Reducing Bloat and Promoting Diversity using Multi-Objective Methods

Edwin D. de Jong[1,2] **Richard A. Watson**[2] **Jordan B. Pollack**[2]
{edwin, richardw, pollack}@cs.brandeis.edu
[1]Vrije Universiteit Brussel, AI Lab, Pleinlaan 2, B-1050 Brussels, Belgium
[2]Brandeis University, DEMO Lab, Computer Science dept., Waltham, MA 02454, USA

Category: Genetic Programming

Abstract

Two important problems in genetic programming (GP) are its tendency to find unnecessarily large trees (bloat), and the general evolutionary algorithms problem that diversity in the population can be lost prematurely. The prevention of these problems is frequently an implicit goal of basic GP. We explore the potential of techniques from multi-objective optimization to aid GP by adding explicit objectives to avoid bloat and promote diversity. The even 3, 4, and 5-parity problems were solved efficiently compared to basic GP results from the literature. Even though only non-dominated individuals were selected and populations thus remained extremely small, appropriate diversity was maintained. The size of individuals visited during search consistently remained small, and solutions of what we believe to be the minimum size were found for the 3, 4, and 5-parity problems.

Keywords: genetic programming, code growth, bloat, introns, diversity maintenance, evolutionary multi-objective optimization, Pareto optimality

1 INTRODUCTION

A well-known problem in genetic programming (GP), is the tendency to find larger and larger programs over time (Tackett, 1993; Blickle & Thiele, 1994; Nordin & Banzhaf, 1995; McPhee & Miller, 1995; Soule & Foster, 1999), called *bloat* or *code growth.* This is harmful since it results in larger solutions than necessary. Moreover, it increasingly slows down the rate at which new individuals can be evaluated. Thus, keeping the size of trees that are visited small is generally an implicit objective of GP.

Another important issue in GP and in other methods of evolutionary computation is that of how diversity of the population can be achieved and maintained. A population that is spread out over promising parts of the search space has more chance of finding a solution than one that is concentrated on a single fitness peak. Since members of a diverse population solve parts of the problem in different ways, it may also be more likely to discover partial solutions that can be utilized through crossover. Diversity is not an objective in the conventional sense; it applies to the populations visited during the search, not to final solutions. A less obvious idea then is to view the contribution of individuals to population diversity as an objective.

Multi-objective techniques are specifically designed for problems in which knowledge about multiple objectives is available, see e.g. Fonseca and Fleming (1995) for an overview. The main idea of this paper is to use multi-objective techniques to add the objectives of size and diversity in addition to the usual objective of a problem-specific fitness measure. A multi-objective approach to bloat appears promising and has been used before (Langdon, 1996; Rodriguez-Vazquez, Fonseca, & Fleming, 1997), but has not become standard practice. The reason may be that basic multi-objective methods, when used with small tree size as an objective, can result in premature convergence to small individuals (Langdon & Nordin, 2000; Ekart, 2001). We therefore investigate the use of a size objective in combination with explicit diversity maintenance.

The remaining sections discuss the n-parity problem (2), bloat (3), multi-objective methods (4), diversity maintenance(5), ideas behind the approach, called FOCUS, (6), algorithmic details (7), results (8), and conclusions (9).

2 THE N-PARITY PROBLEM

The test problems that will be used in this paper are even n-parity problems, with n ranging from 3 to 5. A correct solution to this problem takes a binary sequence of length n as input and returns true (one) if

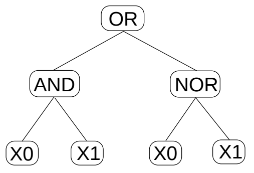

Figure 1: A correct solution to the 2-parity problem

the number of ones in the sequence is even, and false (zero) if it is odd. It is named *even* to avoid confusion with the related odd parity problem, which gives the inverse answer. Trees may use the following boolean operators as internal nodes: AND, OR, NAND, and NOR. Each leaf specifies an element of the sequence. The fitness is the fraction of all possible length n binary sequences for which the program returns the correct answer. Figure 1 shows an example.

The n-parity problem has been selected because it is a difficult problem that has been used by a number of researchers. With increasing order, the problem quickly becomes more difficult. One way to understand its hardness is that for any setting of the bits, flipping any bit inverts the outcome of the parity function. Equivalently, its Karnaugh map (Zissos, 1972) equals a checkerboard function, and thus has no adjacencies.

2.1 SIZE OF THE SMALLEST SOLUTIONS TO N-PARITY

We believe that the correct solutions to n-parity constructed as follows are of minimal size, but are not able to prove this. The principle is to recursively divide the bit sequence in half and, take the parity of each halve, and feed these two into a parity function. For subsequences of size one, i.e. single bits, the bit itself is used instead of its parity. When this occurs for one of the two arguments, the outcome would be inverted, and thus the odd 2-parity function is used to obtain the even 2-parity of the bits.

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:

$$P(S) = \begin{cases} S & \text{if} \quad |S| = 1 \\ \text{ODD}(P(L), P(R)) & \text{if} \quad |S| > 1 \wedge g(L, R) \\ \text{EVEN}(P(L), P(R)) & \text{otherwise} \end{cases}$$

where
ODD(A, B) = NOR(AND(A, B), NOR(A, B)),
EVEN(A, B) = OR(AND(A, B), NOR(A, B)), and

$$g(A, B) = \begin{cases} \text{TRUE} & \text{if} \quad (|A| = 1) \text{ XOR } (|B| = 1) \\ \text{FALSE} & \text{else} \end{cases}$$

Table 1: Length of the shortest solution to n-parity using the operators AND, OR, NAND, and NOR.

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 1 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.

3 THE PROBLEM OF BLOAT

A well-known problem, known as *bloat* or *code growth*, is that the trees considered during a GP run grow in size and become larger than is necessary to represent good solutions. This is undesirable because it slows down the search by increasing evaluation and manipulation time and, if the growth consists largely of non-functional code, by decreasing the probability that crossover or mutation will change the operational part of the tree. Also, compact trees have been linked to improved generalization (Rosca, 1996).

Several causes of bloat have been suggested. First, under certain restrictions (Soule, 1998), crossover favors smaller than average subtrees in removal but not in replacement. Second, larger trees are more likely to produce fit (and large) offspring because non-functional code can play a protective role against crossover (Nordin & Banzhaf, 1995) and, if the probability of mutating a node decreases with increasing tree size, against mutation. Third, the search space contains more large than small individuals (Langdon & Poli, 1998).

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, and hence it may be concluded that in those experiments bloat was mainly due to growth of ineffective code (*introns*).

Finally, it is conceivable that in some circumstances non-functional code may be useful. It has been suggested that introns may be useful for retaining code that is not used in the current individual but is a helpful building block that may be used later (Nordin, Francone, & Banzhaf, 1996).

Table 2: Properties of the basic GP method used.

Problem	3-Parity
Fitness	Fraction of correct answers
Operators	AND, OR, NAND, and NOR
Stop criterion	500,000 evaluations or solution
Initial tree size	Uniform [1..20] internal nodes
Cycle	generational
Population Size	1000
Parent selection	Boltzmann with T = 0.1
Replacement	Complete
Uniqueness check	Individuals occur at most once
P(crossover)	0.9
P(mutation)	0.1
Mutation method	Mutate node with P = $\frac{1}{n}$

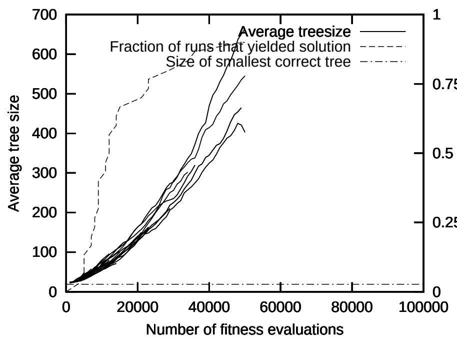

Figure 2: Average tree sizes of ten different runs (solid lines) using basic GP on the 3-parity program.

3.1 OBSERVATION OF BLOAT USING BASIC GP

To confirm that bloat does indeed occur in the test problem of n-parity using basic GP, thirty runs where performed for the 3-parity problem. The parameters of the run are shown in Table 2. A run ends when a correct solution has been found. Figure 2 shows that average tree sizes increase rapidly in each run. If a solution is not found at an early point in the run, bloating rapidly increases the sizes of the trees in the population, thus increasingly slowing down the search. A single run of 111,054 evaluations already took more than 15 hours on a current PC running Linux due to the increasing amount of processing required per tree as a result of bloat. The population of size-unlimited trees that occurred in the single 4-parity run that was tried (with trees containing up to 6,000 nodes) filled virtually the entire swap space and caused performance to degrade to impractical levels. Clearly, the problem of bloat must be addressed in order to solve these and higher order versions of the problem in an efficient manner.

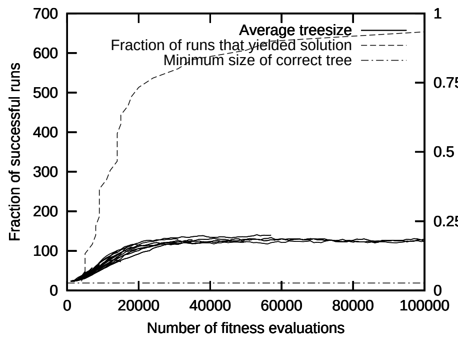

Figure 3: Average tree sizes and fraction of successful runs in the 3-parity problem using basic GP with a tree size limit of 200. Tree sizes are successfully limited, of course, but the approach is not ideal (see text).

3.2 USING A FIXED TREE SIZE LIMIT

Probably the most common way to avoid bloat is to simply limit the allowed tree size or depth (Langdon & Poli, 1998; Koza, 1992), although the latter has been found to lead to loss of diversity near the root node when used with crossover (Gathercole & Ross, 1996). Figure 3 shows the effect of using a limit of 200 on 3-parity. This limit is well above the minimum size of a correct solution, but not too high either since several larger solutions were found in the unrestricted run. The average tree size is around 140 nodes.
On the 4-parity problem (with a tree size limit of 200), the average tree size varied around 150. However, whereas on 3-parity 90% of the runs found a solution within 100,000 evaluations, on 4-parity only 33% of the runs found a solution within 500,000 evaluations, testifying to the increased difficulty of this order of the parity problem. For 5-parity, basic GP found no solutions within 1,000,000 evaluations for any of the 30 runs. Thus, our version of GP with fixed tree size limit does not scale up well. Furthermore, a fundamental problem with this method of preventing bloat is that the maximum tree size has to be selected before the search, when it is often unknown.

3.3 WEIGHTED SUM OF FITNESS AND SIZE

Instead of choosing a fixed tree size limit in advance one would rather like to have the algorithm search for trees that can be as large as they need to be, but not much larger. A popular approach that goes some way towards this goal is to include a component in the fitness that rewards small trees or programs. This is mostly done by adding a component to the fitness, thus making fitness a linear combination of a performance measure and a parsimony measure (Koza, 1992; Soule, Foster, & Dickinson, 1996). However, this approach is not without its own problems (Soule & Fos-

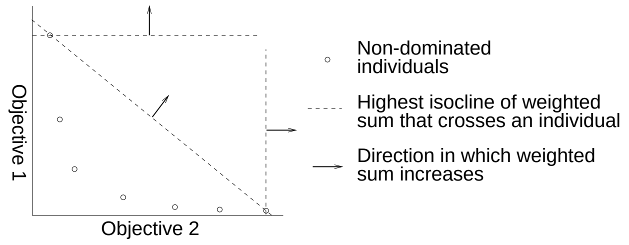
Non-dominated individuals
Highest isocline of weighted sum that crosses an individual
Direction in which weighted sum increases
Objective 1
Objective 2

Figure 4: Schematic rendition of a concave tradeoff surface. This occurs when better performance in one objective means worse performance in the other, *vice versa*. The lines mark the maximum fitness individuals for three example weightings (see vectors) using a linear weighting of the objectives. No linear weighting exists that finds the in-between individuals, with reasonable performance in both objectives.

ter, 1999). First, the weight of the parsimony measure must be determined beforehand, and so a choice concerning the tradeoff between size and performance is already made before the search. Furthermore, if the tradeoff surface between the two fitness components is concave[1] (see Fig. 4), a linear weighting of the two components favors individuals that do well in one of the objectives, but excludes individuals that perform reasonably in both respects (Fleming & Pashkevich, 1985).

Soule and Foster (1999) have investigated why a linear weighting of fitness and size has yielded mixed results. It was found that a weight value that adequately balances fitness and size is difficult to find. However, if the required balance is different for different regions in objective space, then adequate parsimony pressure cannot be specified using a single weight. If this is the case, then methods should be used that do not attempt to find such a single balance. This idea forms the basis of multi-objective optimization.

4 MULTI-OBJECTIVE METHODS

After several early papers describing the idea of optimizing for multiple objectives in evolutionary computation (Schaffer, 1985; Goldberg, 1989), the approach has recently received increasing attention (Fonseca & Fleming, 1995; Van Veldhuizen, 1999). The basic idea is to search for multiple solutions, each of which satisfy the different objectives to different degrees. Thus, the selection of the final solution with a particular combination of objective values is postponed until a time when it is known what combinations exist.

A key concept in multi-objective optimization is that of dominance. Let individual x_A have values A_i for the n objectives, and individual x_B have objective values B_i. Then A dominates B if

$$\forall i \in [1..n] : A_i \geq B_i \wedge \exists i : A_i > B_i$$

Multi-objective optimization methods typically strive for *Pareto optimal* solutions, i.e. individuals that are not dominated by any other individuals.

5 DIVERSITY MAINTENANCE

A key difference between classic search methods and evolutionary approaches is that in the latter a *population* of individuals is maintained. The idea behind this is that by maintaining individuals in several regions of the search space that look promising (*diversity maintenance*), there is a higher chance of finding useful material from which to construct solutions.

In order to maintain the existing diversity of a population, evolutionary methods typically keep some or many of the individuals that happen to have been generated and have relatively high fitness, but lower than that found so far. In the same way, evolutionary multi-objective methods usually keep some dominated individuals in addition to the non-dominated individuals (Fonseca & Fleming, 1993). However, this appears to be a somewhat arbitrary way of maintaining diversity. In the following section, we present a more directed method. The relation to other diversity maintenance methods is discussed.

6 THE *FOCUS* METHOD

We propose to do diversity maintenance by using a basic multi-objective algorithm and including an objective that actively promotes diversity. To the best of our knowledge, this idea has not been used in other work, including multi-objective research. If it works well, the need for keeping arbitrary dominated individuals may be avoided. To test this, we use the diversity objective in combination with a multi-objective method that only keeps non-dominated individuals, as reported in section 8.

The approach strongly directs the attention of the search towards the explicitly specified objectives. We therefore name this method FOCUS, which stands for Find Only and Complete Undominated Sets, reflecting the fact that populations only contain non-dominated individuals, and contain all such individuals encountered so far. Focusing on non-dominated individuals combines naturally with the idea that the objectives are responsible for exploration, and this combination defines the FOCUS method.

The concept of diversity applies to populations, meaning that they are dispersed. To translate this aim into an objective for individuals, a metric has to be defined that, when optimized by individuals, leads to diverse populations. The metric used here is that of average

[1]Since fitness is to be *maximized*, the tradeoff curve shown is concave.

squared distance to the other members of the population. When this measure is maximized, individuals are driven away from each other.

Interestingly, the average distance metric strongly depends on the current population. If the population were centered around a single central peak in the fitness landscape, then individuals that moved away from that peak could survive by satisfying the diversity objective better than the individuals around the fitness peak. It might be expected that this would cause large parts of the population to occupy regions that are merely far away from other individuals but are not relevant to the problem. However, if there are any differences in fitness in the newly explored region of the search space, then the fitter individuals will come to replace individuals that merely performed well on diversity. When more individuals are created in the same region, the potential for scoring highly on diversity for those individuals diminishes, and other areas will be explored. The dynamics thus created are a new way to maintain diversity.

Other techniques that aim to promote diversity in a directed way exist, and include fitness sharing (Goldberg & Richardson, 1987; Deb & Goldberg, 1989), deterministic crowding (Mahfoud, 1995), and fitness derating (Beasley, Bull, & Martin, 1993). A distinguishing feature of the method proposed here is that in choosing the diversity objective, problem-based criteria can be used to determine which individuals should be kept for exploration purposes.

7 ALGORITHM DETAILS

The algorithm selects individuals if and only if they are not dominated by other individuals in the population. The population is initialized with 300 randomly created individuals of 1 to 20 internal nodes. A cycle proceeds as follows. A chosen number n of new individuals (300) is generated based on the current population using crossover (90%) and mutation (10%). If the 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 if necessary. After evaluation, all population members are checked against other population members, and removed if dominated by any of them.

A slightly stricter criterion than Pareto's is used: A dominates B if $\forall i \in [1..n] : A_i \geq B_i$. Of multiple individuals occupying the same point on the tradeoff surface, precisely one will remain, since the removal criterion is applied sequentially. This criterion was used because the Pareto criterion caused a proliferation of individuals occupying the same point on the trade-off surface when no diversity objective was used[2].

[2]In later experiments including the diversity objective, this proliferation was not observed, and the standard Pareto criterion also worked satisfactorily.

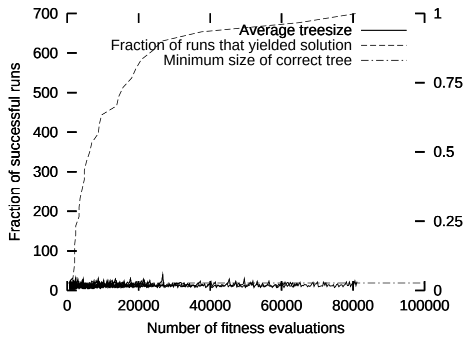

Figure 5: Average tree size and fraction of successful runs for the [fitness, size, diversity] objective vector on the 3-parity problem. The trees are much smaller than for basic GP, and solutions are found faster.

The following distance measure is used in the diversity objective. 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 tree of the two.

8 EXPERIMENTAL RESULTS

In the following experiments we use fitness, size, and diversity as objectives. The implementation of the objectives is as follows. Fitness is the fraction of all 2^n input combinations handled correctly. For size, we use 1 over the number of nodes in the tree as the objective value. The diversity objective is the average squared distance to the other population members.

8.1 USING FITNESS, SIZE, AND DIVERSITY AS OBJECTIVES

Fig. 5 shows the graph of Fig. 3 for the method of using fitness, size, and diversity as objectives. The average tree size remains extremely small. In addition, a glance at the graphs indicates that correct solutions are found more quickly. To determine whether this is indeed the case, we compute the *computational effort*, i.e. the expected number of evaluations required to yield a correct solution with a 99% probability, as described in detail by Koza (1994).

The impression that correct solutions to 3-parity are found more quickly for the multi-objective approach (see Figure 6) is confirmed by considering the computational effort E; whereas GP with the tree size limit requires 72,044 evaluations, the multi-objective approach requires 42,965 evaluations. For the 4-parity problem, the difference is larger; basic GP needs

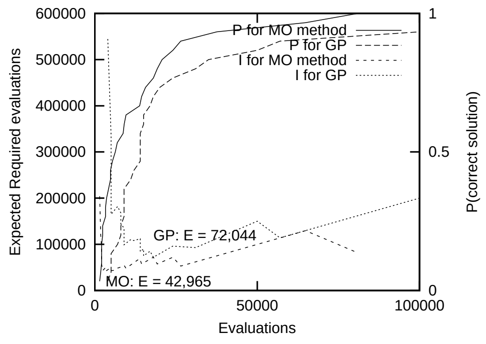

Figure 6: Probability of finding a solution and computational effort for 3-parity using basic GP and the multi-objective method.

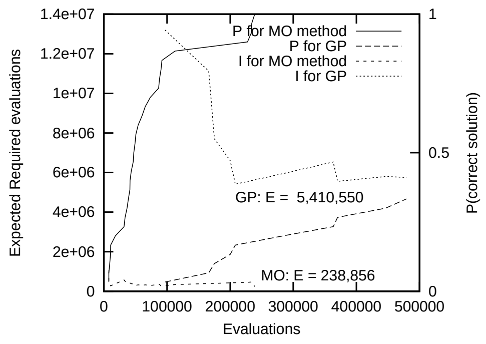

Figure 7: Probability of finding a solution and computational effort for 4-parity for basic GP and the multi-objective method. The performance of the multi-objective method is considerably superior.

5,410,550 evaluations, whereas the multi-objective approach requires only 238,856. This is a dramatic improvement, and demonstrates that our method can be very effective.

Finally, experiments have been performed using the even more difficult 5-parity problem. For this problem, basic GP did not find any correct solutions within a million evaluations. The multi-objective method did find solutions, and did so reasonably efficiently, requiring a computational effort of 1,140,000 evaluations.

Table 3 summarizes the results of the experiments. Considering the average size of correct solutions on 3-parity, the multi-objective method outperforms all methods that have been compared, as the first solution it finds has 30.4 nodes on average. What's more, the multi-objective method also requires a smaller number of evaluations to do so than the other methods. Finally, perhaps most surprisingly, it finds correct solutions using extremely small populations, typically containing less than 10 individuals. For example, the average population size over the whole experiment for 3-parity was 6.4, and 8.5 at the end of the experiment, and the highest population size encountered in all 30 runs was 18. This suggests that the diversity maintenance achieved by using this greedy multi-objective method in combination with an explicit diversity objective is effective, since even extremely small populations did not result in premature convergence.

Table 3: Results of the experiments (GP and Multi-Objective rows). For comparison, results of Koza's (1994) set of experiments (population size 16,000) and the best results with other configurations (population size 4,000) found there. E: computational effort, S: average tree size of first solution, Pop: average population size.

3-parity	E	S	Pop
GP	72,044	93.67	1000
Multi-objective	42,965	30.4	6.4
Koza GP	96,000	44.6	16,000
Koza GP-ADF	64,000	48.2	16,000
4-parity	E	S	Pop
GP	5,410,550	154	1000
Multi-objective	238,856	68.5	15.8
Koza GP	384,000	112.6	16,000
Koza GP-ADF	176,000	60.1	16,000
5-parity	E	S	Pop
GP	∞^1	n.a.	n.a
Multi-objective	1,140,000	218.7	49.7
Koza GP	6,528,000	299.9	16,000
Koza GP	1,632,000	299.9	4,000
Koza GP-ADF	464,000	156.8	16,000
Koza GP-ADF	272,000	99.5	4,000

[1]No solutions were found for 5-parity using basic GP.

Considering 4 and 5-parity, the GP extended with the size and diversity objectives outperforms both basic GP methods used by Koza (1994) and the basic GP method tested here, both in terms of computational effort and tree size. The Automatically Defined Function (ADF) experiments performed by Koza for these and larger problem sizes perform better. These probably benefit from the inductive bias of ADFs, which favors a modular structure. Therefore, a natural direction for future experiments is to also extend ADFs with size and diversity objectives.

For comparison, we also implemented an evolutionary multi-objective technique that does keep some dominated individuals. It used the number of individuals by which an individual is dominated as a rank, similar to the method described by Fonseca and Fleming (1993). The results were similar in terms of evaluations, but the method keeping strictly non-dominated individuals worked faster, probably due to the calculation of the distance measure. Since this is quadratic in the population size, the small populations of multi-objective save much time (about a factor 7 for 5-parity), which made it preferable.

As a control experiment, we also investigated whether the diversity objective is really required by using only fitness and size as objectives using the algorithm that was described. The individuals found are small (around 10 nodes), but the fitness of the individuals found was well below basic GP, and hence the diversity objective was indeed performing a useful function in the experiments.

8.2 OBTAINING STILL SMALLER SOLUTIONS

Finally, we investigate whether the algorithm is able to find smaller solutions, after finding the first. After the first correct solution is found, we monitor the smallest correct solution. Although the first solution size of 30 was already low compared to other methods, the algorithm rapidly finds smaller correct solutions. The average size drops to 22 within 4,000 additional evaluations, and converges to around 20. The smallest tree (found in 12 out of 30 runs) was 19, i.e. equalling the presumed minimum size. On 4-parity, solutions dropped in size from the initial 68.5 to 50 in about 10,000 evaluations, and to 41 on average when runs were continued longer (85,000 evaluations). In 12 of the 30 runs, minimum size solutions (31 nodes) were found. Using the same method, a minimum size solution to 5-parity (55 nodes) was also found.

The quick convergence to smaller tree sizes shows that at least for the problem at hand, the method is effective at finding small solutions when it is continued running after the first correct solutions have been found, in line with the seeding experiments by Langdon and Nordin (2000).

9 CONCLUSIONS

The paper has discussed using multi-objective methods as a general approach to avoiding bloat in GP and to promoting diversity, which is relevant to evolutionary algorithms in general. Since both of these issues are often implicit goals, a straightforward idea is to make them explicit by adding corresponding objectives. In the experiments that are reported, a size objective rewards smaller trees, and a diversity objective rewards trees that are different from other individuals in the population, as calculated using a distance measure.

Strongly positive results are reported regarding both size control and diversity maintenance. The method is successful in keeping the trees that are visited small without requiring a size limit or a relative weighting of fitness and size. It impressively outperforms basic GP on the 3, 4, and 5-parity problem both with respect to computational effort and tree size. Furthermore, correct solutions of what we believe to be the minimum size have been found for all problem sizes examined, i.e. the even 3, 4, and 5-parity problems.

The effectiveness of the new way of promoting diversity proposed here can be assessed from the following, which concerns the even 3, 4, and 5-parity problems. The multi-objective algorithm that was used only maintains individuals that are not dominated by other individuals found so far, and maintains all such individuals (except those with identical objective vectors). Thus, only non-dominated individuals are selected after each generation, and populations (hence) remained extremely small (6, 16, and 50 on average, respectively). In defiance of this uncommon degree of greediness or elitism, sufficient diversity was achieved to solve these problems efficiently in comparison with basic GP method results both as obtained here and as found in the literature. Control experiments in which the diversity objective was removed (leaving the fitness and size objectives) failed to maintain sufficient diversity, as would be expected.

The approach that was pursued here is to make desired characteristics of search into explicit objectives using multi-objective methods. This method is simple and straightforward and performed well on the problem sizes reported, in that it improved the performance of basic GP on 3 and 4-parity. It solved 5-parity reasonably efficiently, even though basic GP found no solutions on 5-parity. For problem sizes of 6 and larger, basic GP is no longer feasible, and more sophisticated methods must be invoked that make use of modularity, such as Koza's Automatically Defined Functions (1994) or Angeline's GLiB (1992). We expect that the multi-objective approach with size and diversity as objectives that was followed here could also be of value when used in combination with these or other existing methods in evolutionary computation.

Acknowledgements

The authors would like to thank Michiel de Jong, Pablo Funes, Hod Lipson, and Alfonso Renart for useful comments and suggestions concerning this work. Edwin de Jong gratefully acknowledges a Fulbright grant.

References

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). Bloomington, Indiana, USA: Lawrence Erlbaum.

Beasley, D., Bull, D. R., & Martin, R. R. (1993). A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, *1*(2), 101–125.

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, saarbrücken)* (pp. 33–38). Im Stadtwald, Building

44, D-66123 Saarbrücken, Germany: Max-Planck-Institut für Informatik (MPI-I-94-241).

Deb, K., & Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer (Ed.), *Proceedings of the 3rd international conference on genetic algorithms* (pp. 42–50). George Mason University: Morgan Kaufmann.

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

Fleming, P. J., & Pashkevich, A. P. (1985). Computer-aided control system design using a multiobjective optimization approach. In *Proceedings of the iee international conference — control '85* (pp. 174–179). Cambridge, UK.

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 Mateo, California: Morgan Kauffman Publishers.

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

Gathercole, C., & Ross, P. (1996). An adverse interaction between crossover and restricted tree depth 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. 291–296). Stanford University, CA, USA: MIT Press.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley.

Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Genetic algorithms and their applications : Proc. of the second Int. Conf. on Genetic Algorithms* (pp. 41–49). Hillsdale, NJ: Lawrence Erlbaum Assoc.

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

Koza, J. R. (1994). *Genetic programming II: Automatic discovery of reusable programs.* Cambridge, MA: MIT Press.

Langdon, W. B. (1996). Advances in genetic programming 2. In P. J. Angeline & K. Kinnear (Eds.), (p. 395-414). Cambridge, MA: MIT Press. (Chapter 20)

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). Edinburgh: Springer-Verlag.

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). Paris: Springer-Verlag.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms.* Unpublished doctoral dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, USA. (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 (icga95)* (pp. 303–309). Pittsburgh, PA, USA: Morgan Kaufmann.

Nordin, P., & Banzhaf, W. (1995). Complexity compression and evolution. In L. Eshelman (Ed.), *Genetic algorithms: Proceedings of the sixth international conference (icga95)* (pp. 310–317). Pittsburgh, PA, USA: 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, USA: MIT Press.

Rodriguez-Vazquez, 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, USA: 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). Stanford University, CA, USA: MIT Press.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In J. J. Grefenstette (Ed.), *Proceedings of the 1st international conference on genetic algorithms and their applications* (pp. 93–100). Pittsburgh, PA: Lawrence Erlbaum Associates.

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., Foster, J. A., & Dickinson, J. (1996). Code growth 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. 215–223). Stanford University, CA, USA: MIT Press.

Tackett, W. A. (1993). Genetic programming for feature discovery and image discrimination. In S. Forrest (Ed.), *Proceedings of the 5th international conference on genetic algorithms, icga-93* (pp. 303–309). University of Illinois at Urbana-Champaign: Morgan Kaufmann.

Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* Unpublished doctoral dissertation, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

Zissos, D. (1972). *Logic design algorithms.* London: Oxford University Press.