The Impact of Population Size on Code Growth in GP: Analysis and Empirical Validation

Riccardo Poli
Department of Computing and Electronic Systems
University of Essex
Colchester, UK
rpoli@essex.ac.uk

Nicholas Freitag McPhee
Division of Science and Mathematics
Univ. of Minnesota, Morris
Morris, MN, USA
mcphee@morris.umn.edu

Leonardo Vanneschi
Department of Informatics, Systems and Communication
University of Milano-Bicocca,
Milan, Italy
vanneschi@disco.unimib.it

ABSTRACT

The crossover bias theory for bloat [18] is a recent result which predicts that bloat is caused by the sampling of short, unfit programs. This theory is clear and simple, but it has some weaknesses: (1) it implicitly assumes that the population is large enough to allow sampling of all relevant program sizes (although it does explain what to expect in the many practical cases where this is not true, e.g., because the population is small); (2) it does not explain what is meant by its assumption that short programs are unfit.

In this paper we discuss these weaknesses and propose a refined version of the crossover bias theory that clarifies the relationship between bloat and finite populations, and explains what features of the fitness landscape cause bloat to occur. The theory, in particular, predicts that smaller populations will bloat more slowly than larger ones. Additionally, the theory predicts that bloat will only be observed in problems where short programs are less fit than longer ones when looking at samples created by fitness-based importance sampling, i.e. samplings of the search space in which fitter programs have a higher probability of being sampled (e.g., the Metropolis-Hastings method). Experiments with two classical GP benchmarks fully corroborate the theory.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming

General Terms

Algorithms, Performance

Keywords

Genetic Programming, Bloat, Population Size

1. INTRODUCTION

A rapid increase in program size is often observed after the initial phases of Genetic Programming (GP) runs. This phenomenon, which is generally referred to as *bloat*, has been widely studied (see for instance [1, 13, 14, 22] [19, Section 11.3]). Properly speaking, bloat corresponds to increases in program sizes that are not connected to improved fitness. In many problems there may be phases of growth where fitness also improves, so not all forms of growth can be termed bloat. For this reason, we will often also use terms such as *program size growth* or *code growth* when this language is more precise. In addition, we will often use the term "length" instead of "program size" to refer to the number of nodes in a program, since this makes it less likely to confuse it with the population *size*, which is a term used throughout the paper frequently. Due to space limitations we cannot provide a complete literature review on bloat and related issues. We will focus, instead, on a subset of results which are particularly relevant to the work presented here.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

1.1 Limiting Distribution of Fitness in the GP Search Space

The characterization of the space of computer programs explored by GP has been an important topic of theoretical research [11]. A key contribution was a series of theoretical results showing that the distribution of functionality of non Turing-complete programs approaches a limit as program length increases. That is, although the number of programs of a particular length grows exponentially with length, beyond a certain threshold the fraction of programs implementing any particular functionality is effectively constant. This implies that, as we look at bigger and bigger programs, the distribution of fitnesses approaches a limit.

There is a very substantial body of empirical evidence indicating that this happens in a variety of systems. There are also mathematical proofs of these convergence results for two important forms of programs: Lisp (tree-like) S-expressions (without side effects) and machine code programs without loops [6, 7, 8, 9, 10, 11]. Also, similar results were derived for: a) cyclic CPU (increment, decrement and NOP instructions), b) bit flip computer, (flip bit and NOP), c) any non-reversible computer, d) any reversible computer, e) CCNOT (Toffoli gate) computer, f) quantum computers, g) the 'average' computer and h) AND, NAND, OR, NOR expressions. In addition, recently, these results have been extended to Turing complete machine code programs [12, 17].

1.2 Crossover Bias Theory for Bloat

One of the most recent explanations for bloat is the so called *crossover bias theory* [18, 3]. On average, each application of subtree crossover removes as much genetic material as it inserts. So, crossover in itself does not produce growth or shrinkage. However, while the *mean* program size is unaffected, *higher moments* of the distribution are affected. In particular [18, 3] showed that crossover pushes the population towards the following distribution

of program sizes, called a *Lagrange distribution of the second kind*:

$$\Pr\{n\} = (1 - ap_a)\binom{an+1}{n}(1-p_a)^{(a-1)n+1}p_a^n \qquad (1)$$

Here n is the number of internal nodes in a tree, $\Pr\{n\}$ is the proportion of the population having length n, p_a is a constant and a is the average arity of the primitives in the function set. The parameter p_a is a function of a and the mean program size μ, and has a role similar to the success rate in a binomial distribution. When $a > 1$, this distribution vaguely resembles the exponential distribution but with a much fatter tail. Because of the distribution's particular profile, short programs have a much higher frequency than longer ones. For example, crossover generates a very high proportion of single-node individuals. However, in virtually all problems of practical interest, very short programs have no chance of solving the problem. As a result, programs of above average length have a selective advantage over programs of below average length. Consequently, the mean program size increases. Note that the distribution in Equation (1) tends to become flatter as μ grows. So, the sampling of short programs becomes less and less frequent as the mean program size increases.

1.3 Size Evolution Equation

Poli and McPhee [16, 20] developed an exact formalisation of the dynamics of average program size for genetic programming:

$$E[\mu(t+1)] = \sum_l S(G_l)p(G_l,t), \qquad (2)$$

where $\mu(t+1)$ is the mean size of the programs in the population at the next generation, $E[\,]$ 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 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). We will call this equation the *size evolution equation*. With simple manipulations Equation (2) can be rewritten in terms of length-classes, rather than tree shapes, obtaining

$$E[\mu(t+1)] = \sum_\ell \ell p(\ell,t) \qquad (3)$$

where the index ℓ ranges over all program sizes and

$$p(\ell,t) = \sum_{l:S(G_l)=\ell} p(G_l,t). \qquad (4)$$

Note that Equation (2) does not directly explain bloat. It only constrains what can happen size-wise in GP populations. The equation predicts that under the effects of crossover the mean program size evolves as if selection only was acting on the population.

1.4 Contributions of this paper

This paper is structured as follows. In Section 2 we combine and refine the theoretical results reviewed above, developing a more precise version of the crossover bias theory of bloat (Section 2.1); we then illustrate the ideas with a simple qualitative example (Section 2.2). In Section 3 we then introduce the two benchmark problems we use to gather the empirical evidence used to test the new theory (Section 3.1), paying particular attention to certain aspects of their fitness landscape which are important in relation to the theory and bloat (Section 3.2). We report experimental results in Section 4, where we look at how the growth in the mean, median and extreme values of the program size distribution are affected by the population size generation after generation. We draw our conclusions in Section 5.

2. BLOAT AND POPULATION SIZE

With the previous background material in place, we are now in a position to suggest why (and when) large populations should bloat faster than small ones. We do this by first combining (in Section 2.1) the theoretical evidence from Sections 1.1 and 1.3. We then provide an informal thought experiment in Section 2.2 to further illustrate the ideas.

2.1 Combining and Refining the Evidence

Based on the size evolution equation (Section 1.3) and Langdon's theory on limiting distribution of fitness (Section 1.1), if all programs in the population are sufficiently large, there should be no bloat on average. However, in practice this situation is not easily reached nor easily maintained because subtree crossover tends to always produce a mix of program lengths as indicated by the crossover bias theory (Section 1.2). Thus while selection will attempt to modify the distribution based on fitness, crossover will work to distribute program lengths according to a Lagrange distribution of the second kind [18] which heavily over-samples the short programs. Because of its bias, crossover tends to always produce a significant proportion of short programs.[1] Consequently, the tendency to bloat should continue even as the programs grow into the region where the distribution of fitnesses is approximately constant.

Earlier we saw that the crossover bias theory assumes that, whenever short programs cannot solve problems and crossover samples such programs often, there is a persistent selective pressure against short programs in the population. This, in turn, produces a selective advantage for the longer programs, ultimately causing bloat.

The theory does not fully clarify in what sense short programs would have below average fitness. One could mean, for example, that when doing an unbiased random sampling of the search space one finds that short programs are on average less fit than large ones. Alternatively, one could mean the short programs are less fit among those programs explored by selection and recombination. There is a distinction between the two because, although GP does random or near-random sampling in the first few generations of a run, as soon as selection can get something to act upon, it will rapidly focus the search on the higher-fitness elements of each length class. In this case, the shape and thickness of the upper tail of the fitness distribution associated with each length class is more important than its mean. We postulate that while the differences in the means of the fitness distribution associated with different length classes may influence the transient behaviour of the dynamics of the mean program length, it is the differences in the tails of the distribution which influence the late stages of runs.

To sum up, it is only when the fitness of the best short programs observed during "genetic sampling" (as opposed to, e.g., random sampling) is inferior to the fitness of the best long programs that the crossover bias theory for bloat is applicable.

A second important clarification is necessary. The crossover bias theory (Section 1.2) assumes that the population is "sufficiently large", i.e., large enough that all length classes are sampled (or at least that the short, sub-quality programs are sampled consistently). Is the theory still applicable if the population is not "sufficiently large"? We think it is, at least in a refined form. In particular, it is clear that the smaller the population and/or the larger the average

[1] One possible exception is the case where only arity 1 functions are used and the mean program length is very large. With only arity one (linear structures) the Lagrange distribution collapses to a discrete gamma distribution. When the mean size is very large, this distribution has less and less mass concentrated on the very short programs. The mean size must be quite large, however, so our assumptions should hold in most cases even for linear structures.

program length, the less likely it is that one will sample the short programs. It is even possible that with very small, highly bloated populations no sample is allocated to the short and sub-average-quality programs in one generation, in which case we should not expect bloat in that generation. So, it stands to reason that, because large populations sample the short programs more consistently, bloat should be faster in larger populations than in smaller ones.

In the following we will call the extension of the crossover bias theory with the previous two ideas the *revised crossover bias theory for bloat* to avoid any confusion with the original. The key aim of the remainder of this paper is to corroborate this theory.

2.2 A Thought Experiment

Let us consider a situation where the population clearly is *not* "sufficiently large" to ensure that all length classes are sampled (or to at least ensure that the short, sub-quality programs are sampled consistently), which is one of the main assumptions of the crossover bias theory (Section 1.2). Assume, for instance, that we have a population of 1,000 individuals. Further assume that, for some reason, this includes very large programs, say 15,000 nodes on average. Finally, assume that 15,000 is way beyond the threshold where the functionality of programs reaches a limit according to Langdon's theory [11]. In this population there is no real advantage or disadvantage (in expectation) in being moderately above or below the mean size of 15,000 nodes.

If we now apply crossover, we know that in principle it can produce programs of sizes from 1 to almost twice the size of the largest individual in the population. If, for example, the largest individual has 200,000 nodes, then we could get anything from 1 to 399,999 nodes. Clearly, however, not all length classes will be equally probable. Exactly what (theoretical) distribution one would get depends in complicated ways on the distribution you had before crossover. Further, in practice we will only draw 1,000 samples from this distribution, so we should expect a size histogram that only vaguely resembles the theoretical distribution. However, we know that there must be *many more programs* that are *shorter* than the 15,000 average, than there are programs that are longer (otherwise we would see a change in the average program size). So, for the sake of argument, assume that 750 programs are below average length and 250 are above average length. So, we have 750 samples to allocate to 15,000 length classes. So, there is *no hope* of sampling all classes. If samples were allocated uniformly between 1 to 15,000 (but we know this is not true) we would have 1 chance in 20 to sample a particular length class. So, there would be a significant probability of *not* having very short programs (those that cannot solve the problem) in that particular generation. This would then eliminate one of the key forces driving up the average tree size, at least in that generation.

Naturally, it may eventually happen that one generation would get unlucky and again sample a short program, thereby giving a selective advantage to the longer programs and producing an increase in average size at that generation. It is clear, however, that every time the mean program size goes up, we create more length classes below the mean, and, so, in future generations it becomes a little harder to sample the short length classes. For this reason, it is reasonable to think that sooner or later the growth rate should slow down.

Now, imagine a population of 100,000 individuals, again with average size 15,000. Assume that 75,000 of these are below average length (same proportion as for previous example). We see that now, if we apply crossover, we have 5 samples per length class. So, we are *practically certain* to sample programs in all the very short length classes, many of which will be unfit. Therefore we should expect to see size growth at each generation.

According to the limiting distribution theory in Section 1.1, sooner or later, the process must limit itself even with the larger population. For example, if the mean program length eventually reached 1,500,000 nodes, we would again only be able to sample each length class (below the mean length) with a probability 1/20.

As a consequence, while we would expect bloat to eventually slow down for all finite populations, we would expect it to continue for longer in larger populations than in the smaller ones. The main objective of this paper is to corroborate this hypothesis with empirical evidence.

3. TEST PROBLEMS, FITNESS LANDSCAPES AND RANDOM SAMPLING

In this section we present the test problems used in the experiments presented here (Section 3.1) and we provide a characterisation of their fitness landscape (Section 3.2).

3.1 Test Problems

Because the test problems given below are well known and established GP benchmarks, their descriptions are very brief. See [4], for example, for a more detailed description of both problems.

3.1.1 Artificial Ant

In this problem, an artificial ant is placed on a 32×32 toroidal grid. Some of the grid cells contain food pellets. The pattern of food follows the so called "Santa Fe Trail" [4]. The goal is to find a navigation strategy for the ant that maximizes its food intake. We use the same set of functions and terminals as in [4], i.e., $\mathcal{F} = \{\texttt{IfFoodAhead}, \texttt{Progn2}, \texttt{Progn3}\}$ and $\mathcal{T} = \{\texttt{Right}, \texttt{Left}, \texttt{Move}\}$, respectively. We use the total number of food pellets lying on the trail (89) minus the amount of food eaten by the ant during its exploration for the raw fitness. This turns the problem into a minimization problem.

3.1.2 Symbolic Regression

This problem aims to find a program that matches a given equation at several test points. Following [15], we use $f(x) = sin(x)$ as the target function, and our set of test points is 100 equidistant points in the range $[0, 1]$ (giving us 100 fitness cases). For this problem, the set of functions used for GP individuals is: $\mathcal{F} = \{*, \%, +, -\}$, where % is a protected division which returns 1 when the divisor is 0. We define the fitness as the root mean squared error between outputs and targets. Consequently, the lower the fitness the better the solution, implying that this problem is also a minimization problem.

3.2 Fitness Landscape Features of the Test Problems

We had two motivations for choosing these particular test problems. The first motivation for choosing these two problems is that they are well known GP benchmarks, and much is known about these problems (see, for instance, [4, 11, 23, 19]).

The second and, maybe, more important motivation is that the two problems show different characteristics in their fitness landscapes, or, more specifically, in their length-class fitness distributions. These differences provide a more extensive test of our first conjectured refinement for the crossover bias theory, namely that the differences in tails of the length-class fitness distribution is what matters in relation to bloat in the long run.

To see these differences, we will sample the search space for the two problems using two different techniques: the random creation

of trees using the Ramped Half and Half algorithm [5], and sampling using the Metropolis-Hastings algorithm [2]. In the following subsections we explain these sampling algorithms and their results on the two test problems.

3.2.1 Random Sampling with Ramped Half and Half

The Ramped Half and Half algorithm constructs random individuals using the Full method (which creates full trees of a given size) half of the times and the Grow method (which creates trees of more varied shapes) the other half of the times. This is done using a range of depth limits (hence the term 'ramped') to help ensure that we generate trees having a variety of sizes and shapes.

Figure 1 reports the average fitness of a sample of 4000 individuals randomly generated with the Ramped Half and Half algorithm for a set of possible values of the maximum tree depth parameter.[2] As this figure clearly shows, for the artificial ant (Figure 1(a)) small sampled individuals have poor average (static) fitness values, and fitness improves as larger and larger values of the maximum tree depth are used. This result is consistent with a similar result reported in [11] and corroborates the applicability of the (refined) crossover bias theory.

Interestingly, however, for the symbolic regression problem (Figure 1(b)) we see the opposite behavior: the average (static) fitness of short programs is higher than the average fitness for large programs sampled by the Ramped Half and Half algorithm. From this result, we might expect that the crossover bias theory of bloat is inapplicable to this benchmark, or even that instead of bloat we should observe shrinking (as one could easily argue using an exact "mirror" argument to the crossover bias theory). As discussed below, however, GP doesn't randomly sample the population, and when we correct for this, we find that the distributions for the two problems look much more similar.

3.2.2 Fitness-based Random Sampling with Metropolis-Hastings

The limitations of using random samples such as those generated by the Ramped Half-and-Half algorithm in investigations of the features of fitness landscapes are discussed at length in [23, 24], where it was suggested that important sampling techniques such as the Metropolis-Hastings method [2] are much more effective tools. The main reason for this is that GP doesn't sample randomly – mechanisms like selection deliberately and heavily bias the sampling of the search space. As a result, methods like Metropolis-Hastings tend to sample sets of programs with characteristics more like those in GP populations.

The Metropolis-Hastings algorithm generates a sample of individuals $\{i_1, i_2, ..., i_N\}$. Individual i_1 is randomly chosen. In our case, we used the grow method to generate it. In our study, subsequent individuals i_j, for $2 \leq j \leq N$, are created by generating 1000 random individuals (all of them with the grow method), letting i_j be the first whose fitness is better than i_{j-1}. If none of these 1000 random individuals is better than i_{j-1}, we set $i_j = i_{j-1}$. The pseudo-code of the general Metropolis-Hastings algorithm can be found in [23] at page 131. The version used in the present study uses a particularly strong acceptance criterion for the next solution i_j to be inserted in the sample. In general, a new solution can be accepted also if its fitness is worse than the previously accepted one, with a given probability. In this work, however, that probability has been set to zero and only better individuals have been accepted, thus simulating a strong selection pressure. Of course, we know that this selection is not the same as performed by GP, but we believe that this is a reasonable approximation. As with the Ramped Half and Half sampling from Section 3.2.1, we generate a final sample set of size $N = 4000$.

Figure 2 shows the results of the Metropolis-Hastings sampling on the two benchmark problems; compare this with the plots in Figure 1. Looking at the ant problem, for example, we see that short programs are still less fit than longer ones, and that in fact the difference in fitness has increased enormously (note the different y-axis scales in Figure 1(a) and 2(a)). Furthermore, we can see how, above a certain depth, all depths provide the same average fitness among the accepted samples of that depth. This further corroborates the applicability of our refined crossover bias theory. Turning to the symbolic regression problem, we see under the fitness-based sampling of the Metropolis-Hastings method, the tails of the length-class fitness distributions behave entirely differently from the samples generated using Ramped Half-and-Half. Here average fitness decreases with depth as for the ant problem. This indicates that our refined crossover bias theory for bloat is applicable to the symbolic regression problem too, at least when selection has had enough time to focus the search of the population. Thus while the initial, randomly generated population might have a distribution like that in Figure 1(b), mechanisms like selection will quickly move (typically in just a few generations) the population to a distribution more like Figure 2(b), at which point our refined crossover bias theory for bloat should apply.

4. EXPERIMENTAL RESULTS

In the following three sections we will report a variety of experimental results where we studied the bloat and sampling behaviour of GP runs. In all the experiments we use the following set of parameters: generational GP, a crossover rate of 100%, no mutation, fitness proportionate selection, Ramped Half and Half initialization, a maximum depth of 6 for the individuals created in the initialisation phase, a maximum depth 200 for the individuals created by crossover, and no elitism. We've deliberately kept our model simple (e.g., by not including mutation) to allow us to more clearly focus on the interaction of population size and crossover. All our experimental results were obtained using 100 independent GP runs on each of four population sizes: 50, 100, 200, 400 and 800.

4.1 Population Size and Program Length

In Figure 3 we plot the average length of the individuals in the population at each generation for five different population sizes for the two studied benchmarks. Each curve is the average over 100 independent GP runs. For both problems the average program size grows more rapidly for bigger populations than for smaller ones, which fully confirms the prediction of the refined crossover bias theory that bloat depends on population size. In addition, we see that the phenomenon reaches a limit for population sizes between 800 and 1600 (not reported). This is presumably because those sizes are large enough to ensure consistent sampling of the short programs at each generation, for the 100 generations in our runs. Presumably we would eventually be able to distinguish among those population sizes if we ran for more generations, allowing the average size to reach the point that, for example, a population of size 800 no longer consistently samples the short, unfit programs.

We want to stress that if we had only considered the evidence provided by the Ramped Half and Half random sampling shown in Figure 1 in conjunction with the original crossover bias theory for bloat, we could have reasonably predicted that no bloat (or even some shrinking) would take place in the symbolic regression problem. The refined theory, however, made the right prediction because it considers the effect of selection on sampling.

[2]By varying the maximum tree depth, we indirectly vary the program length-classes being sampled.

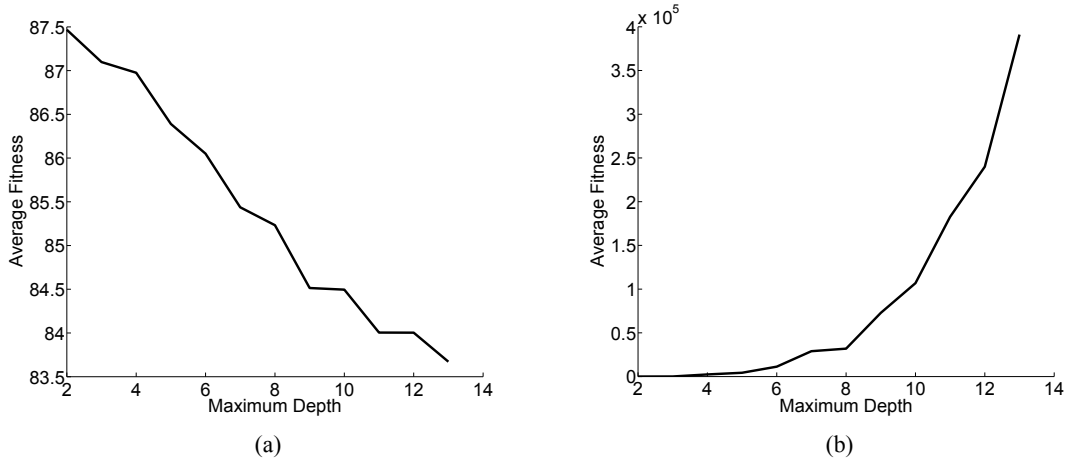

Figure 1: Average fitness vs. maximum depth over a sample of 4000 individuals generated with the Ramped Half and Half algorithm. All problems use minimization (the smaller the fitness, the better). (a): Artificial Ant on the Santa Fe trail. (b): Symbolic regression.

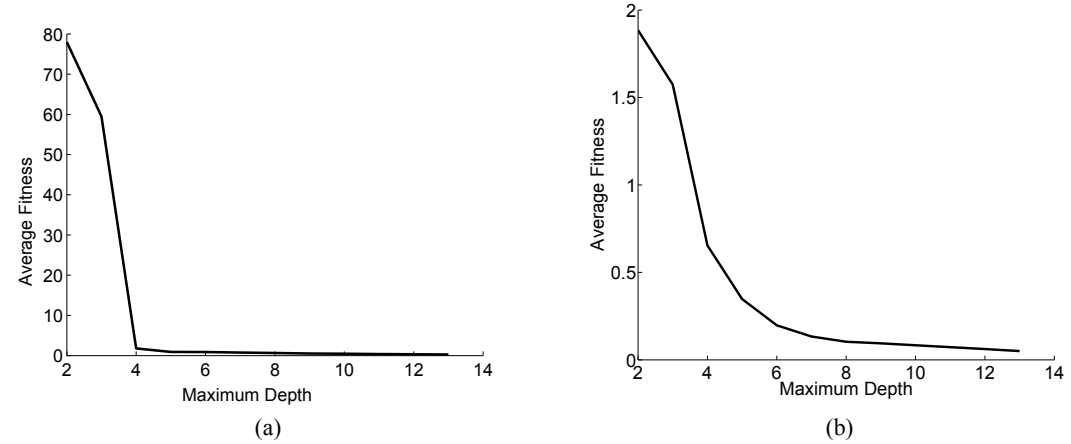

Figure 2: Average fitness vs. maximum depth over a sample of 4000 individuals generated with the Metropolis-Hastings algorithm (see the text for a definition of the acceptance criterion used). All problems use minimization (the smaller the fitness, the better). (a): Artificial Ant on the Santa Fe trail. (b): Symbolic regression.

4.2 Minimum, Median and Maximum Length

To gain further information on how the size distribution changes over time with different population sizes, we plot the minimum, median and maximum length of the individuals in the population in Figure 4. Curves are averages over 100 independent GP runs. We have used a logarithmic scale for the ordinates to better reveal the dynamics of the extreme values.

As we can clearly see, in both problems, the medians have approximately the same behaviour as the means (shown in Figure 3). That is, bloat occurs in all configurations but at a rate that depends on the population size. For the symbolic regression problem, however, we see an interesting pattern emerge (as also reported in [21]) for all population sizes in the first few generations where the median decreases instead of increasing. (The mean also decreased in the early generations, but this was not visible in Figure 3 due to its linear scale.) We should not be surprised to see this, because, as we discussed earlier, in the first few generations GP populations are random or nearly-random and so the fitness landscape perceived by GP is more similar to the one obtained by sampling with the Ramped Half and Half method (where short programs are relatively fit compared to the long ones) than with Metropolis-Hasting (where short programs are typically unfit).

If we now focus on the curves of the mean largest and mean smallest programs in the population shown in Figure 4, we see that, broadly speaking, these follow the same profile as the median (remember that the ordinates are in log scale). It is particularly interesting, for example, to note that as populations bloat more and more, the mean of the smallest program in each generation grows, indicating that the small unfit programs become less and less likely to be sampled. It is also interesting to see how the population size influences the relative position of the curves representing the extreme sizes in each generation. We see, for example, that for very small populations the longest programs tend to be closer to the median than for larger populations.

If we focus on the shortest programs in each generation, for the

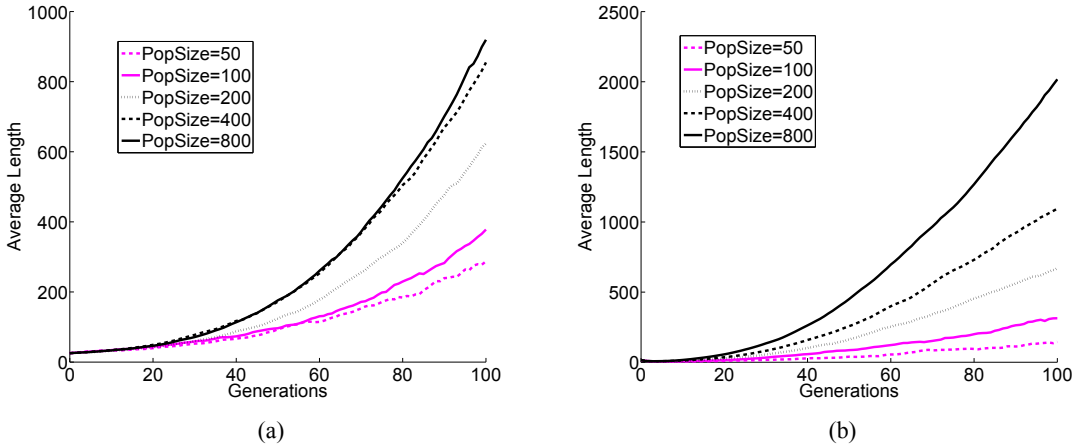

Figure 3: Average length of the individuals in the population against generations. Results are averages over 100 independent GP runs. (a): Artificial Ant on the Santa Fe trail. (b): Symbolic regression.

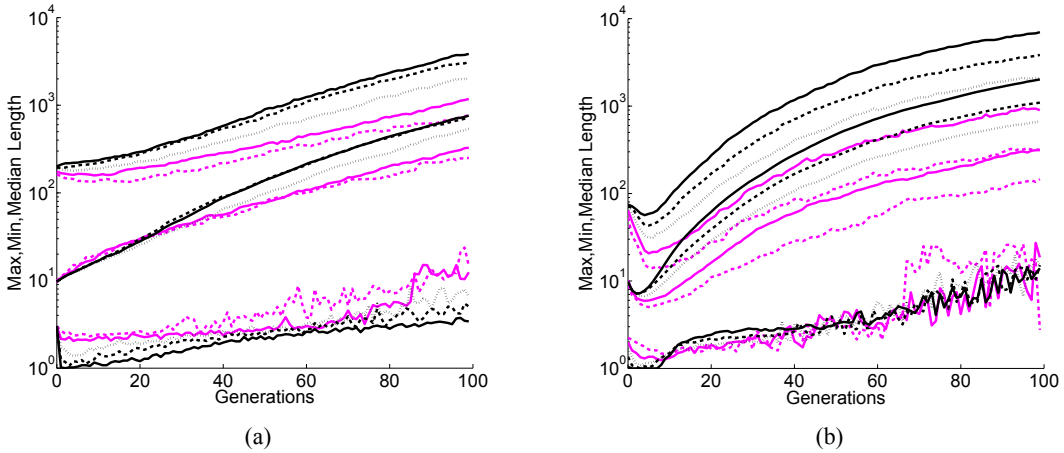

Figure 4: Maximum, minimum and median length of the individuals in the population against generations. Logarithmic scale has been used on the ordinates. Legends have not been included to avoid cluttering the pictures. They are as in Figure 3. Results are averages over 100 independent GP runs. (a): Artificial Ant on the Santa Fe trail. (b): Symbolic regression.

ant problem we see that on average the shortest programs in small populations are *longer* than for large populations. This happens despite the fact that the size distribution for larger populations is increasing faster than for smaller ones (as indicated both by their means and medians). These differences in growth rates partially mask the fact that there is a similar population-size effect also for the symbolic regression problem, as shown in Figure 5 where we compare the ratios between minimum and median lengths of the individuals in populations of different sizes over time for the symbolic regression problem. As the figure illustrates, the smaller populations tend to produce larger shortest programs than the large populations. So, we expect small populations to sample less often and for a shorter time the very short unfit programs, as predicted by the our theory.

As a final note, we can conjecture that as the population size grows further (beyond the maximum size 800 considered here), eventually we should see a saturation effect, by which no further population-size dependency will be observed. We have seen this effect start emerge at population sizes of 1,600 (experiments not reported), but, more importantly, we expect it to happen by the application of extreme value theory.[3]

4.3 Length Improvement

Figure 6 shows the length increases at each generation for both problems studied. The length increase at generation g is defined as $L_g = \overline{L}_g - \overline{L}_{g-1}$, where $\overline{L}_g$ is the average length of the individuals in the population at generation g. Curves are averages over 100 independent GP runs.

For the artificial ant problem (see Figure 6(a)) length increases are clearly bigger in larger populations than in smaller ones. The

[3]Extreme value theory predicts that the distribution of the maximum and minimum of a sample of stochastic variables approaches a limit that does not depend on the distribution of the variables as the number of samples grows. So, as the population size grows, eventually the sampling distribution and its lower tail should reach a limit. Thus, the rate of bloat should also reach a limit as predicted by our refined crossover bias theory.

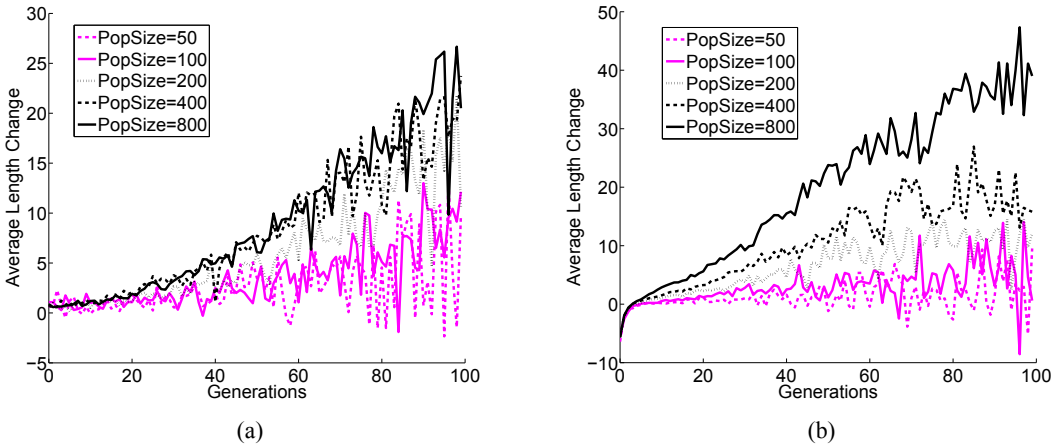

Figure 6: Average length increase of the individuals in the population against generations. Results are averages over 100 independent GP runs. (a): Artificial Ant on the Santa Fe trail. (b): Symbolic regression.

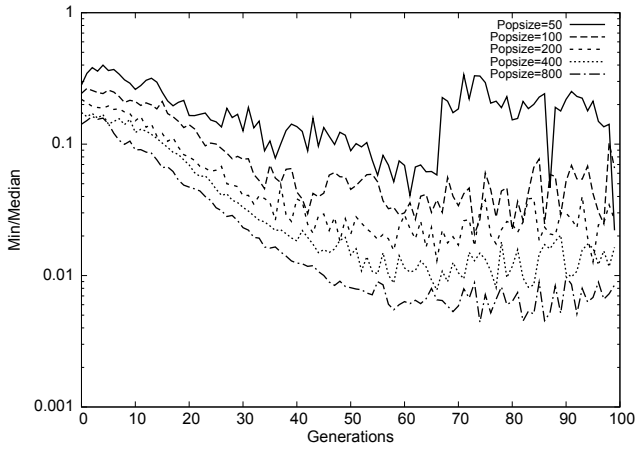

Figure 5: Ratio between minimum and median lengths of the individuals in the population against generations for the symbolic regression problem. Logarithmic scale has been used on the ordinates. Note that smaller populations have higher ratios.

same pattern can be observed for the symbolic regression problem (Figure 6(b)). These results are again consistent with the prediction that larger populations experience more rapid code growth. Additionally, for both problems, there is more noise in the data for small populations, and these oscillations can even lead L_g to become negative by a significant amount at certain generations.

All of this is consistent with the refined crossover bias theory and helps further explain the relationship between code growth and population size.

5. CONCLUSIONS AND FUTURE WORK

The crossover bias theory [18] (Section 1.2) is the most recent theory for bloat. Its attraction is its simplicity: bloat is simply caused by the sampling of short, unfit programs. The theory, however, has some elements that have not been fully clarified.

In particular, it assumes that the population is "sufficiently large" that all length classes are sampled (or at least that the short, subquality programs are sampled sufficiently often). In some practical cases this may not be true, for example if the population is relatively small and/or the mean program size is large.

In addition, the crossover bias theory does not explain what is meant by its assumption that short programs are unfit or less fit than longer ones. There can, for example, be significant differences between the static fitness of programs and the dynamic average fitness resulting from the interaction of the many processes (such as selection) going on inside a GP run.

In this paper we have discussed these issues and proposed a refined version of the crossover bias theory that helps clarify what effects population size has on bloat. The theory, in particular, predicts that smaller populations will bloat more slowly than larger ones. In addition, the theory predicts that bloat will be observed only in problems where small programs are less fit than longer ones when looking at samples created by fitness-based importance sampling, such as that performed by the Metropolis-Hastings method.

Experiments with two classical GP benchmarks, the artificial ant on the Santa Fe trail and a symbolic regression problem, have fully corroborated the theory.

The analysis presented in this paper has been largely qualitative. So, although the empirical evidence has fully supported it, in the future, we plan to look at the issues touched upon in this paper from a more formal, theoretical point of view. This should lead to a more complete model of individuals' size growth and its relationship with population size, and might lead to a formalisation of the refined crossover bias theory.

6. REFERENCES

[1] T. Blickle and L. Thiele. Genetic programming and redundancy. In J. Hopf, editor, *Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94, Saarbrcken)*, pages 33–38, Saarbrcken, Germany, 1994. Max-Planck-Institut fr Informatik (MPI-I-94-241).

[2] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, Nov. 1995.

[3] S. Dignum and R. Poli. Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In D. Thierens, et al., editors, *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, volume 2, pages 1588–1595, London, 7-11 July 2007. ACM Press.

[4] J. R. Koza. A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, volume IV, pages 310–318. IEEE Press, 1992.

[5] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[6] W. B. Langdon. Convergence rates for the distribution of program outputs. In W. B. Langdon, et al., editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 812–819, New York, 9-13 July 2002. Morgan Kaufmann Publishers.

[7] W. B. Langdon. How many good programs are there? How long are they? In K. A. De Jong, et al., editors, *Foundations of Genetic Algorithms VII*, pages 183–202, Torremolinos, Spain, 4-6 Sept. 2002. Morgan Kaufmann. Published 2003.

[8] W. B. Langdon. Convergence of program fitness landscapes. In E. Cantú-Paz, et al., editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1702–1714, Chicago, 12-16 July 2003. Springer-Verlag.

[9] W. B. Langdon. The distribution of reversible functions is Normal. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practise*, chapter 11, pages 173–188. Kluwer, 2003.

[10] W. B. Langdon. The distribution of amorphous computer outputs. In S. Stepney and S. Emmott, editors, *The Grand Challenge in Non-Classical Computation: International Workshop*, York, UK, 18-19 Apr. 2005.

[11] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[12] W. B. Langdon and R. Poli. The halting probability in von Neumann architectures. In P. Collet, et al., editors, *Proceedings of the 9th European Conference on Genetic Programming*, volume 3905 of *Lecture Notes in Computer Science*, pages 225–237, Budapest, Hungary, 10 - 12 Apr. 2006. Springer.

[13] N. F. McPhee and R. Poli. A schema theory analysis of the evolution of size in genetic programming with linear representations. In J. Miller, et al., editors, *Proceedings of the Fourth European Conference on Genetic Programming (EuroGP-2001)*, volume 2038 of *LNCS*, pages 108–125, Lake Como, Italy, 2001. Springer, Berlin, Heidelberg, New York. Lecture notes in Computer Science vol. 2038.

[14] P. Nordin and W. Banzhaf. Complexity compression and evolution. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 310–317, Pittsburgh, PA, USA, 1995. Morgan Kaufmann.

[15] A. Piszcz and T. Soule. Dynamics of evolutionary robustness. In M. Keijzer *et al.*, editor, *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 1, pages 871–878, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

[16] R. Poli. General schema theory for genetic programming with subtree-swapping crossover. In J. F. Miller, et al., editors, *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, pages 143–159, Milan, 18-20 Apr. 2001. Springer-Verlag.

[17] R. Poli and W. B. Langdon. Efficient markov chain model of machine code program execution and halting. In R. L. Riolo, et al., editors, *Genetic Programming Theory and Practice IV*, volume 5 of *Genetic and Evolutionary Computation*, chapter 13. Springer, Ann Arbor, 11-13 May 2006.

[18] R. Poli, W. B. Langdon, and S. Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In M. Ebner, et al., editors, *Proceedings of the 10th European Conference on Genetic Programming*, volume 4445 of *Lecture Notes in Computer Science*, pages 193–204, Valencia, Spain, 11 - 13 Apr. 2007. Springer.

[19] R. Poli, W. B. Langdon, and N. F. McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[20] R. Poli and N. F. McPhee. General schema theory for genetic programming with subtree-swapping crossover: Part II. *Evolutionary Computation*, 11(2):169–206, June 2003.

[21] S. Silva and J. Almeida. Dynamic maximum tree depth. In E. Cantú-Paz, et al., editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2724 of *LNCS*, pages 1776–1787, Chicago, 12-16 July 2003. Springer-Verlag.

[22] T. Soule and J. A. Foster. Effects of code growth and parsimony pressure on populations in genetic programming. *Evolutionary Computation*, 6(4):293–309, 1998.

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

[24] L. Vanneschi, Y. Pirola, P. Collard, M. Tomassini, S. Verel, and G. Mauri. 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, 2006.