Chapter 7

ANALYSIS OF THE EFFECTS OF ELITISM ON BLOAT IN LINEAR AND TREE-BASED GENETIC PROGRAMMING

Riccardo Poli[1], Nicholas F. McPhee[2] and Leonardo Vanneschi[3]
[1]*Department of Computing and Electronic Systems, University of Essex, UK;* [2]*Division of Science and Mathematics, University of Minnesota, Morris, USA;* [3]*Dipartimento di Informatica, Sistemistica e Comunicazione, University of Milano-Bicocca, Milan, Italy.*

Abstract Elitism, a technique which consists of copying, unchanged, one or more of the most fit individuals from one generation to the next, is widely used in generational evolutionary algorithms, including Genetic Programming (GP). Elitism ensures that the best individuals discovered in a generation (and hence in the whole run) are not lost, and, perhaps even more importantly, are made available to new generations for possible further improvements. In a recent study on the evolution of robustness in GP the average size of best of run individuals was reported to grow more slowly in the presence of elitism. This is an important finding, but no explanation was provided for why this happened nor whether this was a general effect. In this paper we model elitism mathematically and explain how, in general, elitism modulates the dynamics of the mean program size of the population, including both its positive and negative effects on bloat. Experimental results with two GP systems and four different problems corroborate the theory.

Keywords: bloat, elitism, size evolution equation, theory, linear GP, tree-based GP

1. Introduction

Bloat is one of the most widely studied aspects of GP, and many GP implementations use elitism. However, to the best of our knowledge, no substantial investigation of their relationships and dependencies has ever been proposed.

This paper takes significant steps towards filling that gap. We provide a mathematical model of elitism and explain how, in general, elitism modulates the dynamics of the mean program size of a GP population, including its effects on bloat. We also present corroborating experimental results obtained from applying two separate GP systems to four different test problems. We find that

in fact elitism can have a significant impact, often substantially slowing the rate of bloat in the later generations of a run.

In the remainder of this section we look at what is known about both areas (bloat and elitism) and we survey the key the contributions of this paper. In Section 2 we present our theoretical analysis of the effects of elitism. We test our theory in Section 3, where we report empirical results. We then conclude in Section 4.

Bloat

It has long been known that in many conditions GP populations exhibit the phenomenon of *bloat*—a rapid increase in program size not accompanied by any significant corresponding increase in fitness. There has been considerable debate as to the causes of this phenomenon and several qualitative theories have been proposed. For example, the *replication accuracy theory* (McPhee and Miller, 1995) states that GP evolves towards (bloated) representations because this increases the replication accuracy of individuals. The *removal bias theory* (Soule and Foster, 1998) observes that inactive code in a GP tree (code that is not executed, or is executed but its output is then discarded) tends to be low in the tree, residing therefore in smaller-than-average-size subtrees. Crossover events excising inactive subtrees produce offspring with the same fitness as their parents. On average the inserted subtree is bigger than the excised one, so such offspring are bigger than average while retaining the fitness of their parent, leading ultimately to growth in the average program size. Another important theory, the *nature of program search spaces theory* (Langdon and Poli, 1997; Langdon et al., 1999), predicts that above a certain size, the distribution of fitnesses does not vary with size. Since there are more long programs, the number of long programs of a given fitness is greater than the number of short programs of the same fitness. Over time GP samples longer and longer programs simply because there are more of them. Finally, the *crossover bias theory* (Poli et al., 2007; Dignum and Poli, 2007) states that crossover pushes the population towards a particular distribution of program sizes, where small programs have a much higher frequency than longer ones. Because in most problems very small programs have low fitness, programs of above average length have a selective advantage over programs of below average length, and, so, the mean program size increases generation after generation.

In (Poli and McPhee, 2003), the following size evolution equation was derived which provides the expected average size of the programs at the next generation in a GP population under the effects of selection, reproduction, and many types of crossover (including standard sub-tree crossover):

$$E[\mu(t+1)] = \sum_l S(G_l)p(G_l,t). \tag{7.1}$$

Here $\mu(t)$ is the mean size of the programs in the population at generation t, $E[\,]$ is the expectation operator, G_l is the set of all programs of a particular shape (shapes are indexed by l), $S(G_l)$ is the size (number of nodes) of programs of shape l and $p(G_l, t)$ is the probability of selecting programs of shape l from the population in the current generation (indexed by t). This equation is a useful theoretical tool to understand program-size changes, and we will make use of this result in the next section to explain how elitism affects bloat. Additionally, it has been used in the design of bloat control techniques such as (Poli, 2003). However, in itself Equation 7.1 does not explain the reasons for bloat: it only describes what kind of things can happen size-wise in a GP run. That is, the size evolution equation effectively says that bloat can only be the result of an inbalance in the selection probabilities (ultimately fitnesses) for differerent length classes of programs. What it does not do is to say why such differences in selection probability exist. A theory of bloat must explain that.

Important efforts to provide quantitative explanations for bloat were also made in (Banzhaf and Langdon, 2002) and (Rosca, 2003) where representation-less executable models of GP were proposed. In these a population of individuals is evolved, where each individual consists of only three numerical values: the fitness, the size of active code and the size of inactive code. Individuals are manipulated by selection and simple abstractions of mutation (Banzhaf and Langdon, 2002) and crossover (Rosca, 2003). Various effects were observed that were similar to corresponding effects found in GP runs.

Despite these various efforts, it is still unclear how the theories we have briefly reviewed above are related, how they fit within the theoretical framework provided by Equation 7.1 or the executable models of (Banzhaf and Langdon, 2002) and (Rosca, 2003), whether there is a single cause of bloat (in which case, presumably, only one of the theories is correct) or whether there are multiple causes (in which case, perhaps, different theories capture different aspects of the bloat phenomenon).

Elitism

Elitism is a commonly used technique where one or more of the highest-fitness individuals are copied, unchanged, from one generation to the next. The amount of elitism used is often characterised by the *elite fraction*, which is the ratio N/M between the elite size, N, and the population size, M.

Elitism is typically used in generational EAs (including GP) to ensure that the best individuals discovered in a generation (and hence in the whole run) are not lost, and, perhaps even more importantly, are made available for possible further improvements in subsequent generations. In certain conditions (Rudolph, 1994), this property can even provably make an EA a global optimiser (given enough generations).

The use of elitism is very widespread in GP. Many implementations use elites of size 1, although many systems, such as Luke's ECJ (Luke et al., 2006), allow for elites of any size. Indeed, GP has successfully been run with much bigger elites. For example, elite fractions of 1% (Piszcz and Soule, 2006), 2.5% (Voss and Foley, 1999), 4% (Yanai and Iba, 2001), 5% (Sastry et al., 2004), 10% (Topchy and Punch, 2001) and even 20% (Alvarez et al., 2000) have been reported in the literature.

An interesting but not widely reported connection exists between generational systems with elitism and steady state GP systems, where new individuals are immediately inserted in the population as soon as they are created. When new individuals replace the worst individual in the population, a steady state system is equivalent to a generational system with an elite of size $N = M - 1$. So, the theoretical results derived in this paper also apply to this kind of steady state GP system.

Linking Elitism and Bloat

In a recent study of the evolution of robustness in GP, (Piszcz and Soule, 2006) used different degrees of elitism in one of their four sets of experiments (Experiment 3, a symbolic regression problem with $\sin(x)$ as target function). Their goal was to better understand how different levels of elitism affected the robustness of best-of-run individuals. They also noticed, however, that the average size of individuals increased more slowly as the elite fraction was increased from 0 to 1% in steps of 0.2%. A detailed study of elitism was not the objective of that paper, so there was no theoretical explanation for why this happened, or whether this was a general effect beyond the specific regression problem and the levels of elitism considered.

In this paper, we aim to better understand how and why elitism influences the evolution of program size in GP. We show that elitism has predictable and general effects on average program sizes. We model these effects mathematically, showing under which conditions (population size, elite size, selection pressure, etc.) elitism encourages or inhibits program size growth. Finally, we provide experimental results which corroborate the theory, using both a linear GP system and a tree-based GP system on four different problems.

2. Size evolution in the presence of elitism

Equation 7.1 was originally derived under the assumption that the GP system is a pure generational system and that only selection and crossover are acting on the population (Poli and McPhee, 2003). Below we look at how the equation changes if a proportion of the next generation is created by copying some of the individuals in the previous generation. Then we discuss some possible implications of the theory.

Theory

Let M be the population size, and let us assume that our first step in the creation of a new generation is copying N maximally fit individuals over from the previous generation. We will call these individuals the *elite*.

We define $p_e(G_l)$ to be the proportion of individuals of shape l in the elite. Note that the elite can be thought of as the result of applying a round of truncation selection (Thierens and Goldberg, 1994) with truncation ratio N/M to the population. So, the quantity $p_e(G_l, t)$ is effectively the selection probability for programs of shape l under truncation selection. Let us assume that we have no ties across the elite boundary, i.e., there is a unique set of N best individuals.[1] Then we have

$$p_e(G_l, t) = \frac{1}{N} \sum_{x \in G_l} \delta(f(x) \geq f_N(t)), \tag{7.2}$$

where $\delta(\ )$ is an indicator function which returns 1 if its argument is true and 0 otherwise, the sum is over all programs of shape G_l in the population at generation t, $f(x)$ is the fitness of program x and $f_N(t)$ is the fitness of the N-th best-fit individual in the population at generation t.

Let $\mu_1(t)$ denote the average size of the individuals at time t in the first N slots of the new population, i.e., those filled by the elite of the previous generation. Then $E[\mu_1(t+1)] = \sum_l S(G_l) p_e(G_l, t)$, where the summation is extended to all possible shapes. The remaining $M - N$ individuals in the new generations are created in the usual way via standard reproduction, selection and recombination. Although we use these operations to create, in effect, a smaller population (of size $M - N$ instead of M), the statistical features of the individuals in this smaller population, such as mean program size, should remain the same. Thus the average size of the programs in the new generation but not in the elite is $E[\mu_2(t+1)] = \sum_l S(G_l) p(G_l, t)$, as prescribed by Equation 7.1.

In expectation, we have $N \times E[\mu_1(t+1)]$ nodes in the elite and $(M-N) \times E[\mu_2(t+1)]$ nodes in the rest of the new population. The expected program size at the next generation in the presence of elitism, $E[\mu_e(t+1)]$, is then obtained by adding these two groups of nodes and dividing by the population size. That is

$$\begin{aligned} E[\mu_e(t+1)] &= \frac{N}{M} E[\mu_1(t+1)] + \frac{M-N}{M} E[\mu_2(t+1)] \\ &= \frac{N}{M} \sum_l S(G_l)\big(p_e(G_l, t) - p(G_l, t)\big) + \sum_l S(G_l) p(G_l, t). \end{aligned}$$

[1] The handling of ties can potentially have subtle effects on an evolutionary system. See page 98 for more.

Then, using Equation 7.1 we obtain

$$E[\mu_e(t+1)] = \frac{N}{M} \sum_l S(G_l)\big(p_e(G_l,t) - p(G_l,t)\big) + E[\mu(t+1)] \quad (7.3)$$

In other words, elitism modulates the expected mean program size at the next generation. In particular, it can contribute to increasing the mean program size if, on average, $p_e(G_l,t) > p(G_l,t)$ for the larger than average program shapes, or $p_e(G_l,t) < p(G_l,t)$ for the smaller than average program shapes or both. (The opposite happens if the inequalities are inverted.) Furthermore, everything else being equal, the effect is modulated by the elite fraction N/M.

With this theoretical result in hand, we are now in a position to try and understand more about the magnitude and the nature of the effects of elitism in different conditions and phases in a GP run.

Insights and Predictions

Theory is often evaluated on the basis of the number and quality of the insights and predictions it allows. Below we look at whether the effects of elitism on program size should be expected to be large or small. Then we consider what effects one should expect to see in the different phases of a run.

Large or Small Effects. As we noted in Section 1 many GP implementations use elites of size $N = 1$. Since GP is often run with large populations, the elite fraction, N/M, in Equation 7.3 is frequently small (e.g., 0.01–0.1%). So, one might expect to see a limited modulation of elitism on the mean program size evolution.

This may not necessarily be the case, however, because Equation 7.3 represents only the expected behaviour of the system over a single time step. Over multiple time steps, we should expect to see the effects of elitism amplified. Furthermore, elitist GP has successfully been run with small populations (e.g., see (Gathercole and Ross, 1997)) or even tiny populations (e.g., Miller's Cartesian GP (Miller, 1999)) where an elite of size $N = 1$ may actually correspond to an elite fraction of 1% to 25%. Small populations are also often associated with long runs, which we would expect to further amplify any effects of elitism on program size. Finally, as we indicated in Section 1, steady state GP is widely used and is likely to behave somewhat like a generational system with a very large elite size. In such cases, we should expect significant changes in GP's behaviour induced by elitism. Indeed, as we will see in Section 3, the impact of the elite size on size evolution is marked even for elite fractions as small as 1%.

Faster or Slower Growth. So far we have not said whether we expect $E[\mu_e(t+1)]$ to be bigger or smaller than $E[\mu(t+1)]$, i.e., whether elitism

induces faster or slower growth.[2] We cannot give a general answer, since this depends on the correlation between fitness and length present in each particular generation. Below, however, we consider two situations where we can use the theory to make reasonable, general predictions: the early stages of a run, and the late, stagnating stages of a run. Because the details depend so heavily on the particulars of the problem and the run, the arguments below are often fairly qualitative.

Early Stages. GP runs typically start with populations of relatively small programs. In particular, standard initialisation methods such as the ramped-half-and-half and the grow method, tend to produce large proportions of very small programs. As noted in (Dignum and Poli, 2007), solutions are rarely found among very short programs, at least for problems of practical interest. As a result, longer than average programs are often fitter than shorter than average ones in the early stages of a run. Selection will, therefore, promote the use of the longer programs, which in turn leads to a growth in mean program size. Note that this growth cannot really be considered "bloat" since it is often associated with rapid improvements in mean and best fitness (i.e., fitness and length are positively correlated).

What should we expect elitism to do in these conditions? It very much depends on the aggressiveness of the selection scheme used and the fitness diversity in the population.[3] If the better (and typically longer) than average programs only have a slightly higher probability of being selected than the others, then the introduction of elitism could substantially increase the speed at which the mean program size grows. This is because the fraction of the population created by elitism is effectively created via truncation selection, which is typically characterised by a high selection pressure. So, unless one is already using a very strong form of selection, or the initial population is already made up of large programs, elitism should make it easier for GP to move the search towards the longer, more fit programs. Furthermore, it is reasonable to expect that the larger the elite fraction the more pronounced this effect would be. Of course, there is a limit to this, since as the elite size, N, grows, the selection pressure exerted by truncation selection decreases (see Equation 7.3).

[2] Here we are assuming that some form of growth *is* present in our GP runs. The theory presented in Section 2 is valid whatever the size-evolution behaviour of GP. So, it is applicable also to those rare cases where one observes average program size reductions or where average program size remains more or less constant in GP runs. We omit the treatment of these cases here due to space limitations.

[3] Effectively all selection schemes are equivalent if there are no fitness differences between members of the population. Conversely, a relatively weak selection scheme, such as fitness proportionate selection, can become very aggressive if there are extreme fitness differences.

To reiterate, however, if the standard selection scheme used in a GP run already provides strong selection pressure at the beginning of a run,[4] then elitism may actually weaken the selection pressure, thereby slowing down growth in the early stages of a run, and doing so proportionally to the elite fraction. (In the next section we explain that this is also expected to happen in the late stages of a run but for different reasons.)

As we will see in Section 3, both of these beginning-of-run scenarios are encountered in real runs, with elitism increasing growth in most of the cases, but also slowing it down in particular conditions.

Late Stages. There are good reasons to believe that in the late stages of a run, when most of the search momentum has run out and fitness increases have become rare, elitism would slow down bloat, and that the effect would be increased as the elite fraction increases. To see this, let us consider the following argument.

Since the expected average size of the programs outside the elite is the same as the expected average size of the programs in the absence of elitism, i.e., $E[\mu_2(t+1)] = E[\mu(t+1)]$, whether $E[\mu_e(t)]$ is bigger or smaller than $E[\mu(t+1)]$ depends entirely on the expected mean size of the programs in the elite, namely $E[\mu_1(t+1)]$. The question then is, what is the average size of these programs?

This is not easily answered mathematically, but it is clear that, particularly during the late, stagnating phases of a run and when using small populations, not all members of the elite change in one generation. For example, the fittest individual in the elite will remain the same until a superior individual is found. As a consequence the programs in the elite will often come from many generations before the current one.

If bloat is acting on the population, programs generated in previous generations tend to be smaller on average than the programs generated in the current generation. Therefore, in these conditions, an aging elite will tend to have programs of an average size which is significantly smaller than the expected program size at the next generation. In other words, $E[\mu_1(t+1)] < E[\mu(t+1)]$ and, so, $E[\mu_e(t+1)] < E[\mu(t+1)]$. Because of the factor N/M in Equation 7.3, the effect should be modulated by the elite fraction, with larger elite showing a more pronounced reduction in the rates of bloat.

Elitism, Bloat and Sorting Algorithms. Elites are often computed by first sorting the population by fitness and then taking the first N individuals to form

[4]This can happen for a variety of reasons, for instance, because the selection scheme imposes a high selection pressure or because there are extreme fitness differences among the members of the population.

the elite.[5] The result of this sorting is not fully specified in the presence of ties, i.e., when there are individuals with identical fitness. When ties occur within a group of individuals that lay across the elite boundary, then the elite is not uniquely determined. The specific details of the sorting algorithm will dictate which individuals will in fact be copied into the elite.

In this situation, stable sorting algorithms[6] can (depending on other details of the system's implementation) tend to copy the same set of individuals into the elite from one generation to the next. Therefore, successful individuals will tend to persist longer in the elite, thereby increasing the difference between the average size of the individuals in the elite and the average size of the individuals in the rest of the population. If, instead, the sorting algorithm randomises ties, then the average age of the programs in the elite may be less and their sizes correspondingly greater.

That different sorting algorithms may produce different elitist behaviours is important, since there may be conditions under which the effects of the specific choice of sorting algorithm could become significant. We expect, however, these effects to be only visible in small and discrete fitness domains, in particularly flat or "needly" landscapes, or in landscapes with extensive neutrality, where ties are more probable. In the experiments reported below we used *quicksort* (which is not typically stable) and *bubble sort* (which in our case was stable). We did not see substantial differences in behaviour across different domains and algorithms, suggesting that typically any differences are small and merely quantitative, rather than qualitative.

3. Experimental Results

We used two GP systems and two test problems for each. We describe these in the next sub-section. Then we describe the size-evolution behaviours recorded in our experiments with different degrees of elitism. Finally, we briefly discuss the impact of elitism on GP's problem-solving effectiveness.

GP Systems, Problems and Primitive Sets

Linear GP. The first system we used is a linear generational GP system. It initialises the population by repeatedly creating random individuals with lengths uniformly distributed between 1 and 100 primitives. The primitives are drawn randomly and uniformly from each problem's primitive set. The

[5] Obviously when N is fixed and very small (e.g, $N = 1$ or $N = 2$), it is faster to identify the elite by performing a linear scan of the population during which one stores the best N individuals found so far. For larger values of N, however, sorting is more efficient.

[6] Stable sorting algorithms maintain the relative order of items with equal comparison values, e.g., individuals with equal fitness.

system uses fitness proportionate selection and crossover applied with a rate of 100%. Crossover creates offspring by selecting two random crossover points, one in each parent, taking the first part of the first parent and the second part of the second w.r.t. their crossover points. We used populations of size 100 and 1000. For populations of size 100 we performed 500 independent runs. For populations of size 1000 we performed 100 independent runs. Runs lasted 100 generations.

With the linear GP system we used two families of test problems: `Polynomial` and `Lawn-Mower`. `Polynomial` is a symbolic regression problem where the objective is to evolve a function which fits a degree d polynomial of the form $x + x^2 + \cdots + x^d$, for x in the range $[-1, 1]$. Polynomials of this type have been widely used as benchmark problems in the GP literature. In particular we considered degree $d = 6$, and we sampled the polynomial at the 21 equally spaced points $x \in \{-1.0, -0.9, \ldots, 0.9, 1.0\}$. We call the resulting problem instance `Poly-6`. Fitness (to be minimised) was the sum of the absolute differences between target polynomial and the output produced by the program under evaluation over these 21 fitness cases. For this problem we considered a primitive set including the following instructions: `R1 = RIN`, `R2 = RIN`, `R1 = R1 + R2`, `R2 = R1 + R2`, `R1 = R1 * R2`, `R2 = R1 * R2`, and `Swap R1 R2`. The instructions refer to three registers: the input register `RIN` which is loaded with the value of x before a fitness case is evaluated and the two registers `R1` and `R2` which can be used for numerical calculations. `R1` and `R2` are initialised to x and 0, respectively. The output of the program is read from `R1` at the end of its execution.

`Lawn-Mower` is a variant of the classical Lawn Mower problem introduced by Koza in (Koza, 1994). As in the original version of the problem, we are given a square lawn made up of grass tiles. In particular, we considered lawns of size 10×10. The objective is to evolve a program which allows a robotic lawnmower to mow all the grass. In our version of the problem (which differs slightly from Koza's), a robot can perform one of three actions at each time step: move forward one step and mow the tile it lands on (`Mow`), turn left by 90 degrees (`Left`) or turn right by 90 degrees (`Right`). Fitness (to be minimised) was measured by the number of tiles left unmowed at the end of the execution of a program. We limited the number of instructions allowed in a program to a small multiple of the number of tiles available in the lawn (400 in these experiments).

Tree-based GP. For these experiments we adapted Fraser and Weinbrenner's tree-based GPC++ (Fraser and Weinbrenner, 1997) so as to allow the use of elites of any size. We considered two classical problems: the `Ant` problem and the `Even-5 Parity` problem. In both cases we used the standard primitives as described in (Koza, 1992), using populations of size 100 and 1000. Runs lasted

300 generations for `Ant` and 500 generations for `Even-5 Parity`. We used fitness proportional selection and Koza's subtree crossover applied with either 80% or 100% probability. In all conditions we performed 100 independent runs.

Size Evolution Results

We ran all the configurations outlined in the previous section using six different elite sizes: 0%, 1%, 5%, 10%, 30%, and 50% of the population. In this paper we will only report the results with the larger populations (size 1000) due to space limitations. However, the results with populations of size 100 were qualitatively similar.

Figure 7-1 shows the results we obtained with our linear GP system on the Poly-6 symbolic regression problem for the different elite percentages. The main thing to note here is that, while all systems behaved rather similarly in the early generations, elitism seems to induce a reduction in the rate of bloat in the later generations, with the runs using the bigger elites bloating more slowly than those with smaller elites and than "no elite" case. By generation 100, the runs with elite fractions $\leq 10\%$ had average sizes that were nearly twice as large as those using 50% elitism, for example. In the early generations we don't see a quicker growth in the presence of elitism (as was suggested as a possibility in Section 2) because in symbolic regression problems of the kind considered here, fitness values across the population have very high variance in the early generations, making fitness proportional selection more aggressive than truncation selection. Essentially the same can be seen for the `Lawn-Mower` problem, as illustrated in Figure 7-2, where initially there is a strong correlation between length and fitness.

For tree-based GP, however, we see the two phases postulated in Section 2, both in the `Ant` problem (Figures 7-3 and 7-5) and in the `Even-5 Parity` problem (Figures 7-4 and 7-6). In all cases we see that, initially, the stronger the elitism the faster the growth in program size. The picture is, however, reversed when the population settles into a stagnating phase. There, as predicted, the rate of bloat is very markedly reduced by increasing the elitism in the system, with elite fractions of 30% and 50% essentially behaving almost identically. This is consistent with our prediction that there would be a limit to the effects of additional elitism. Note also that elitism affects bloat in essentially the same way whether or not the reproduction operator is used to create a proportion of the individuals in the new generation as one can see by comparing Figure 7-3 with Figure 7-5 and Figure 7-4 with Figure 7-6.

Only one case *appears* to deviate from what we expected. This is represented by the runs of the `Even-5 Parity` problem where no elitism was used. These runs (see Figures 7-4 and 7-6) appear not to bloat at all, contrary to our

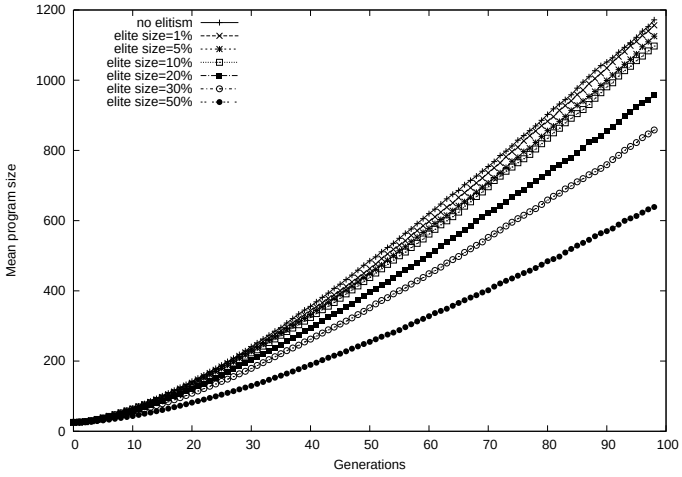

Figure 7-1. Plots of the average size of programs vs generation in a linear GP system with populations of size 1000 solving the Poly-6 problem for different elite fractions. Plots are averages over 100 independent runs.

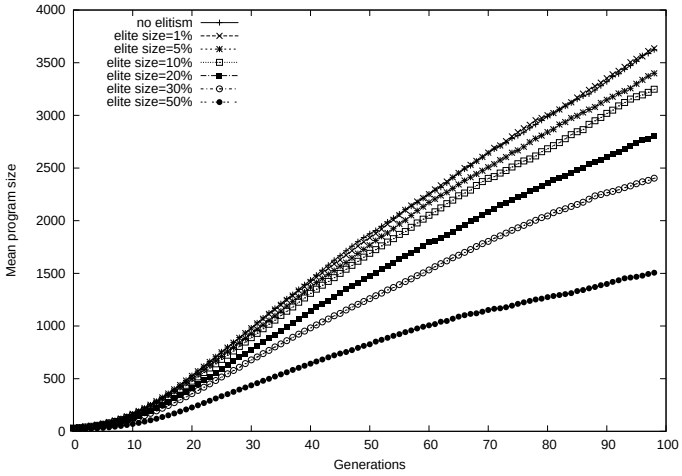

Figure 7-2. Plots of the average size of programs vs generation in a linear GP system with populations of size 1000 solving the lawnmower problem for different elite fractions. Plots are averages over 100 independent runs.

expectation that the no-elitism case would be, in general, the fastest growing. The reason for this is very simple. In this problem, with a population of 1,000 individuals, most (if not all) programs in the first generation satisfy 16 out of the 32 fitness cases (see for instance (Langdon and Poli, 2002; Vanneschi, 2004; Vanneschi et al., 2006)). When an improvement is found, it typically is a program that satisfies one extra case. With fitness proportionate selection this

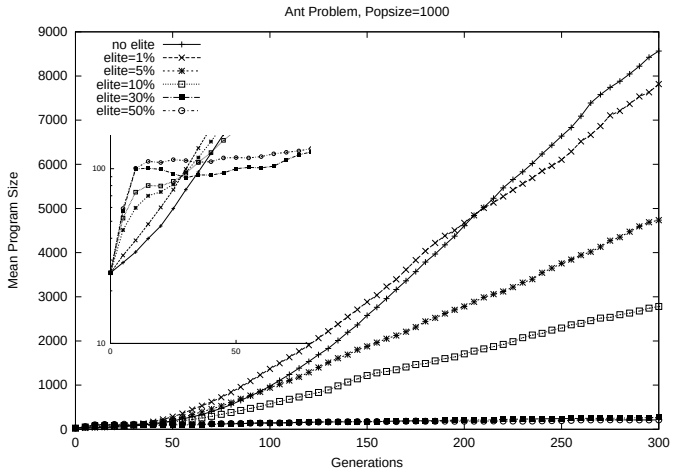

Figure 7-3. Plots of the average size of programs vs generation in a tree-based GP system with populations of size 1000 solving the `Ant` problem for different elite fractions and a crossover rate of 100%. Plots are averages over 100 independent runs. The first 50 generations are also shown in the inset with a log scale. Note that the values for 30% elite and 50% elite were almost identical, and the 30% values tend to obscure the 50% values in this plot.

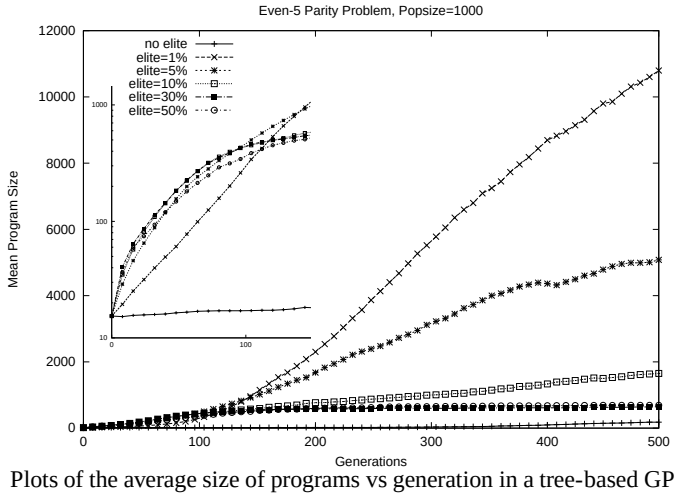

Figure 7-4. Plots of the average size of programs vs generation in a tree-based GP system with populations of size 1000 solving the `Even-5 Parity` problem for different elite fractions and a crossover rate of 100%. Plots are averages over 100 independent runs. The first 200 generations are also shown in the inset with a log scale. Note that the values for 30% elite and 50% elite were almost identical, and the 30% values tend to obscure the 50% values in this plot.

produces a very small increase in the selection probability for such a program. Since we use 100% crossover, without elitism this improved program is very likely to be destroyed at the next generation. Its offspring (which statistically might be slightly longer then average) are likely to have the same fitness as the

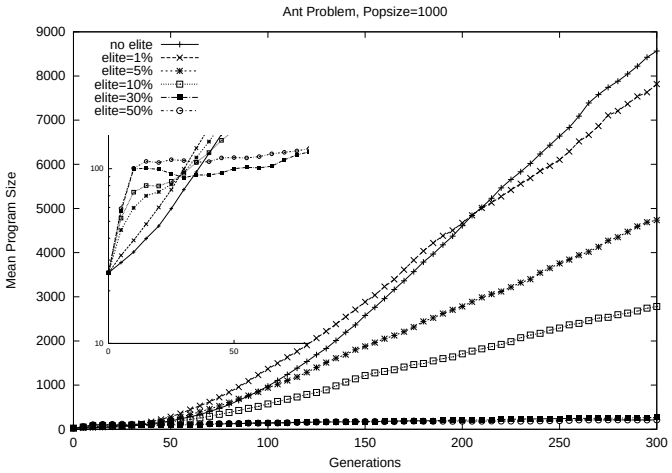

Figure 7-5. Plots of the average size of programs vs generation in a tree-based GP system with populations of size 1000 solving the `Ant` problem for different elite fractions and a crossover rate of 80%. Plots are averages over 100 independent runs. The first 50 generations are also shown in the inset with a log scale.

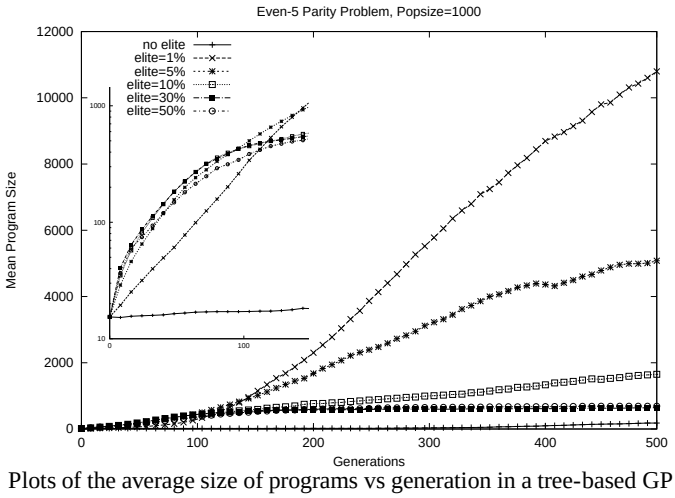

Figure 7-6. Plots of the average size of programs vs generation in a tree-based GP system with populations of size 1000 solving the `Even-5 Parity` problem for different elite fractions and a crossover rate of 80%. Plots are averages over 100 independent runs. The first 200 generations are also shown in the inset with a log scale.

rest of the population. So, they are not likely to be reselected. In short, there is very little correlation between size and fitness, so many generations are needed before significant progress can be made on the problem and bloat can really set in. Eventually it does so; notice the small rise in the very bottom right corner of the figure, which corresponds to an average size of 67.1 nodes. For populations of 100, the final average size was 160.9. In both cases the average size in the

initial population was 15.3. We conjecture that if given many more generations populations without elitism would eventually overtake those with elitism.

Fitness and Success Rates

In order to convey a fuller picture of the effects of elitism, we include some information on fitness and success rates for the various combinations studied here.

That the `Even-5 Parity` problem cannot be solved with populations of 1000 individuals without elitism is very apparent in the fitness plot shown in Figure 7-7 (bottom). In the absence of elitism the mean fitness of the population (not shown) improves extremely slowly from its initial value of 16. Furthermore, the average over the 100 runs of the best fitness in the population (which, without elitism, cannot be retained) does not improve from its initial value of around 14 and, in fact, slightly worsens over time (Figure 7-7, bottom). As soon as elitism is added, however, there is something like a phase transition, with the larger elite fractions leading to larger advantages both in terms of mean fitness and best fitness. This is consistent with (Vanneschi et al., 2006) where this problem was shown to present a large neutral network with fitness 16 from which one can reach individuals of high fitness only by following very narrow paths. Each path is formed by a chain of neighbouring individuals that are one mutation apart. Thus, the only way to escape from the network is to ensure individuals of fitness higher than 16 are kept in the population and progressively further improved. This is why, effectively, in the parity problem elitism not only provides smaller solutions, but it also makes the search for solutions easier.

Figure 7-7 also shows the best fitness plots for the `Ant` problem with different levels of elitism. Because of the nature of the problem and the higher granularity of the fitness function, here the absence of elitism does not produce catastrophic results. Elitism, however, does appear to be advantageous up to a point both in terms of mean population fitness and in terms of best fitness in each generation. Best results are obtained with small amounts of elitism, e.g., 1%.[7] Increasing the elite fraction slightly decreases performance. So, for the `Ant` problem one can trade accuracy for solution size by changing the elite size. In this sense, elitism acts as a typical anti-bloat method.

A similar kind of behaviour was also observed with the linear GP system when solving the `Poly-6` problem. As shown in Table 7-1, the success rate tended to decrease slightly at the highest elite fractions (50%, and possibly also 30%), although it its not clear whether there is any significant difference with smaller elites. The table also shows the mean best fitness (error) obtained

[7]This does not mean that the traditional $N = 1$ is best. The smallest value of the elite fraction in our runs was 1%, which corresponds, for populations of 1000, to an elite of $N = 10$ individuals.

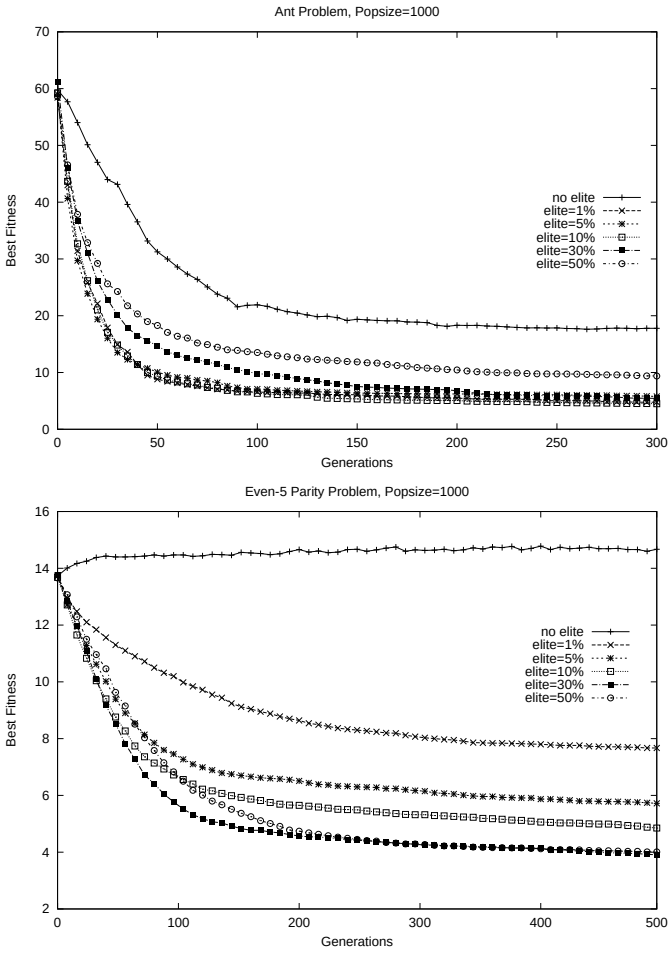

Figure 7-7. Plots of the best fitness of programs vs generation in a tree-based GP system with populations of size 1000 solving the ANT (top) and Even-5 Parity (bottom) problems for different elite fractions and a crossover rate of 100%. Plots are averages over 100 independent runs.

in different configurations. This is substantially unaffected by the elite size, except, possibly, at the highest elite fraction. The success rates and fitnesses for the (easier) `Lawn-Mower` problem (not reported) showed no variation with the elite size.

4. Conclusions

Elitism is widely used in GP, particularly when the population is not very large and there is a greater chance of losing the best individual. Surprisingly, despite its widespread use, to the best of our knowledge there is very little

Table 7-1. Success rate and mean best fitness in runs with a linear GP system solving the poly-6 problem for different degrees of elitism. Figures were estimated based on 500 independent runs for population size 100 and 100 independent runs for population size 1000.

Population Size	*Elite Fraction*	*Success Rate*	*Mean Best Fitness*
100	No Elite	0.41	0.64
100	1%	0.34	0.60
100	5%	0.39	0.63
100	10%	0.38	0.62
100	30%	0.31	0.57
100	50%	0.24	0.51
1000	No Elite	0.71	0.85
1000	1%	0.76	0.88
1000	5%	0.65	0.82
1000	10%	0.78	0.89
1000	30%	0.69	0.84
1000	50%	0.53	0.75

literature and no theory on the effects of elitism in GP. The only case where different degrees of elitism were tested was one set of experiments (out of four) with one problem in the work by Piszcz and Soule (Piszcz and Soule, 2006), and the focus there was on the evolution of robustness in GP rather than on explaining the effects of elitism on bloat.

In this paper we provided a general mathematical model of the effects of elitism on program size evolution, a series of predictions on the qualitative behaviour of GP based on this model and a plethora of experimental results corroborating both the model and its predictions.

Our results make it clear that elitism can have a powerful effect on bloat, generally reducing the level of bloat in the later generations, with larger elite sizes typically controlling bloat more strongly. Elitism also improved performance in most cases. So, in general the use of elitism (possibly with bigger elites than the single individual used routinely) brings significant advantages to GP.

References

Alvarez, Luis F., Toropov, Vassili V., Hughes, David C., and Ashour, Ashraf F. (2000). Approximation model building using genetic programming methodology: applications. In Baranger, Thouraya and van Keulen, Fred, editors, *Second ISSMO/AIAA Internet Conference on Approximations and Fast Reanalysis in Engineering Optimization.*

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

Dignum, S. and Poli, R. (2007). Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In et al., Dirk Thierens, editor, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1588–1595, London. ACM Press.

Fraser, Adam and Weinbrenner, Thomas (1993, 1994, 1997). Gpc++ genetic programming c++class library.

Gathercole, C. and Ross, P. (1997). Small populations over many generations can beat large populations over few generations in genetic programming. In John R. Koza *et al.*, editor, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 111–118, San Francisco, CA, USA. Morgan Kaufmann.

Koza, John R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

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

Langdon, W. B. and Poli, R. (1997). Fitness causes bloat. In Chawdhry, P. K., Roy, R., and Pant, R. K., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 13–22. Springer-Verlag London.

Langdon, W. B. and Poli, R. (2002). *Foundations of Genetic Programming*. Springer, Berlin, Heidelberg, New York, Berlin.

Langdon, William B., Soule, Terry, Poli, Riccardo, and Foster, James A. (1999). The evolution of size and shape. In Spector, Lee, Langdon, William B., O'Reilly, Una-May, and Angeline, Peter J., editors, *Advances in Genetic Programming 3*, chapter 8, pages 163–190. MIT Press, Cambridge, MA, USA.

Luke, S., Panait, L., Balan, G., Paus, S., Skolicki, Z., Bassett, J., Hubley, R., and Chircop, A. (2006). ECJ, a Java-based Evolutionary Computation Research System. Downloadable versions and documentation can be found at the following url: http://cs.gmu.edu/ eclab/projects/ecj/.

McPhee, Nicholas Freitag and Miller, Justin Darwin (1995). Accurate replication in genetic programming. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 303–309, Pittsburgh, PA, USA. Morgan Kaufmann.

Miller, Julian F. (1999). An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In Banzhaf, Wolfgang, Daida, Jason, Eiben, Agoston E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1135–1142, Orlando, Florida, USA. Morgan Kaufmann.

Piszcz, Alan and Soule, Terence (2006). Dynamics of evolutionary robustness. In Keijzer *et al.*, M., editor, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 871–878, Seattle, Washington, USA. ACM Press.

Poli, Riccardo (2003). A simple but theoretically-motivated method to control bloat in genetic programming. In Ryan, Conor, Soule, Terence, Keijzer, Maarten, Tsang, Edward, Poli, Riccardo, and Costa, Ernesto, editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 204–217, Essex. Springer-Verlag.

Poli, Riccardo, Langdon, William B., and Dignum, Stephen (2007). On the limiting distribution of program sizes in tree-based genetic programming. In Ebner *et al.*, M., editor, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 193–204, Valencia, Spain. Springer.

Poli, Riccardo and McPhee, Nicholas Freitag (2003). General schema theory for genetic programming with subtree-swapping crossover: Part II. *Evolutionary Computation*, 11(2):169–206.

Rosca, Justinian (2003). A probabilistic model of size drift. In Riolo, Rick L. and Worzel, Bill, editors, *Genetic Programming Theory and Practice*, chapter 8, pages 119–136. Kluwer.

Rudolph, Günter (1994). Convergence analysis of canonical genetic algorithm. *IEEE Transactions on Neural Networks*, 5(1):96–101.

Sastry, Kumara, O'Reilly, Una-May, and Goldberg, David E. (2004). Population sizing for genetic programming based on decision making. In O'Reilly, Una-May, Yu, Tina, Riolo, Rick L., and Worzel, Bill, editors, *Genetic Programming Theory and Practice II*, chapter 4, pages 49–65. Springer, Ann Arbor.

Soule, Terence and Foster, James A. (1998). Removal bias: a new cause of code growth in tree based evolutionary programming. In *1998 IEEE International Conference on Evolutionary Computation*, pages 781–186, Anchorage, Alaska, USA. IEEE Press.

Thierens, Dirk and Goldberg, David (1994). Convergence models of genetic algorithm selection schemes. In Davidor, Yuval, Schwefel, Hans-Paul, and Manner, Reinhard, editors, *Parallel Problem Solving from Nature – PPSN III*, number 866 in Lecture Notes in Computer Science, pages 119–129, Jerusalem. Springer-Verlag.

Topchy, A. and Punch, W. F. (2001). Faster genetic programming based on local gradient search of numeric leaf values. In Spector, L., *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO '01*, pages 155–162, San Francisco, CA. Morgan Kaufmann.

Vanneschi, L. (2004). *Theory and Practice for Efficient Genetic Programming*. Ph.D. thesis, Faculty of Sciences, University of Lausanne, Switzerland.

Vanneschi, L., Pirola, Y., Collard, P., Tomassini, M., Verel, S., and Mauri, G. (2006). A quantitative study of neutrality in GP boolean landscapes. In M. Keijzer *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'06*, volume 1, pages 895–902. ACM Press.

Voss, Mark S. and Foley, Christopher M. (1999). Evolutionary algorithm for structural optimization. In Banzhaf, Wolfgang, Daida, Jason, Eiben, Agoston E., Garzon, Max H., Honavar, Vasant, Jakiela, Mark, and Smith, Robert E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 678–685, Orlando, Florida, USA. Morgan Kaufmann.

Yanai, Kohsuke and Iba, Hitoshi (2001). Multi-agent robot learning by means of genetic programming : Solving an escape problem. In Liu, Yong, Tanaka, Kiyoshi, Iwata, Masaya, Higuchi, Tetsuya, and Yasunaga, Moritoshi, editors, *Evolvable Systems: From Biology to Hardware: 4th International Conference, ICES 2001*, volume 2210 of *LNCS*, pages 192–203, Tokyo, Japan. Springer-Verlag.