Generalisation of the Limiting Distribution of Program Sizes in Tree-based Genetic Programming and Analysis of its Effects on Bloat

Stephen Dignum
Department of Computer Science
University of Essex, UK
sandig@essex.ac.uk

Riccardo Poli
Department of Computer Science
University of Essex, UK
rpoli@essex.ac.uk

ABSTRACT

Recent research [1] has found that standard sub-tree crossover with uniform selection of crossover points, in the absence of fitness pressure, pushes a population of GP trees towards a Lagrange distribution of tree sizes. However, the result applied to the case of single arity function plus leaf node combinations, e.g., unary, binary, ternary, etc trees only. In this paper we extend those findings and show that the same distribution is also applicable to the more general case where the function set includes functions of mixed arities. We also provide empirical evidence that strongly corroborates this generalisation. Both predicted and observed results show a distinct bias towards the sampling of shorter programs irrespective of the mix of function arities used. Practical applications and implications of this knowledge are investigated with regard to search efficiency and program bloat. Work is also presented regarding the applicability of the theory to the traditional 90%-function 10%-terminal crossover node selection policy.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance

Keywords

Genetic Programming, Crossover Bias, Program Size Distribution, Bloat, Initialisation

1. INTRODUCTION

Program size is important to the efficiency of Genetic Programming (GP). Too large a program will produce an inefficient solution and one that is costly to quantify in terms of fitness, whilst too small a program will not produce a viable solution.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

Theoretical and empirical work in this area has often looked at how populations will be distributed by length with repeated application of a crossover operator on a flat fitness landscape [2, 3]. Although this is a highly improbable real-world scenario, if we take account of the fact that once parents are selected the process of reproduction becomes a purely mechanical gene swapping/creation exercise, with no regard to fitness[1], the worth of such experimentation becomes obvious with regard to analysis of potential operator length bias.

Early work has concentrated on Linear GP [2], particularly how individuals made up of unary functions and terminals will over time become distributed by length according to the following fixed-point Gamma distribution on repeated application of the standard crossover operator with uniform selection of crossover points:[2]

$$\Pr\{\ell\} = \ell r^{\ell-1}(1-r)^2 \tag{1}$$

where $r = (\mu_0 - 1)/(\mu_0 + 1)$ and μ_0 is the average size of the individuals in the population at generation 0. Recent research [1], however, has shown that length distributions of populations with different combinations of terminals and functions can also be predicted accurately. In particular, it was found that the limiting distribution is the following Lagrange distribution of the second kind:

$$\Pr\{n\} = (1 - ap_a)\binom{an+1}{n}(1-p_a)^{(a-1)n+1}p_a^n \tag{2}$$

where $\Pr\{n\}$ is the probability of selecting a tree with n internal nodes and a is the arity of functions that can be used in the creation an individual. The parameter p_a was shown to be related to μ_0 and a according to the formula:

$$p_a = \frac{2\mu_0 + (a-1) - \sqrt{((1-a) - 2\mu_0)^2 + 4(1-\mu_0^2)}}{2a(1+\mu_0)} \tag{3}$$

The distribution in Equation (1) was shown to be a special case of Equation (2) (obtained by setting $a = 1$).

As we can see the distribution is limited to the prediction of GP populations made up of individuals that contain functions of a single arity. The first contribution of this paper

[1]In fact, in the case of most commonly used GP variation operators there is also no analysis of node labels [4].
[2]This is an ordinary subtree crossover where all nodes in a tree (including terminals) are selected with equal probability, i.e., without the 90%-function 10%-terminal node policy often used by practitioners.

will be to overcome this limit. We will then use this extension to understand the practical impact of crossover aiming for a Lagrange distribution on GP effectiveness.

The paper is organised as follows. Section 2 looks at how the work described in [1] and summarised above can be generalised to predict a distribution of program sizes for GP experiments with a function set that contains a mix of arities (previous work looked at function sets with only one function arity). Section 3 provides empirical evidence to support our extension. Section 4 looks at possible applications of the result, namely initialisation and the sampling of particular program sizes. In Section 5 we look at the applicability and implications of the theory in the case where crossover points are chosen using the traditional 90%-function 10%-terminal node selection policy. Finally, Section 6 draws some conclusions.

2. GENERALISATION

Within [1], the distribution in Equation (2) was shown to describe the limiting distribution of tree sizes obtained in a population under repeated operation of standard crossover with uniform selection of nodes, when the internal nodes are all of the same arity, a. The theory leading to this result is extremely complex, and, so, it appears very difficult to generalise this result to the case of generic GP primitive sets starting from first principles.

Before embarking on such a difficult analysis, therefore we wondered to what extent Equation (2) would still be applicable if functions of mixed arities were allowed. In particular, rather than viewing a as simply the identical arity of a particular set of functions selected we can choose to use this as the expected number of children of a non-terminal picked from a function set, i.e. an average arity. Hence, an average a, which we will call $\bar{a}$ to avoid confusion, can be calculated for mixed function arities from experimental initialisation parameters as follows

$$\bar{a} = E(arity(F)) = \sum_f arity(f)P(F = f) \tag{4}$$

where f is a non-terminal to be used in the GP experiment, $arity(f)$ is a function returning the arity of the non-terminal f, and $P(F = f)$ being the probability that a particular non-terminal f will be selected for a non-terminal node by the tree inititialisation procedure. For traditional FULL and GROW initialisation methods non-terminals are chosen with equal probability [5]. Alternatively, $\bar{a}$ can be given simply by the calculation of non-terminal average arity from the initial population. (For larger populations these will of course be almost identical.)

With this new definition of a, we have the problem that the term $an + 1$ in the binomial coefficient $\binom{an+1}{n}$ in Equation (2) may be non-integer. So, the next step is to alter the definition of binomial coefficient so that it will also work with non-integer data values, as demanded by our new average arity version of a. This can be done simply by using the Gamma function as an alternative for the factorials used, i.e., $\Gamma(n + 1) = n!$ [6]. As a result we can rewrite the binomial coefficient as follows

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{\Gamma(n+1)}{\Gamma(n-k+1)\Gamma(k+1)} \tag{5}$$

Therefore

$$\binom{an+1}{n} = \frac{\Gamma(an+2)}{\Gamma(an-n+2)\Gamma(n+1)} \tag{6}$$

which, substituted into Equation (2), gives us the distribution

$$\Pr\{n\} = (1-\bar{a}p_{\bar{a}})\frac{\Gamma(\bar{a}n+2)}{\Gamma((\bar{a}-1)n+2)\Gamma(n+1)}(1-p_{\bar{a}})^{(\bar{a}-1)n+1}p_{\bar{a}}^n \tag{7}$$

We conjecture that this will be the limiting distribution for function sets of mixed arities. As we will see in the following section, this assumption appears to be very realistic.

3. EMPIRICAL VALIDATION

In order to verify empirically the distribution proposed we performed a number of runs of a GP system in Java. As in [1] a relatively large population of 100,000 individuals was used in order to reduce drift of average program size and to ensure that enough programs of each length class were available. The FULL [7] initialisation method was used with each function having the same probability of selection. Each run consisted of 500 generations.

Histograms were collected from the final generations (in order to ensure the effects of our chosen initialisation method have been washed-out[3]), each bin being the number of internal nodes contained within a program. A comparison of theoretical predictions and observed results, averaged over twenty runs, is available in Figures 1–4. Figures 1 and 2 are particularly interesting since they represent the behaviour of the primitive sets for the Parity and Artificial Ant problems [7] respectively (the other two figures represent hypothetical function sets where even more spread of primitive arities is present). In all cases the match between theoretical predictions and empirical data is striking (note that sampling noise is more marked in the longer-length classes because there are fewer programs in each).

As we can see from both our theoretical and observed results there is a distinct bias towards the sampling of smaller programs as it was found in [1]. The allowance of mixed arity function sets, in addition to single arity sets, does not alter this bias.

4. PRACTICAL APPLICATIONS

Now that we have a formula to predict our distribution of program lengths under crossover the next step is to see how we can apply our theory to practical GP applications.

4.1 Sampling of Program Size

With some problems we may have an initial idea of likely program lengths that may be required to provide an acceptable solution. For example, this knowledge may range from knowing that a minimum length is required in that enumerated search has been unsuccessful up to a length when the search was found to be intractable, or simply that previous attempts using specific designs or other search algorithms had provided some initial success at certain solution lengths. Also, for classical test problems, a great deal of information

[3]It would, of course, be interesting to see how many generations are required to nullify length effects of different initialisation methods. However, this is beyond the scope of this paper and will be a subject for future research.

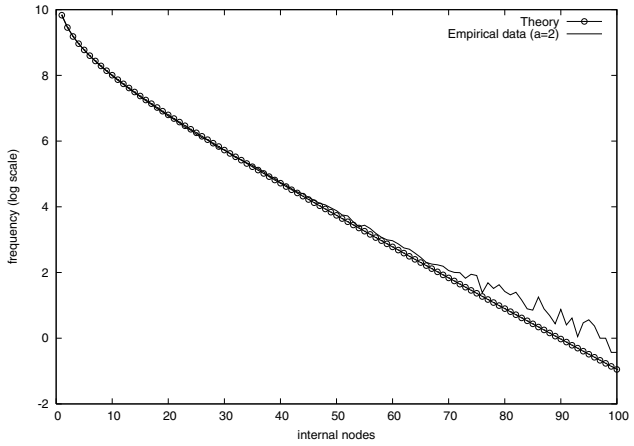

Figure 1: Comparison between empirical and theoretical program size distributions for trees made up of only an arity 2 function (a=2) initialised with FULL method (depth=3, initial mean size $\mu_0 = 15.0$, mean size after 500 generations $\mu = 14.494514$)

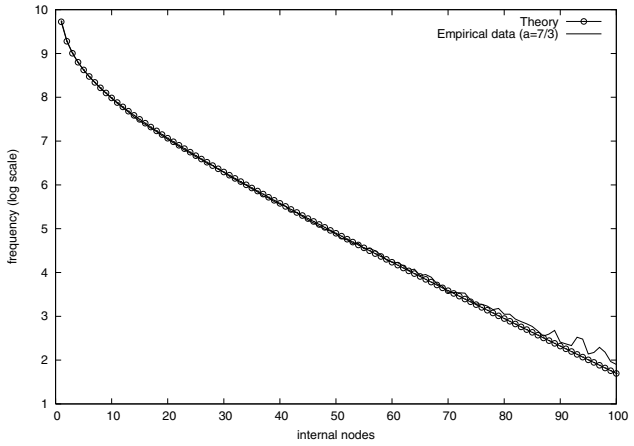

Figure 2: Comparison between empirical and theoretical program size distributions for trees made up of a mix of 2, 2, and 3 arity functions (a=7/3) initialised with FULL method (depth=3, initial mean size $\mu_0 = 21.48$, mean size after 500 generations $\mu = 23.5739605$)

is available about the distribution of program functionality (including fitness) as the length of programs is varied (see, for example, [8]). So, it is possible to infer from such distributions what are the best length classes on which to concentrate the search for solutions.

Our first step is to see how we can use the parameters of the limiting distribution of crossover to at least begin to sample a significant number of programs of a particular size. The average arity value $\bar{a}$ is derived from our function set which we assume is fixed for the experiment i.e. that all functions are deemed likely to be required to solve the problem. We are, therefore, left with our value for $p_{\bar{a}}$ which is in turn derived from $\bar{a}$ which we do not want to change and the mean program size of the initital generation μ_0. This latter value can be directly altered by manipulating the parameters of our initialisation method in order that significantly large trees are produced.

To illustrate this if we start with the Artificial Ant problem [7], we have three functions: IF-FOOD-AHEAD, PROGN2, PROGN3 with arities of 2, 2, and 3 respectively. As seen in figure 2 our value of $\bar{a}$ for this group of arities is 7/3. Knowing this we can alter the value of μ_0 to enable the sampling by crossover of different program length spaces. Figure 5 shows how varying μ_0 alters the crossover distribution. We can see that there is a consistent high sampling of smaller programs, however, starting from a reasonable base figure relatively small increases in μ_0 allow larger programs to be sampled more consistently.

As an illustration, the Artificial Ant problem is known to have no ideal solutions before a program size of 11 [8], if we were to ensure that a percentage of programs sampled were to have at least 5 internal nodes[4] for a μ_0 value of 5, 10, and 20 crossover would sample 16%, 36%, and 54% of the population respectively for that program size or greater.

So, if we initialised the population so that the length distribution is a Lagrange distribution of the second kind (as we will discuss in the following section), we could perform an informed choice of what is the best μ_0 to use to maximise the chance of solving a problem. Naturally, if, instead, we initialised the population with a distribution that is very different from a Lagrange distribution of the second kind, as is the case, for example, for FULL or GROW initialisation, then it would take a few generations before crossover transforms the size distribution into something resembling such a distribution. However, eventually this would happen. At that point, a good choice of μ_0 would start paying off.

[4]This estimate is derived from $length = an + 1$ see [1].

4.2 Initialisation

By choosing to apply GP to a particular problem we have made an assumption that both fitness based selection and crossover are likely to provide an efficient search method to provide a solution. Normally a GP initialisation method is used GROW, FULL, RAMPED etc that takes no account of the eventual distributions 'desired' by either crossover or fitness (or any other GP operator such as mutation). Normally, eventual fitness values are not known in advance of a GP experiment, however, we now have evidence to show that crossover will, with repeated application, distribute the population according to a predictable distribution. Our next step is to investigate the possible benefits or disadvantages of initialising a population by length to take account of crossover.

We could of course write an algorithm to initialise the population according to the eventual predicted distribution desired by crossover. The most straightforward programatical method, however, is to simply run the first 'X' number of generations of a GP experiment without fitness, thereby allowing crossover to distribute program lengths without having to endure the cost of fitness calculations.

To test this idea we took two out-of-the-box problems from the ECJ [9] evolutionary toolkit, the Artificial Ant as previously discussed and 4 Bit Even Parity, making adjustments to remove mutation, ensure uniform selection of crossover points, and to prevent a depth limit being applied. A population size of 1024 individuals was used and the results averaged over a hundred runs. All experiments were

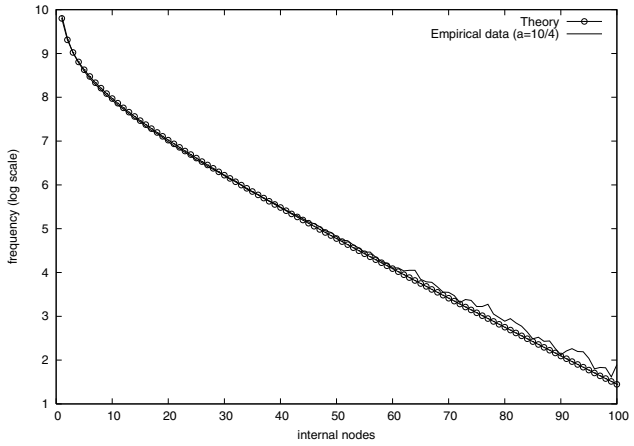

Figure 3: Comparison between empirical and theoretical program size distributions for trees made up of a mix of 1, 2, 3 and 4 arity functions (a=10/4) initialised with FULL method (depth=3, initial mean size $\mu_0 = 25.38$, mean size after 500 generations $\mu = 23.76$)

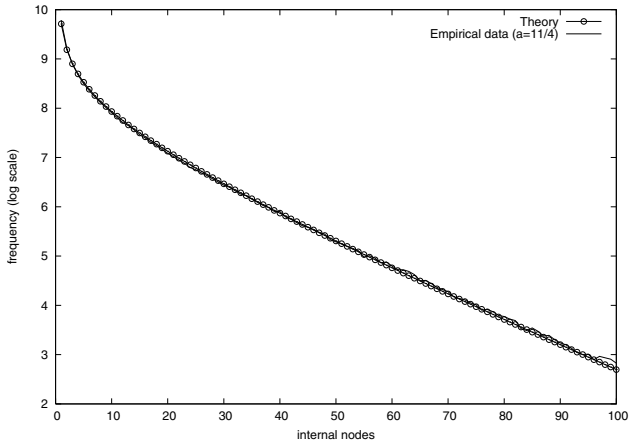

Figure 4: Comparison between empirical and theoretical program size distributions for trees made up of a mix of 1, 3, 3 and 4 arity functions (a=11/4) initialised with FULL method (depth=3, initial mean size $\mu_0 = 32.12$, mean size after 500 generations $\mu = 33.22$).

initialised using the FULL method with a depth of 3, each experiment looked at the effect of running with zero, twenty or fifty initial generations where a constant fitness value was applied before allowing the experiment to continue as normal. Naturally, the flat-fitness phase was much faster than normal, since no fitness evaluation was required. During this phase crossover was free to distribute the population towards its limit size distribution.

As we can see in figures 6-7, there is noticable degrading in mean fitness for the populations with initial crossover-only generations compared to those were fitness is applied straight away. This is also seen in figures 8-9 where best fitness has been recorded. If we look at figures 10-11 we can see there is much greater variation in individual fitness for the fitter populations.

The reason for this effect is in the sampling of smaller programs produced by "Lagrange-like" initialisation. A population distributed by length through the application of crossover will contain large numbers of relatively small programs. In both the Artificial Ant and Parity problems these short programs are associated with low fitness [8]. Crossover has, therefore, created lots of smaller programs with relatively poor fitness values, whilst FULL originally produced programs above a reasonable threshold[5].

We can also apply these findings to the problem of understanding the origins of bloat, as we discuss in the next section.

4.3 Bloat

Bloat, the growth of program size during a GP run without a significant return in terms of program fitness, is seen in many GP experiments. Figures 12-13 show graphs of program growth for the two problems described in the previous section. There is a noticeable increase in program growth for the populations with "Lagrange-like" initialisation.

[5]It should be noted, however, that in problems where high fitness is associated very small programs the opposite may be true

At first one might find this result very surprising. How is it that initialisation has such a big effect on bloat, which is typically associated with late generations of a run, when effectively the population starts stagnating? An explanation for this lies in the combination of the crossover sampling distribution and fitness.

In every generation we produce a mating pool of relatively fit programs which will be used for reproduction. The standard crossover operator (with uniform selection of crossover points) will create a certain distribution of program sizes irrespective of the fitness of programs selected in that mating pool. For example, if size 45 programs are for some reason very fit, once in the mating pool, crossover takes no account of this fitness and produces a distribution of differing programs. From the theory provided earlier in the paper we know that this distribution is biased, in frequency, towards smaller programs, but with a tail of larger programs.

As the smaller programs will tend to be less fit (as a proportion of the population), these will not be selected for mating in the next generation. The larger programs will be selected as parents, thereby providing a higher probability of larger trees being created as children.

This explanation fits completely within the mathematical explanation for bloat provided in [10], where the following size evolution equation was derived[6]

$$E[\mu(t+1) - \mu(t)] = \sum_l N(G_l)(p(G_l, t) - \Phi(G_l, t))$$

where μ is mean program size, G_l represents all programs of a particular shape, $N(G_l)$ represents the size of programs of shape G_l, $p(G_l, t)$ represents the selection probability for programs of shape G_l and $\Phi(G_l, t)$ represents their frequency in the population. If, as is the case for the Ant and Par-

[6]A major finding of this paper is that for symmetric subtree-swapping crossover operators, e.g., standard crossover, "The mean program size evolves as if selection only was acting on the population". The derivation of this equation is explained in section 5.4 of [10].

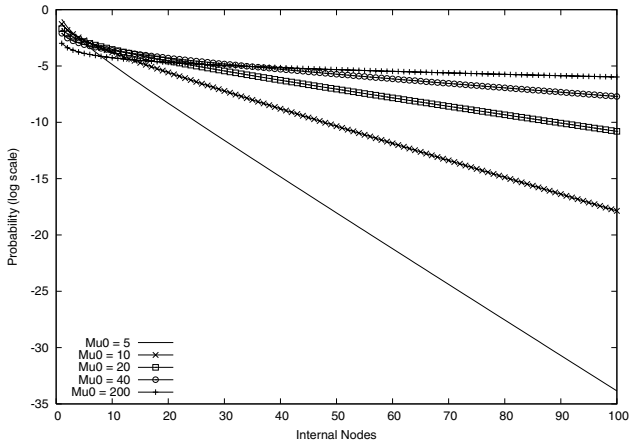

Figure 5: Comparison of Pa probability distributions for different μ_0 parameters for the Artificial Ant problem. Note: the use of a logarithmic scale for the probability axis.

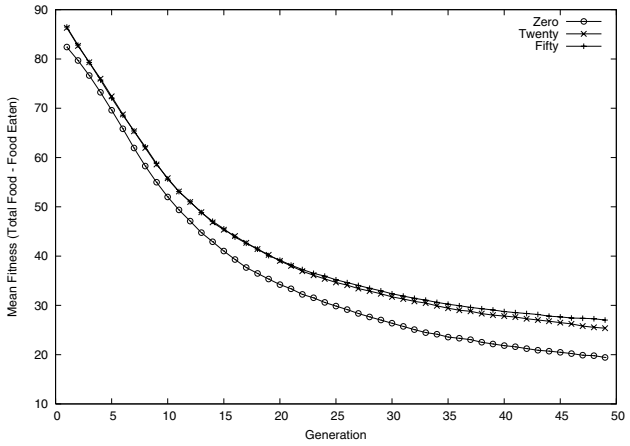

Figure 6: Comparison of mean fitness values for populations with zero, twenty and fifty prior generations of crossover without fitness for the Artificial Ant problem. Note: values for generations where the fitness function was "switched off" are not shown.

ity problems, the selection probability for short programs is consistently lower than their frequency, then, everything else being equal, one must expect $E[\mu(t+1) - \mu(t)] > 0$, and hence bloat will occur.

In essence where a reasonable minimum size threshold exists for relatively fit programs, the repeated application of fitness based selection and standard crossover (with uniform selection of crossover points) to a population will cause bloat to occur. Our experimentation has shown that by allowing crossover to distribute a population before applying fitness we make this effect more acute. That is we are producing populations with proportionately more shorter programs than those created using more traditional GP initialisation methods.

5. CROSSOVER WITH 90%-FUNCTION/ 10%-TERMINAL CROSSOVER-POINT SELECTION POLICY

The theoretical results on the evolution of size during GP runs developed in [2, 3, 1] and in this paper assume that crossover points are selected uniformly at random out of the set of all primitives in a tree (including terminals). This assumption simplifies the, already complex, mathematics required to obtain results in this area. However, GP researchers and practitioners almost universally use a 90%-function/10%-terminal crossover-point selection policy. We will call this policy 90/10 for brevity. It is, therefore, interesting to investigate to which extent the theory for uniform crossover point selection applies to this, non-uniform case. In this section, we will present a preliminary analysis of this issue. Much more work will need to be devoted to it in future research.

The starting point for our analyis is that in the 90/10 policy crossover points are still uniformly distributed within each class of nodes (internal vs. terminals). So, 90/10 crossover may still have the symmetries required to model the tree population using a branching process as was done in [1]. This, in turn, suggests that some form of modified Lagrange distribution of the second kind might still be applicable to describe the limiting distribution of tree sizes for 90/10 crossover.

We are not in a position to say what the exact size distribution under 90/10 crossover would be. However, we note that the 90/10 policy has a considerable effect on the proportion of programs of size 1, (i.e., with no internal nodes). This is because such programs are only created if the crossover point in the root donating parent is the root node itself and the crossover point in the subtree donating parent is a terminal. In the 90/10 policy the probability of a leaf node is artificially set to 10%, which, with typical primitive sets, is notably smaller than with uniform selection of crossover points. Naturally, in the same conditions, the probability of the root node being chosen grows, but not enough to compensate for the drop the probability of selecting terminals. So, their product – the probability of creating size 1 programs – is much smaller than in the uniform case. Naturally, if there is a drop in the frequency of size 1 programs, there must be a corresponding increase in the other length classes.

On the basis of these observations, it is clear that the distribution of tree sizes under 90/10 crossover cannot be a pure Lagrange distribution of the second kind. However, we might expect to see a "Lagrange-like" distribution for programs with one or more internal nodes. We conjecture that the following family of distributions may provide a reasonable first order approximation of the true limiting distribution of sizes for subtree crossover with 90/10 policy:

$$\Pr_{90/10}\{n\} = \begin{cases} \alpha & \text{if } n = 0, \\ (1-\alpha)\frac{\Pr\{n,\bar{a},p_{\bar{a}}\}}{1-\Pr\{0,\bar{a},p_{\bar{a}}\}} & \text{otherwise,} \end{cases} \quad (8)$$

where α is a constant and $\Pr\{n, \bar{a}, p_{\bar{a}}\}$ stands for the extension to the Lagrange distribution of the second kind in Equation (7). In this formula the denominator $1 - \Pr\{0, \bar{a}, p_{\bar{a}}\}$ has the effect of normalising the numerator $\Pr\{n, \bar{a}, p_{\bar{a}}\}$ in such a way to make it a probability distribution for $n \geq 1$.[7]

[7]Naturally $\Pr\{n, \bar{a}, p_{\bar{a}}\}$ is already a probability distribution for $n \geq 0$.

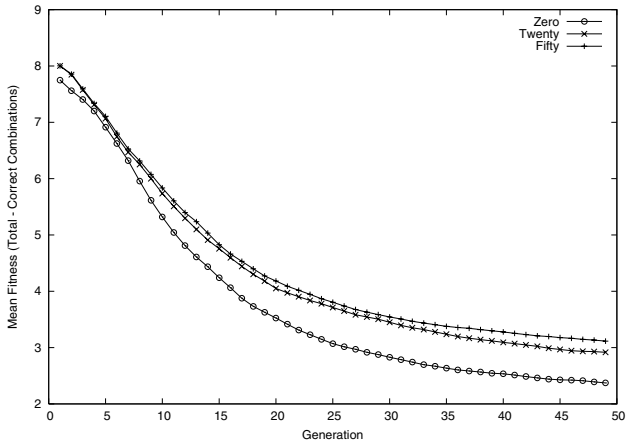

Figure 7: Comparison mean fitness values for populations with zero, twenty and fifty prior generations of crossover without fitness for the 4 Bit Even Parity problem

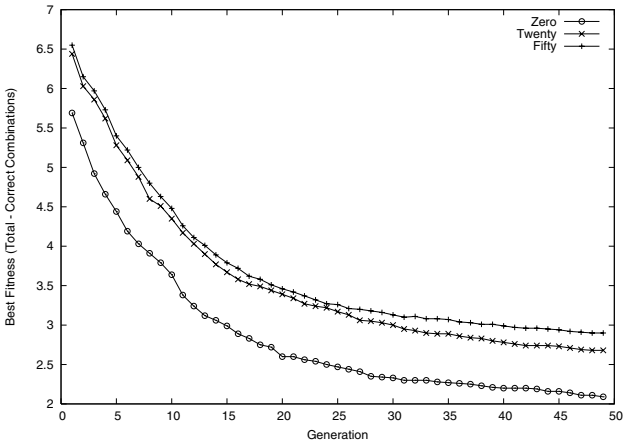

Figure 9: Comparison best fitness values for populations with zero, twenty and fifty prior generations of crossover without fitness for the 4 Bit Even Parity problem

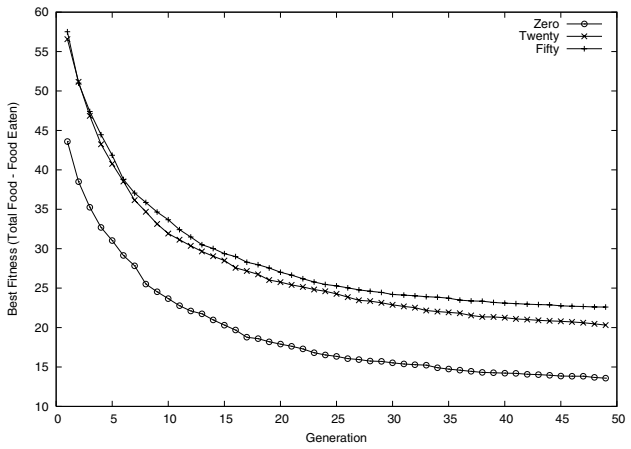

Figure 8: Comparison best fitness values for populations with zero, twenty and fifty prior generations of crossover without fitness for the Artificial Ant problem

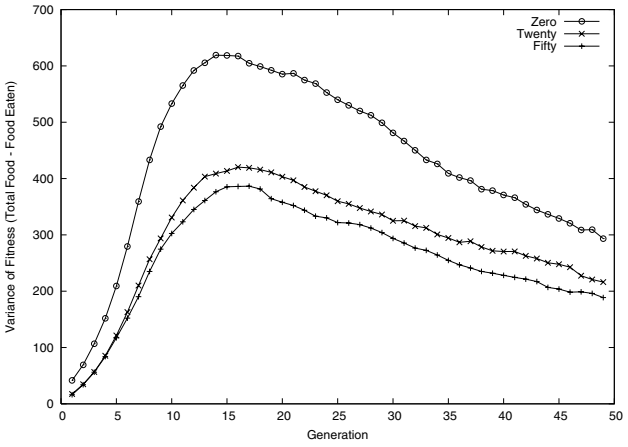

Figure 10: Comparison fitness variance values for populations with zero, twenty and fifty prior generations of crossover without fitness for the Artificial Ant problem

That is $\sum_{n\geq 1}\left(\frac{\Pr\{n,\bar{a},p_{\bar{a}}\}}{1-\Pr\{0,\bar{a},p_{\bar{a}}\}}\right)=1$. Then the multiplication by $(1-\alpha)$ ensures that $\Pr_{90/10}\{n\}$ is a probability distribution for any value of $\alpha\in[0,1]$, $\bar{a}>0$ and $p_{\bar{a}}\in[0,1/\bar{a}]$.

To corroborate this conjecture we performed GP runs with the same configuration as in Section 4 except that this time we used 90/10 crossover. Figure 14 shows a comparison between empirical size distributions observed at generation 500 in our runs and corresponding $\Pr_{90/10}\{n\}$ modified Lagrange distributions for the case of trees made up with primitives of average arity $\bar{a}=1.5$, 2 and 2.5 (see caption of Figure 14 for additional information). The theoretical models were obtained by setting α equal to the number of programs with no internal nodes and finding the value of $p_{\bar{a}}$ which minimised the mean square error between empirical data and Equation (8). Naturally, our choice of α guarantees a perfect fit for $n=0$. Its value, however, influences the scaling of the whole distribution for $n>0$. It is, then, remarkable to see that such choice allows a very accurate match between our conjectured theoretical distribution and the distribution observed in real runs.

These results suggest that many of the implications and applications of the size bias of subtree crossover with uniform selection of crossover points discussed in Section 4 carry over to the case of 90/10 crossover.

6. CONCLUSIONS

A decoupling of fitness from the mechanical reproduction process means that an awareness of bias within our reproduction operators is essential in the understanding of how efficiently an algorithm will search the possible program space.

We have presented within this paper a generalisation of the limiting distribution of program sizes in tree-based genetic programming (Equation (7)).

The formula provides evidence that the reproduction process has a strong bias to sample shorter programs when crossover with uniform selection of crossover points is ap-

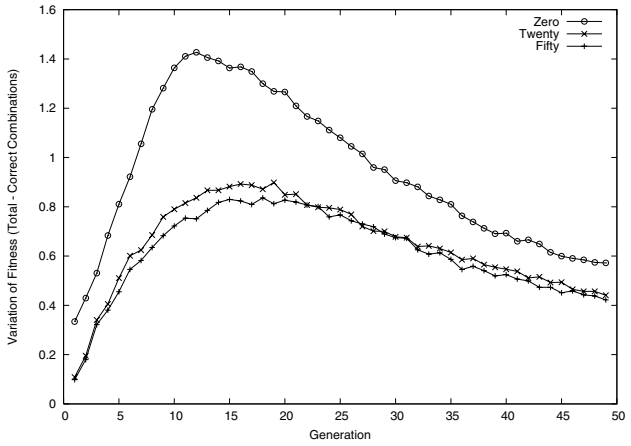

Figure 11: Comparison fitness variance values for populations with zero, twenty and fifty prior generations of crossover without fitness for the 4 Bit Even Parity problem

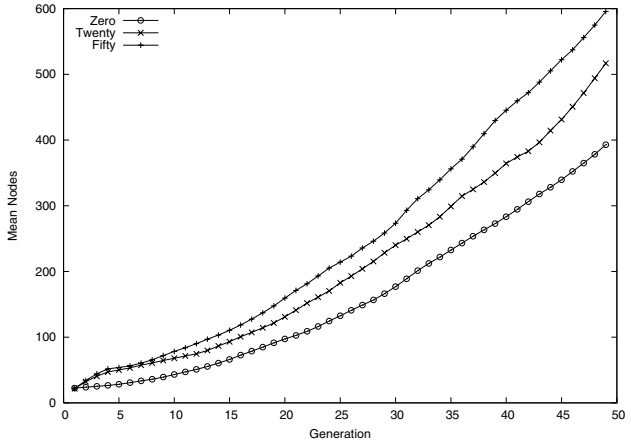

Figure 12: Comparison mean number of nodes for populations with zero, twenty and fifty prior generations of crossover without fitness for the Artificial Ant problem

plied. This bias is most acute when a population has a small mean program size.

A number of practical concerns have been presented in light of this bias. The first is an inability to sample large proportions of the problem space, i.e., those associated with larger programs. Additional problems with crossover are also encountered when fitness is associated with certain minimum program sizes leading to relatively poor performance in the finding of fit solutions and a tendency for the population to bloat. In particular, we have shown that bloat can occur even from generation 0. It normally happens effectively only in much later generations because it takes time for crossover to heavily skew the size distribution towards short programs, towards its limiting Lagrange distribution of the second kind.

It is, therefore, evident that future research should look into ways of modifying the bias of crossover if bloat is to be prevented. One way to achieve this is to "equalise" the

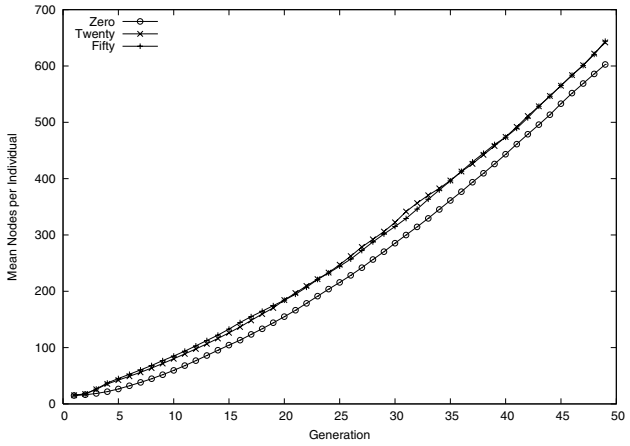

Figure 13: Comparison mean number of nodes for populations with zero, twenty and fifty prior generations of crossover without fitness for the 4 Bit Even Parity problem

distribution of tree sizes produced by crossover. This can be done, for example, by using an adaptive process of crossover point selection which actively streers the offspring size distribution towards a more desirable distribution.

Aware of the fact that many practitioners use crossovers with a non-uniform selection of crossover points, in Section 5 we extended our theoretical results to the case where crossover points are chosen using the traditional 90%-function 10%-terminal node selection policy. Surprisingly, the model we obtain for the distribution of program sizes (Equation (8)) is a simple modification of the one for the uniform case. This reveals that, except for the case of programs with no internal nodes, the two crossovers have effectively the same size biases. So, all of the implications and applications we considered in Sections 4.1–4.3, including our explanation for bloat, carry over to the 90/10 crossover case.

7. REFERENCES

[1] Riccardo Poli, William B. Langdon, and Stephen Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In *Proceedings of the 10th European Conference on Genetic Programming*, Lecture Notes in Computer Science, Valencia, Spain, 11-13 April 2007. Forthcoming.

[2] Riccardo Poli and Nicholas Freitag McPhee. Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In Julian F. Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi, and William B. Langdon, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 126–142, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

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

[4] Riccardo Poli and Nicholas Freitag McPhee. General schema theory for genetic programming with

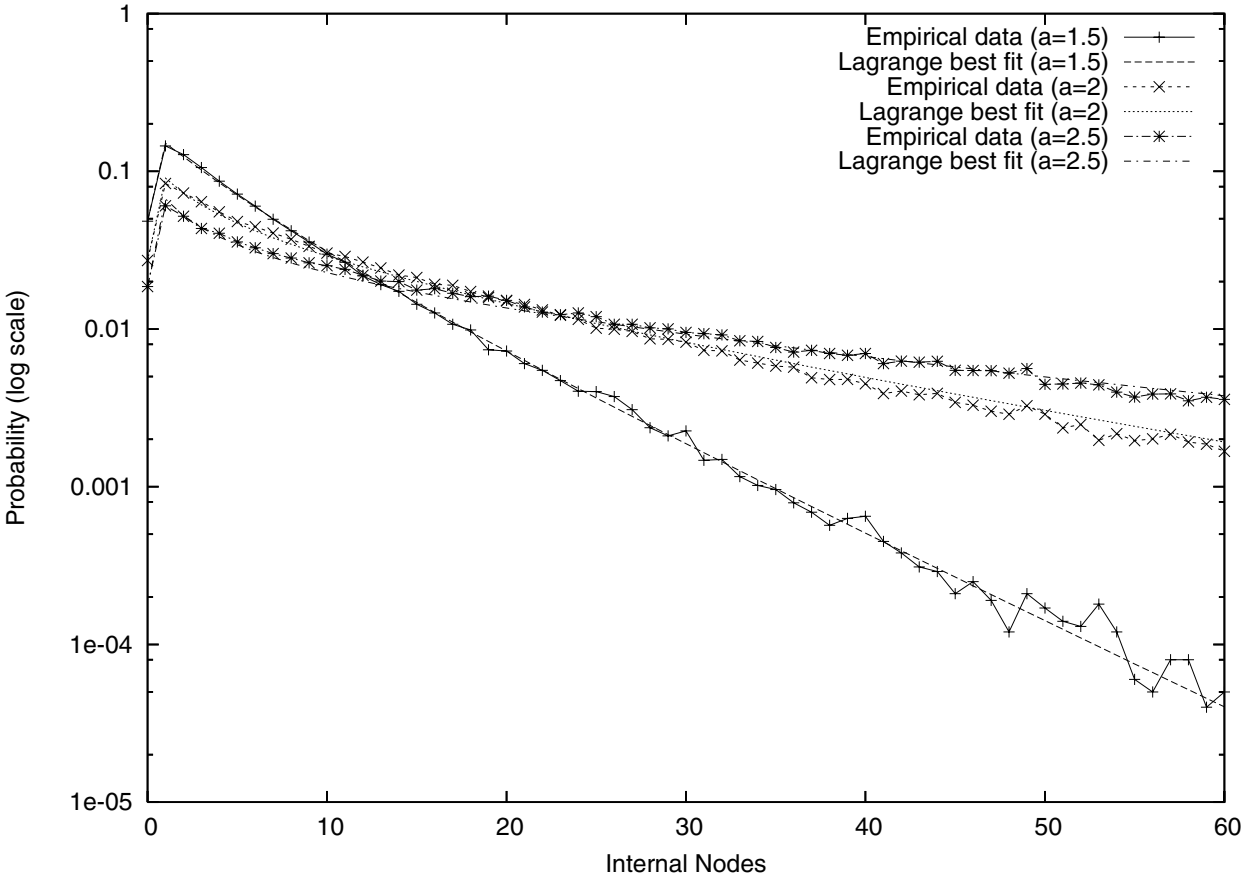

Figure 14: Comparison between empirical size distributions and modified Lagrange size distributions obtained by best fit for trees made up with primitives of average arity 1.5, 2 and 2.5 initialised with FULL method (depth=3, initial mean sizes $\mu_0 = 8.13$, $\mu_0 = 15.0$ and $\mu_0 = 25.36$, respectively) and manipulated by subtree crossover with 90%/10% node selection policy (mean sizes after 500 generations $\mu = 6.64$, $\mu = 15.69$ and $\mu = 28.44$, respectively).

subtree-swapping crossover: Part I. *Evolutionary Computation*, 11(1):53–66, March 2003.

[5] Sean Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 4(3):274–283, September 2000.

[6] Saeed Ghahramani. *Fundamentals of Probability*. Prentice-Hall Inc, Upper Saddle River, NJ 07458, 1996.

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

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

[9] Sean Luke. ECJ 13: A Java evolutionary computation library. http://cs.gmu.edu/∼eclab/projects/ecj/, 2005.

[10] Riccardo Poli. A simple but theoretically-motivated method to control bloat in genetic programming. In Conor Ryan, Terence Soule, Maarten Keijzer, Edward Tsang, Riccardo Poli, and Ernesto Costa, editors, *Genetic Programming, Proceedings of EuroGP'2003*, volume 2610 of *LNCS*, pages 204–217, Essex, 14-16 April 2003. Springer-Verlag.