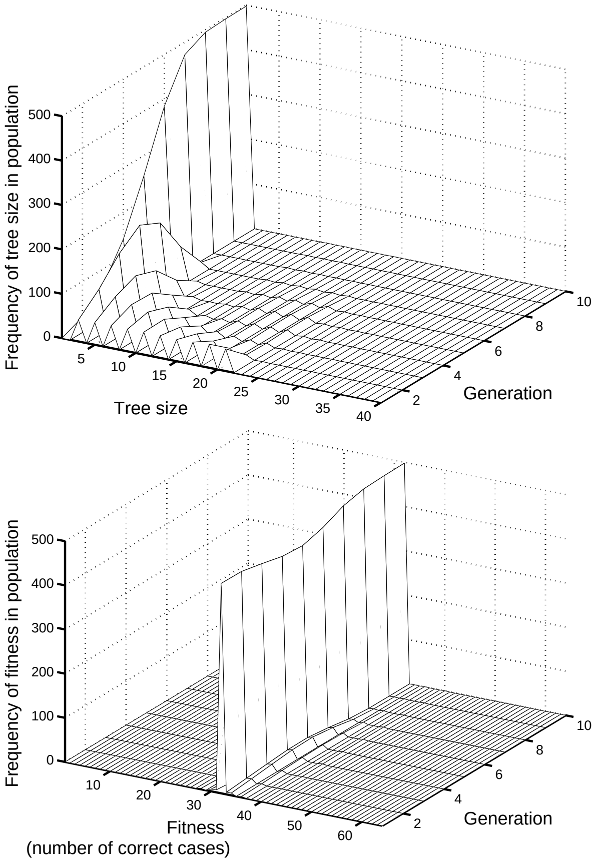

Figure 2. The distribution of tree size (top) and fitness (bottom) using a basic multi-objective method (see text). Within the first ten generations, the population converges to individuals of minimum size and baseline fitness.

population, the smallest individuals with that fitness are favored over larger individuals with the fitness. All such favored individuals are non-dominated, hence they all have equal probability of being maintained in the population after selection. This leads to a front of different combinations of fitness and size. In figure 2, this is visible as the mildly multi-modal distribution in generations 4 through 7.

Selection A possible explanation for the subsequent disappearance of the non-minimum size individuals is that due to the variance of selection (Mahfoud, 1995), also known as genetic drift, the uneven distribution eventually leads the small individuals to take over the population; from figure 2, we know that the distribution of the initial population is indeed highly skewed towards baseline fitness, and the minimum size for individuals of this fitness is 1.

To test whether this explanation holds, we focus on the effect of selection by disregarding the generation of new individuals. This is done by using no crossover but only copying to produce the new generation, so that the expected distribution of the new generation equals that of the current population. If the explanation is correct, it would be expected that after some period of maintaining a non-dominated front, individuals with fitness 32 and size three will come to dominate the population. However, this is not what was observed. In the standard setup shown in figure 2, 99.56% percent of the population is of minimum size after ten generations, and this converged to 100% by generation 25. In the control experiment, 96% are of the smallest possible size[4], and this figure continued to fluctuate between 95.4 and 96.2% up to generation 100. Thus, we can conclude that the variance of selection is not sufficient to account for the disappearance of non-dominated individuals of non-minimum size.

Generation The effects of selection were found insufficient to explain the disappearance of non-dominated individuals of non-minimum size. To investigate the problematic disappearance, we focus on non-dominated individuals and study their distribution in a new generation compared to the current population. If newly generated non-dominated individuals are smaller on average than the existing non-dominated individuals, then the distribution is expected to shift even without any effects due to selection, i.e. if all and only non-dominated individuals are maintained. We investigate this by using an algorithm that simply maintains all non-dominated individuals. Hence, the population size is variable. 500 new individuals are generated at each generation, using crossover in 66.6% or 100% of the cases and copying in the remaining cases; no mutation is used.

The graph in figure 3 shows the relation between the size of newly generated non-dominated individuals and the existing non-dominated individuals. When all generated trees are considered, this relation closely approximates the identity function $y = x$. This conforms with expectation, since both crossover and mutation are size neutral. However, considering non-dominated trees only, newly generated trees are considerably smaller on average than existing ones. Typically, all of the non-dominated trees have equal chances in selection. Once the population cannot accommodate all non-dominated individuals anymore, even minimal variance selection will therefore shift the distribution of

---

[4] In the control experiment, the smallest possible size is three, as no new individuals are generated; this does not affect matters, since it does not make it easier for larger individuals to be non-dominated.

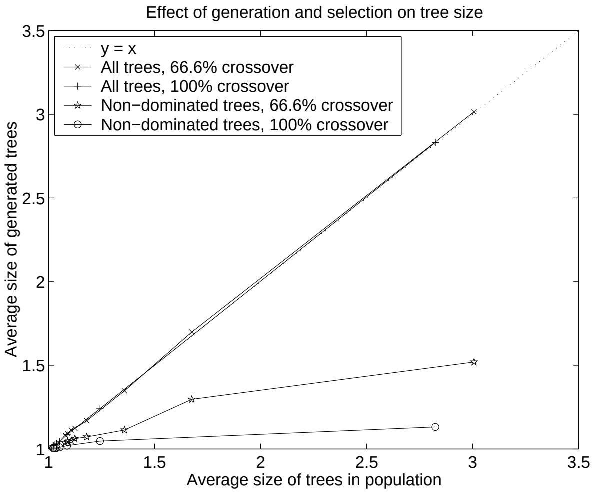

Figure 3. Effect of generation and selection on the size of trees. On average, crossover and copying have no effect on tree size, as seen by the two lines in the graph approximating the diagonal y = x. However, of these generated trees, the non-dominated ones are substantially smaller than average for both the 100% crossover and the 66.6% crossover experiment.

tree sizes downwards. Thus, the effects of generation are sufficient to explain the convergence to small trees.

We believe that the above findings may explain other unsatisfactory results where a basic multi-objective approach was used with fitness and size as objectives. Small size is typically easier to achieve than high fitness, as seen from the high frequency of small trees among non-dominated individuals. Thus, by searching for non-dominated individuals, the population shifts towards smaller individuals. The resulting change in the size distribution of the population causes newly generated trees to be smaller, reinforcing this effect. This leads to a positive feedback loop that rapidly brings down the size of the trees. The effect can rule out the future generation of larger individuals; crossover can generate trees of twice the size of the largest individual, but this may not be sufficient to reach a fitness increase that would warrant a place in the population. Thus, the use of a size objective can lead to degenerate populations.

Apart from the evaluation advantage for small trees in the current multi-objective setting, theoretical analysis of linear variable length evolution has pointed to a bias favoring small trees for crossover in flat fitness landscapes (Poli & McPhee, 2001).

In the experiments with the basic multi-objective approach to bloat reported at the beginning of this section (see figure 2), every run converged within 25 generations to a population in which the largest individual consists of a single node. We conclude that if multi-objective methods are to be used for size control, special care must be taken to maintain diverse populations. In the following section, we will test this hypothesis by employing a minimally complex diversity maintenance method.

5. Diversity Maintenance

The previous analysis showed that many of the non-dominated individuals generated were of small size and baseline fitness. This suggests that the multi-objective approach to bloat requires diversity maintenance. To test this hypothesis, we will simply reduce the number of individuals with identical objective values, i.e. identical size and fitness. The motivation for this is that individuals with identical objective values are generally less likely to contribute to population diversity than individuals with different objective values.

A reduction of the number of individuals in a location in objective space can be achieved by using a modification of the dominance relation that discounts the ranks of additional individuals in a location. The result of this measure is that additional individuals in a location will have lower probabilities of being maintained. Thus, a larger number of individuals in one location is compensated by smaller probabilities, so that overall the expected number of individuals will be reduced, and a spread of the population over different locations is maintained.

The method described above can be implemented within a dominance based selection scheme as follows. For each location on the front, the first individual x_0 occupying it has a domination number $no_{dom}(x_0) = 0$, i.e. it is not dominated by any individuals. The second individual x_1 is only dominated by x_0, and hence $no_{dom}(x_1) = 1$, and so forth. Thus, an individual at some location of the trade-off front is only dominated by the individuals that precede it: $no_{dom}(x_i) = i$. The order of individuals is arbitrary, and determined by the position in the list of individuals that constitute the population. Under this scheme, any new individual in another non-dominated location will be preferred over the second individual in an existing location. This means that selection will give priority to selecting at least one individual from each location on the front. The next level of priority is assigned to a second individual from each location, then a third, continuing in this manner. This balancing scheme establishes a uniform distribution over the trade-off front when possible.

5.1. EXPERIMENTAL RESULTS WITH DIVERSITY MAINTENANCE

Figure 2 in section 4.2 showed that using the standard multi-objective method, the population quickly converged to trees of minimum size and baseline fitness. Figure 4 shows the results of the front balancing method described in the previous section on the same problem. The use of diversity maintenance adequately addresses the problem. Rather than converging to a single size and fitness, the population spreads over various tree sizes without resulting in bloat, and finds trees of different fitnesses, including the maximal fitness of 64.

While the 6-parity problem with the `AND`, `OR`, `NAND`, and `XOR` operators has been described as unlikely to be addressed successfully by simple genetic programming (Soule & Foster, 1999), these results show that basic genetic programming (with size control) is able to solve the problem. We note that the choice of operators is of great influence on the difficulty of the n-parity problems. A more difficult version is obtained by not using the `XOR` operator, and using the `NOR` operator instead. While the `XOR` version of the 6-parity problem can be solved by trees of 13 nodes, the smallest solution to the `NOR` version of the problem that we are aware of, and which will be described, requires 79 nodes. The following section will report on experiments with this more difficult version of n-parity lacking the `XOR` operator, and with two other test problems.

Several features of the multi-objective approach to tree size control are worth mentioning. The distribution of tree sizes quickly spreads, but is focused around trees of appropriate size, i.e. the extremes of bloat and problematic convergence to small trees are both avoided. Since no parameters that regulate the trade-off between fitness and size are used, this size is determined by the problem, thus providing an adaptive form of tree size control. As the runs progress, higher fitness individuals are found; this is seen as the rightward shift of the fitness distribution. As higher fitness individuals are found, lower fitness individuals continue to be represented. Thus, as intended, the multi-objective method maintains a front of solutions rather than striving towards a single high fitness individual.

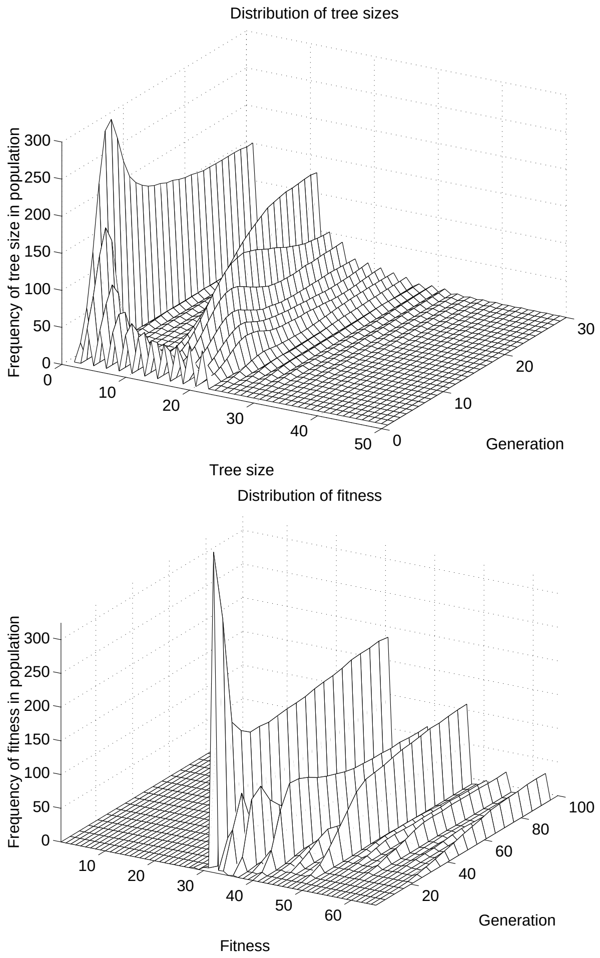

Figure 4. Distribution of tree size (top) and fitness (bottom) in the population for the multi-objective method using front balancing. Even though at each generation more small and unfit trees than large and fit trees are produced, the method maintains individuals along the front of fitness and size.

6. Comparison on Three Test Problems

In the previous section, it was seen that problematic results with a basic multi-objective method were resolved by adding a simple mechanism for diversity maintenance. This led to balanced populations, adequate size control, and higher fitness values. In the following, we will test the resulting approach on three test problems from the literature: 6-parity, the 11- multiplexer, and symbolic regression, see (Koza, 1992, 1994). The multi-objective method that will be used is the method described in section 4.1, except that here we simply maintain at most one individual per location rather than discounting multiple individuals in a location. As a comparison method, we use basic genetic programming without size control. This method is identical to the multi-objective method, except that evaluation employs the fitness objective only, rather than fitness and size. Thus, it uses rank-based selection, where ranks are determined by fitness. Unless otherwise noted, the properties of the methods are those in table I. For both methods and all three problems, new individuals are always generated by crossover, followed by mutation with a probability of 0.1. All experiments consist of 100 runs of 1,000,000 node evaluations each.

For each combination of method and problem, the population size has been selected by trying out the sizes 100, 250, and 500 for ten runs each, and selecting the population size that maximized the average best fitness at the end of the runs. For the basic genetic programming method, this resulted in population size 250 for each of the three problems. For the multi-objective method, it resulted in size 250 for symbolic regression and size 100 for the remaining two problems.

For the 6-parity problem, the operators `AND, OR, NAND,` and `NOR` are used. This makes the problem considerably more difficult compared to the version that includes `XOR`. For symbolic regression, the function $x^6 - 2x^4 + x^2$ (Koza, 1994) is used on 20 test points chosen randomly from [-1, 1]. The function set consists of `+, -, /, *, sin, cos, exp, log`, and the terminal set contains the input variable (x) and a random constant. For the 11-multiplexer, the function set is `AND, OR, NOT, IF`, and 3 of the 11 input variables are address bits.

Figures 5-7 show the development of fitness and tree size for the genetic programming method without size control and the multi-objective method using size and fitness. For 6-parity and the 11-multiplexer, fitness is to be maximized, while for symbolic regression the fitness represents an error that is to be minimized. Tree size graphs are plotted as a function of node generations to allow for comparison with the corresponding fitness graphs.

Concerning tree size, basic genetic programming exhibits bloat on all three test problems; it reaches sizes well beyond what is required for correct solutions, thus allowing only a limited number of tree evaluations for the given amount of node evaluations. The growth of tree sizes appears to level off over time, but this is a result of viewing tree size in terms of the number of node evaluations; as trees grow larger, the number of generation and selection events per horizontal unit in the graph decreases.

In contrast with the basic genetic programming method, the multi-objective approach successfully controls tree size on all three problems. It does so without making use of parameters that regulate the trade-off between fitness and size. This indicates that multi-objective optimization could be an adequate method for controlling tree size.

A further requirement for a successful size control method is that it should not have a detrimental effect on fitness. As figures 5-7 (right) show, this requirement is more than satisfied; the multi-objective approach for size control achieves better fitness for the given amount of computation on all three test problems. In summary, on all three test problems, the multi-objective approach not only provides adequate size control, but also improves the efficiency of the search.

To test the statistical significance of these results, we have performed a Mann-Whitney test, also known as the Wilcoxon rank sum test, see e.g. (Conover, 1980). This test has the advantage of not requiring assumptions about the distribution, such as normality. It is applied to the value (tree size or best fitness) in the last population of each run of an experiment[5]. The tree size is significantly lower for the multi-objective method on all three problems at a significance level of $\alpha = 0.0001$. The improved fitness for the multi-objective method is significant for $\alpha = 0.0005$ in the 6-parity experiment, and for $\alpha = 0.0001$ in the symbolic regression and 11-multiplexer experiments. Thus, all differences are statistically significant.

7. Reducing tree size: active promotion of diversity

If successful, methods for size control may not merely reduce bloat, but provide a mechanism to find compact solutions. Indeed, reducing unnecessary elements is often a goal in itself. This is the case for instance in data mining and knowledge discovery, where the interpretability of solutions is an important issue. Furthermore, in engineering design, reducing the amount of material used can improve the quality and lower the cost of a design.

In this final results section, we will therefore consider to what extent the multi-objective approach to size control is able to reduce unnecessary material and thereby identify compact solutions. We report results with the FOCUS algorithm. This algorithm actively promotes diversity, rather than simply maintaining some of the existing diversity. This is achieved by making diversity an additional objective.

---

[5] Technically, the Mann-Whitney test determines whether $P(X < Y) \leq \frac{1}{2}$. With the additional assumption that a difference between the populations must be in the location of the distribution, it may be concluded that $E(X) \geq E(Y)$.

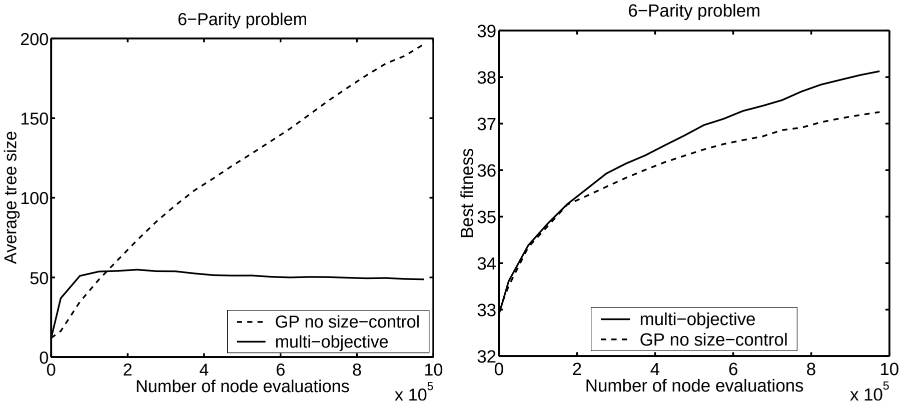

Figure 5. Development of tree size and fitness for the 6-parity problem, averaged over 100 runs (see text).

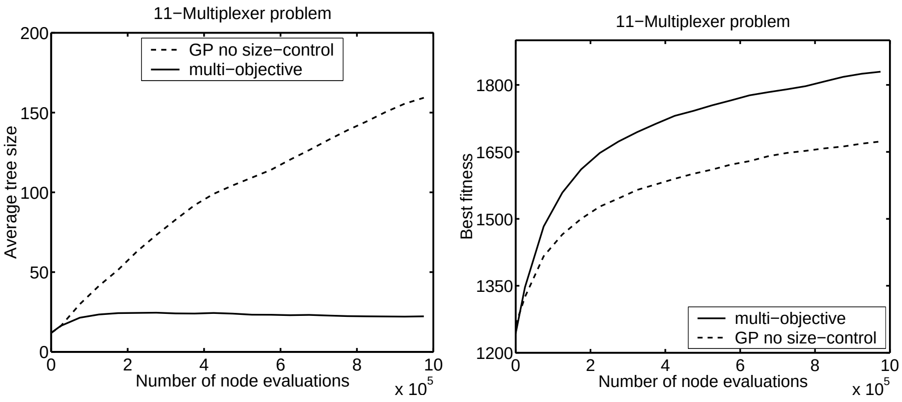

Figure 6. Development of tree size and fitness for the 11-multiplexer problem.

7.1. Minimum Size of n-parity Solutions

The problem we study is the even n-parity problem using operators `AND`, `OR`, `NAND`, and `NOR`. We first consider how many nodes are required for a correct solution. To this end, we describe a procedure that specifies correct solutions to even n-parity. The construction provides an upper bound for this minimum size. We have not been able to prove or disprove whether this bound is tight, or to find bounds for this or related problems (not allowing repeated use of outputs) in the literature.

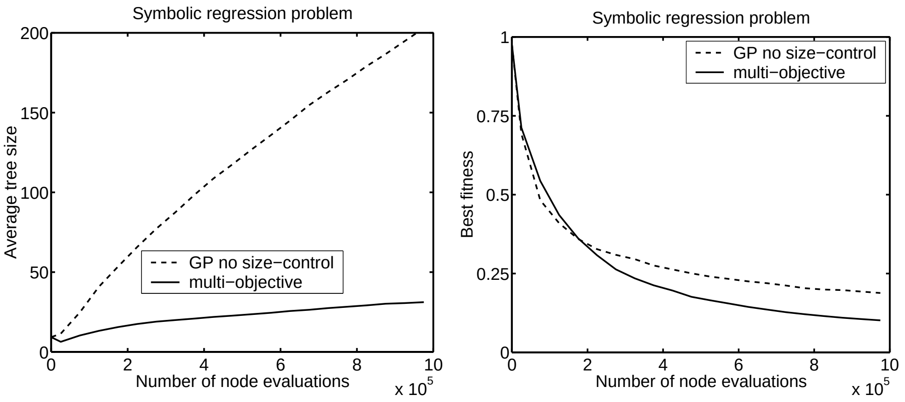

Figure 7. Development of tree size and fitness for the symbolic regression problem.

The principle of the construction is as follows. We divide the sequence of input bits in half and apply a parity function to the parity of each half, where the parities of the halves are obtained by recursively applying the same procedure again. For sub-sequences of size one, i.e. single bits, the bit itself is used instead of its parity. When this occurs for precisely one of the two arguments, the outcome would be inverted, and an odd 2-parity function is used instead of even-parity to correct this.

Let S be a binary sequence of length $|S| = n \geq 2$. S is divided in half yielding two subsequences L and R with, for even n, length $\frac{n}{2}$ or, for odd n, lengths $\frac{n-1}{2}$ and $\frac{n+1}{2}$. Then the following recursively defined function P(S) gives a correct expression for the even-parity of S for $|S| \geq 2$ in terms of the above operators:

$$P(S) = \begin{cases} S & \text{if } |S| = 1 \\ \text{Odd}(P(L), P(R)) & \text{if } |S| > 1 \wedge \text{must\_invert}(L, R) \\ \text{Even}(P(L), P(R)) & \text{otherwise} \end{cases}$$

where
Odd(A, B) = NOR(AND(A, B), NOR(A, B))
Even(A, B) = OR(AND(A, B), NOR(A, B))

$$\text{must\_invert}(A, B) = \begin{cases} \text{True} & \text{if } |A| = 1 \ \wedge \ |B| \neq 1 \\ \text{True} & \text{if } |A| \neq 1 \ \wedge \ |B| = 1 \\ \text{False} & \text{otherwise} \end{cases}$$

Table II. Length of hand-constructed solutions to n-parity (see text).

n	1	2	3	4	5	6	7
Length	3	7	19	31	55	79	103

The length $|P(S)|$ of the expression $P(S)$ satisfies:

$$|P(S)| = \begin{cases} 1 & \text{for} \quad |S| = 1 \\ 3 + 2|P(L)| + 2|P(R)| & \text{for} \quad |S| > 1 \end{cases}$$

For $n = 2^i, i > 0$, this expression can be shown to equal $2n^2 - 1$. Table II gives the lengths of the expressions for the first seven even-n-parity problems. For $|S| = 1$, the shortest expression is NOR(S, S); for $|S| > 1$, the length is given by the above expression. The rapid growth with increasing order stems from the repeated doubling of the required inputs. This makes clear why using a `NOR` operator makes the problem more difficult than using `XOR`.

7.2. THE FOCUS ALGORITHM

We now describe a multi-objective algorithm that combines the active promotion of diversity with a high degree of elitism. The algorithm aims to Find Only and Complete Undominated Sets, and is therefore called FOCUS. This is pursued by keeping all and only those individuals that are non-dominated with respect to the individuals encountered so far. The resulting elitism is countered by employing a diversity objective, measuring each individual's contribution to population diversity. The idea is that by explicitly rewarding individuals for diversity, the usual procedure of maintaining arbitrary low fitness individuals that happen to be present can be avoided.

The efficiency of FOCUS was compared to genetic programming in (De Jong et al., 2001). Here, we will describe the algorithm, and consider its ability to find compact solutions.

The population is initialized with m randomly created individuals. A cycle proceeds as follows. A chosen number of n new individuals is generated based on the current population using crossover and mutation, and added to the population. If a new individual already exists in the population, it is mutated. If the result also exists, it is discarded; otherwise it is added to the population. All individuals are then evaluated. After evaluation, all population members are checked against all other population members, and removed if dominated by any of them. As in section 6, of multiple individuals having the same objective vectors, only one individual is retained.

The removal of dominated individuals and individuals with equal objective vectors leads to small population sizes: populations in the following experiments contain between 5 and 50 individuals on average. This has a positive effect on the speed of the algorithm as it reduces the cost of the existence and dominance checks, and furthermore makes it possible to perform high numbers of generations. The removal of individuals with equal objective vectors leads to a high degree of elitism, and may be thought to reduce the potential for exploration. However, the active promotion of diversity counteracts this effect and explicitly favors different individuals.

In the experiments, population size parameters of $m = n = 300$ are used. The probability of crossover is 90% and that of mutation 10%. Random individuals contain between 1 and 20 internal nodes and are generated as described. The objectives used are fitness, size, and diversity. For the diversity objective, the following distance measure is used. The distance between two corresponding nodes is zero if they are identical and one if they are not. The distance between two trees is the sum of the distances of the corresponding nodes, i.e. nodes that overlap when the two trees are overlaid, starting from the root. The distance between two trees is normalized by dividing by the size of the smaller of the two trees.

7.3. FINDING COMPACT TREES USING THE FOCUS ALGORITHM

We investigate to what extent the method described above can be used to find compact solutions to n-parity problems. This is done as follows. To average over the different runs, we align them at the point where the first correct solution is found, and monitor the smallest correct solution from then on. For 3-parity, the results are as follows. The first correct solution found is of size 30 on average. After finding a first solution, the algorithm rapidly finds smaller correct solutions: the average size drops to 22 within on average 4,000 additional fitness evaluations, and converges to around 20. The smallest tree, found in 12 out of 30 runs, was 19. As seen from table II, this equals the hypothesized minimum size for even 3-parity trees. For 4-parity, solutions dropped in size from an initial 68.5 to 50 in about 10,000 fitness evaluations, and to 41 on average when runs were continued to around 85,000 evaluations. In 12 of the 30 runs, solutions of hypothesized minimum size (31 nodes) were found. Finally, using the FOCUS algorithm on the 5-parity problem, the smallest solution found was of size 55, which again equals the

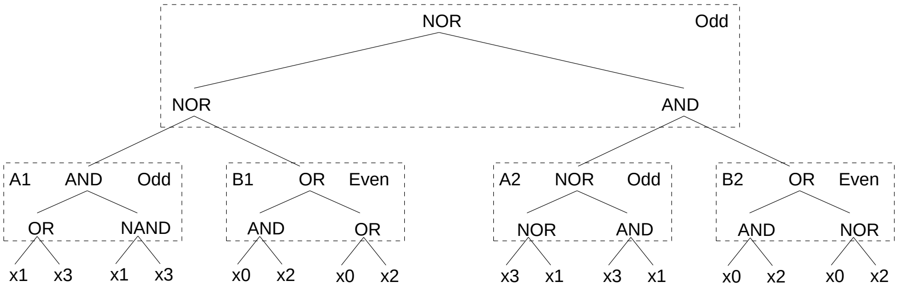

Figure 8. Solution to 4-parity of hypothesized minimum size (31 nodes) found by the FOCUS algorithm. The function consists entirely of equivalents of Even and Odd functions. It can therefore be represented in simplified form, as shown in figure 9 (right).

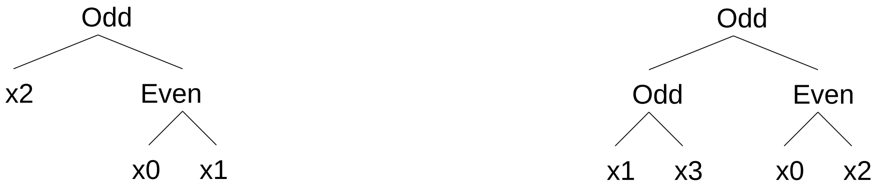

Figure 9. Hypothesized minimum size solutions to 3-parity (left) and 4-parity (right) found by the FOCUS algorithm, shown in simplified form (see text).

hypothesized minimum size. In summary, for all three problems, multi-objective size control combined with active diversity promotion led to solutions of the hypothesized optimally compact size.

To determine whether the reduction of bloat improves the intelligibility of solutions, we inspect the hypothesized minimal size solutions found for the 3-, 4- and 5-parity problems. Figure 8 shows a correct minimal size solution found for even 4-parity. The top 3 operators (`NOR(NOR(A1,B1),AND(A2,B2))`) specify the `Odd(A,B)` function which was also used in our construction, provided that A1 $\equiv$ A2 and B1 $\equiv$ B2. The Odd function can also be built using `AND(NAND,OR)`, and likewise the Even function can be built using `OR(AND,NOR)` or `NAND(NAND,OR)`. By writing occurrences of these constructs as `Even(A,B)` and `Odd(A,B)`, the function can be represented in a more compact manner (fig. 9), and it is seen that the function is simply a combination of binary `Even` and `Odd` functions. The solution to 3-parity that we have inspected has the same structure, as shown in the same figure.

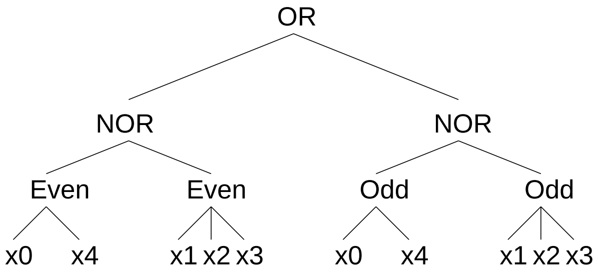

Figure 10. Hypothesized minimum size solution to 5-parity found by the FOCUS algorithm, shown in simplified form. Interestingly, the function employs ternary `Even` and `Odd` functions. It returns `True` if the numbers of ones in two subsets of the variables are both odd (left subtree) or both even (right subtree).

The minimal size solution for 5-parity was again equal to the size of our hypothesized minimum size construction, but introduces a new principle. Rather than constructing the function out of binary `Even` and `Odd` functions, the search has produced ternary `Even` and `Odd` functions. The resulting solution is shown in simplified form in figure 10.

Finally, we note that larger instances of the parity problem can be addressed by the use of methods that form modules during the search such as ADFs (Koza, 1994) and GLiB (Angeline & Pollack, 1992). Therefore, the combination of such methods with multi-objective size control may provide an interesting direction for further research.

8. Conclusions

Bloat can lead to large and progressive increases in the time required for the evaluation of individuals. This impacts both the design of search algorithms and the measurement of their performance. Concerning the latter, we have proposed the use of node evaluations, rather than tree or fitness evaluations, as an indicator of computational effort spent when measuring performance.

The main part of the article studies the use of Evolutionary Multi-Objective Optimization (EMOO) for size control. It has been observed that using size and fitness as objectives in an EMOO setup can lead to a convergence to small size individuals. We have presented an empirical analysis showing that the distribution of newly generated non-dominated trees is highly skewed towards trees of sizes smaller than the current average tree size. Thus, repeated replacement of existing trees by uniformly selected newly generated non-dominated trees will lead to a decreasing average tree size, as confirmed by experiments.

This finding suggests that multi-objective approaches to bloat require specific attention to diversity maintenance. To test this explanation, we have taken a basic multi-objective method, and added a minimal mechanism for diversity maintenance. This modification addressed the problem of premature convergence to small trees, and thereby supports this conclusion.

To test the efficacy of the resulting approach as a method for size control, we have performed experiments on three test problems: even 6-parity with the `AND, OR, NAND,` and `NOR` operators, the 11-multiplexer, and a symbolic regression problem. On all three problems the multi-objective approach adequately controlled the tree size, and significantly improved performance as a function of computational expense.

Finally, we present the FOCUS algorithm, which actively promotes diversity by making it an explicit objective. The FOCUS algorithm finds solutions of hypothesized minimum size for the difficult NOR-version of the even n-parity problems up to 5-parity. These results also demonstrate a valuable side effect of size control, in that the solutions are easily interpretable. This strongly contrasts with standard genetic programming, which due to bloat produces solutions that are typically very difficult to interpret, see (Keijzer, 2002).

We conclude that evolutionary multi-objective optimization in combination with diversity maintenance can provide an adequate and parameterless approach to tree size control in genetic programming. The proposed method may also be of use in other forms of evolutionary computation that employ variable size representations.

A promising direction for further improvement may be to combine multi-objective size control with more powerful techniques for diversity maintenance, see e.g. (Laumanns, Thiele, Deb, & Zitzler, 2002). Furthermore, while it will remain necessary to balance fitness and size, the final selection of a solution will typically be based on fitness. Thus, performance improvement may be possible by focusing on the high fitness end of the tradeoff front during evolution.

Acknowledgments

The authors want to thank Maarten Keijzer, Bill Langdon, Sean Luke, the members of the DEMO-Lab, and the anonymous reviewers for useful comments and suggestions. EdJ gratefully acknowledges a Fulbright grant and a TALENT fellowship from the Netherlands Organization for Scientific Research (NWO).

References

Angeline, P. J. (1994). Genetic programming and emergent intelligence. In K. E. Kinnear, Jr. (Ed.), *Advances in Genetic Programming* (pp. 75–98). Cambridge, MA: The MIT Press.

Angeline, P. J., & Pollack, J. B. (1992). The evolutionary induction of subroutines. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society* (p. 236-241). Hillsdale, NJ: Lawrence Erlbaum Associates.

Banzhaf, W., Banscherus, D., & Dittrich, P. (1998). Hierarchical genetic programming using local modules. *InterJournal Complex Systems*(228). (URL: http://www.interjournal.org)

Banzhaf, W., & Langdon, W. B. (2002). Some considerations on the reason for bloat. *Genetic Programming and Evolvable Machines*, *3*(1), 81–91.

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic Programming – An Introduction: On the Automatic Evolution of Computer Programs and its Applications.* San Francisco, CA: Morgan Kaufmann.

Blickle, T., & Thiele, L. (1994). Genetic programming and redundancy. In J. Hopf (Ed.), *Genetic Algorithms within the Framework of Evolutionary Computation. Workshop at KI-94* (pp. 33–38). Saarbrücken, Germany: Max-Planck-Institut für Informatik (MPI-I-94-241).

Brindle, A. (1981). *Genetic Algorithms for Function Optimization.* Unpublished doctoral dissertation, University of Alberta, Canada. (Computer Science Department, Technical Report TR81-2)

Coello, C. A. C. (2000). An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, *32*(2), 109–143.

Conover, W. (1980). *Practical Nonparametric Statistics.* New York, NY: Wiley & Sons.

Corne, D. W., Jerram, N. R., Knowles, J. D., & Oates, M. J. (2001). PESA-II: Region-based selection in evolutionary multiobjective optimization. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, & E. Burke (Eds.), *Proceedings of the Genetic* *and Evolutionary Computation Conference, GECCO-2001* (pp. 283–290). San Francisco, CA: Morgan Kaufmann.

Cramer, N. L. (1985). A representation for the adaptive generation of simple sequential programs. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (p. 183-187). Hillsdale, NJ: Lawrence Erlbaum Associates.

De Jong, E. D., Watson, R. A., & Pollack, J. B. (2001). Reducing bloat and promoting diversity using multi-objective methods. In L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, & E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001* (p. 11-18). San Francisco, CA: Morgan Kaufmann.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms.* New York, NY: Wiley & Sons.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature - PPSN VI* (Vol. 1917, pp. 849–858). Berlin: Springer.

Ekárt, A., & Németh, S. (2001). Selection based on the Pareto nondomination criterion for controlling code growth in genetic programming. *Genetic Programming and Evolvable Machines*, *2*, 61-73.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms, ICGA-93* (pp. 416–423). San Francisco, CA: Morgan Kaufmann.

Fonseca, C. M., & Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, *3*(1), 1–16.

Fourman, M. P. (1985). Compaction of Symbolic Layout using Genetic Algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and their*

Applications (pp. 141–153). Hillsdale, NJ: Lawrence Erlbaum Associates.

Keijzer, M. (2002). *Scientific Discovery Using Genetic Programming.* Unpublished doctoral dissertation, Danish Technical University, Lyngby.

Kinnear, Jr., K. E. (1993). Generality and difficulty in genetic programming: Evolving a sort. In S. Forrest (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms, ICGA-93* (pp. 287–294). San Francisco, CA: Morgan Kaufmann.

Koza, J. R. (1992). *Genetic Programming.* Cambridge, MA: The MIT Press.

Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* Cambridge, MA: The MIT Press.

Langdon, W. B. (1996). Data structures and genetic programming. In P. J. Angeline & K. Kinnear (Eds.), *Advances in Genetic Programming 2* (p. 395-414). Cambridge, MA: The MIT Press.

Langdon, W. B. (1998). The evolution of size in variable length representations. In *1998 IEEE International Conference on Evolutionary Computation* (pp. 633–638). Piscataway, NJ: IEEE Press.

Langdon, W. B., & Nordin, J. P. (2000). Seeding GP populations. In R. Poli, W. Banzhaf, W. B. Langdon, J. F. Miller, P. Nordin, & T. C. Fogarty (Eds.), *Genetic Programming, Proceedings of EuroGP-2000* (Vol. 1802, pp. 304–315). Berlin: Springer.

Langdon, W. B., & Poli, R. (1998). Fitness causes bloat: Mutation. In W. Banzhaf, R. Poli, M. Schoenauer, & T. C. Fogarty (Eds.), *Proceedings of the First European Workshop on Genetic Programming* (Vol. 1391, pp. 37–48). Berlin: Springer.

Langdon, W. B., & Poli, R. (2002). *Foundations of Genetic Programming.* Berlin: Springer.

Langdon, W. B., Soule, T., Poli, R., & Foster, J. A. (1999). The evolution of size and shape. In L. Spector, W. B. Langdon, U.-M. O'Reilly, & P. J. Angeline (Eds.), *Advances in Genetic Programming 3* (pp. 163–190). Cambridge, MA: The MIT Press.

Laumanns, M., Thiele, L., Deb, K., & Zitzler, E. (2002). Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, *10*(3), 263–282.

Luke, S., & Panait, L. (2002). Fighting bloat with nonparametric parsimony pressure. In H.-P. Schwefel, J.-J. Merelo Guervós, P. Adamidis, H.-G. Beyer, & J.-L. Fernández-Villacañas (Eds.), *Parallel Problem Solving from Nature - PPSN VII.* (p. 411 ff.). Berlin: Springer.

Mahfoud, S. W. (1995). *Niching Methods for Genetic Algorithms.* Unpublished doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL. (IlliGAL Report 95001)

McPhee, N. F., & Miller, J. D. (1995). Accurate replication in genetic programming. In L. Eshelman (Ed.), *Genetic Algorithms: Proceedings of the Sixth International Conference, ICGA-95* (pp. 303–309). San Francisco, CA: Morgan Kaufmann.

Nordin, P., & Banzhaf, W. (1995). Complexity compression and evolution. In L. Eshelman (Ed.), *Genetic Algorithms: Proceedings of the Sixth International Conference, ICGA-95* (pp. 310–317). San Francisco, CA: Morgan Kaufmann.

Nordin, P., Francone, F., & Banzhaf, W. (1996). Explicitly defined introns and destructive crossover in genetic programming. In P. J. Angeline & K. E. Kinnear, Jr. (Eds.), *Advances in Genetic Programming 2* (pp. 111–134). Cambridge, MA: The MIT Press.

Olsson, R. (1995). Inductive functional programming using incremental program transformation. *Artificial Intelligence*, *74*(1), 55–81.

Poli, R. (1997). Discovery of symbolic, neuro-symbolic and neural networks with parallel distributed genetic programming. In *3rd International Conference on Artificial Neural Networks and Genetic Algorithms, ICANNGA'97.* Berlin: Springer.

Poli, R., & McPhee, N. F. (2001). Exact schema theorems for gp with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tettamanzi, & W. Langdon (Eds.), *Genetic Programming. 4th European Conference, EuroGP 2001* (pp. 126–142). Berlin: Springer.

Rodríguez-Vázquez, K., Fonseca, C. M., & Fleming, P. J. (1997). Multiobjective genetic programming: A nonlinear system identification application. In J. R. Koza (Ed.), *Late Breaking Papers at the 1997 Genetic Programming Conference* (pp. 207–212). Stanford University, CA: Stanford Bookstore.

Rosca, J. (1996). Generality versus size in genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, & R. L. Riolo (Eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference* (pp. 381–387). Cambridge, MA: The MIT Press.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and their Applications* (pp. 93–100). Hillsdale, NJ: Lawrence Erlbaum Associates.

Smith, S. (1980). *A Learning System based on Genetic Adaptive Algorithms.* Unpublished doctoral dissertation, University of Pittsburgh.

Soule, T. (1998). *Code Growth in Genetic Programming.* Unpublished doctoral dissertation, University of Idaho.

Soule, T., & Foster, J. A. (1999). Effects of code growth and parsimony presure on populations in genetic programming. *Evolutionary Computation*, *6*(4), 293–309.

Soule, T., & Heckendorn, R. B. (2002). An analysis of the causes of code growth in genetic programming. *Genetic Programming and Evolvable Machines*, *3*, 283-309.

Zhang, B.-T., & Mühlenbein, H. (1995). Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation*, *3*(1), 17–38.

Zitzler, E., & Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, *3*(4), 257–271.