Fitness Causes Bloat

W. B. Langdon and R. Poli
School of Computer Science,
The University of Birmingham,
Birmingham B15 2TT, UK
Email: {W.B.Langdon,R.Poli}@cs.bham.ac.uk
www: http://www.cs.bham.ac.uk/~wbl, ~rmp
Tel: +44 (0) 121 414 4791, Fax: +44 (0) 121 414 4281

Technical Report: CSRP-97-09

24 February 1997

Abstract

The problem of evolving an artificial ant to follow the Santa Fe trail is used to demonstrate the well known genetic programming feature of growth in solution length. Known variously as "bloat", "redundancy", "introns", "fluff", "Structural Complexity" with antonyms "parsimony", "Minimum Description Length" (MDL) and "Occam's razor". Comparison with runs with and without fitness selection pressure shows the tendency for solutions to grow in size is caused by fitness based selection. We argue that such growth is inherent in using a fixed evaluation function with a discrete but variable length representation. Since with simple static evaluation search converges to mainly finding trial solutions with the same fitness as existing trial solutions. In general variable length allows many more long representations of a given solution than short ones of the same solution. Thus with an unbiased random search we expect longer representations to occur more often and so representation length tends to increase. I.e. fitness based selection leads to bloat.

1 Introduction

The tendency for programs in genetic programming (GP) populations to grow in length has been widely reported [Tackett, 1993; Tackett, 1994; Angeline, 1994; Tackett, 1995; Langdon, 1995; Nordin and Banzhaf, 1995; Soule *et al.*, 1996] This tendency has gone under various names "bloat", "redundancy", "introns", "fluff", "Structural Complexity" and "parsimony", "Minimum Description Length" (MDL) and "Occam's razor" have been used to indicate the opposite. The principle explanation advanced for bloat has been the growth of "introns" i.e. code which has no effect on the operation of the program which contains it. ([Wu and Lindsay, 1996] contains a survey of recent research in biology on the original wetware "introns"). Such introns are said to protect the program containing them from crossover [Blickle and Thiele, 1994; Blickle, 1996; Nordin *et al.*, 1995; Nordin *et al.*, 1996]. [McPhee and Miller, 1995] presents an analysis of some simple GP problems designed to investigate bloat. This shows with some of their function sets longer programs can "replicate" more "accurately" when using crossover. [Rosca and Ballard, 1996a] provides a detailed anaysis of bloat using tree schemata specifically for GP.

In this paper we advance a more general explanation which should apply generally to any discrete variable length representation and generally to any progressive search technique. That is bloat is not specific to genetic programming applied to trees and tree based crossover but should also be found with other genetic operators and non-population based stochastic search techniques such as simulated annealing and stochastic iterated hill climbing.

In the next section we expand our argument that bloat is inherent in variable length representations such as GP. In Sections 3 and 4 we analyse a typical GP demonstration problem, showing it suffers from bloat but also showing that bloat is not present in the absence of fitness based selection. Section 5 describes the results we have achieved and this is followed in Section 6 by a discussion of the potential advantages and disadvantages of bloat and potential responses to it. Finally Section 7 summarises our conclusions.

2 Bloat in Variable Length Representations

In general with variable length discrete representations there are multiple ways of representing a given behaviour. If the evaluation function is static and concerned only with the quality of the partial solution (i.e. is independent of the representation) then these representations all have equal worth. If the search strategy (e.g. genetic operations) were unbiased, each of these would be equally likely to be found. In general there are many more long ways to represent a specific behaviour than short representations of the same behaviour. Thus we would expect a predominance of long representations. In practice search techniques commence with simple (i.e. short) representations, i.e. they have an in built bias in favour of short representations. Common search techniques have a bias in favour of continuing the search from previously discovered high fitness representations and retaining them as points for future search. I.e. there is a bias in favour of representations that do at least as well as their initiating point(s). On problems of interest, finding improved solutions is relatively easy initially but becomes increasingly more difficult. This is a widespread phenomenon, e.g. athletics world records show a consistent trend of smaller improvement as each new record is set. In these circumstances, especially with a discreet fitness function, there is little chance of finding a representation that does better than the representation(s) from which it was created (cf. "death of crossover" [Langdon, 1996a, page 222]). So the selection bias favours representations which have the same fitness as those from which they were created. In many cases the most likely way to create one representation from another and have the same fitness as it, is for the new representation to represent identical behaviour. Thus, in the absence of improved solutions, the search may become a random search for new representations of the best solution found so far. As we said above, there are many more long representations than short ones for the same solution, such a random search (other things being equal) will find more long representations than short ones. In GP this has become known as bloat.

3 The Artificial Ant Problem

The artificial ant problem is described in [Koza, 1992, pages 147–155]. Briefly the problem is to devise a program which can successfully navigate an artificial ant along a twisting trail on a square 32×32 toroidal grid. The program can use three operations, Move, Right and Left, to move the ant forward one square, turn to the right or turn to the left. Each of these operations take one time unit. The sensing function IfFoodAhead looks into the square the ant is currently facing and then executes one of its two arguments depending upon whether that square contains food or is empty. Two other functions, Prog2 and Prog3, are provided. These take two and three arguments respectively which are executed in sequence.

The artificial ant must follow the the "Santa Fe trail", which consists of 144 squares with 21 turns. There are 89 food units distributed non-uniformly along it. Each time the ant enters

Table 1: Ant Problem

Objective:	Find an ant that follows the “Santa Fe trail”
Terminal set:	Left, Right, Move
Functions set:	IfFoodAhead, Prog2, Prog3
Fitness cases:	The Santa Fe trail
Fitness:	Food eaten
Selection:	Tournament group size of 7, non-elitist, generational
Wrapper:	Program repeatedly executed for 600 time steps.
Population Size:	500
Max program:	500
Initial population:	Created using “ramped half-and-half” with a maximum depth of 6
Parameters:	90% one child crossover, no mutation. 90% of crossover points selected at functions, remaining 10% selected uniformly between all nodes.
Termination:	Maximum number of generations G = 50

a square containing food it eats it. The amount of food eaten is used as the fitness measure of the controlling program.

4 GP Parameters

Our GP system was set up to be the same as given in [Koza, 1992, pages 147–155] except the populations were allowed to continue to evolve even after an ant succeeded in traversing the whole trail, programs are restricted to a maximum length of 500 rather than to a maximum depth of 17, each crossover produces one child rather than two, tournament selection was used and the ants were allowed 600 operations (Move, Left, Right) to complete the trail. The details are given in Table 1, parameters not shown are as [Koza, 1994, page 655]. On each version of the problem 50 independent runs were conducted.

Note in these experiments we allow the evolved programs to be far bigger than required to solve the problem. For example the 100% correct solution given on [Koza, 1992, page 154] takes about 543 time steps to traverse the Santa Fe trail but has a length of only 18 nodes and this is not the most compact solution possible.

5 Results

5.1 Standard Runs

In 50 independent runs 6 found ants that could eat all the food on the Santa Fe trail within 600 time steps. The evolution of maximum and mean fitness averaged across all 50 runs is given in Figure 1. (In all cases the average minimum fitness is near zero). These curves show the GP fitness behaving as expected with both the maximum and average fitness rising rapidly initially but then rising more slowly later in the runs. The GP population converges in the sense that the average fitness approaches the maximum fitness. However the spread of fitness values of the children produced in each generation remains large and children which eat either no food or only one food unit are still produced even in the last generation.

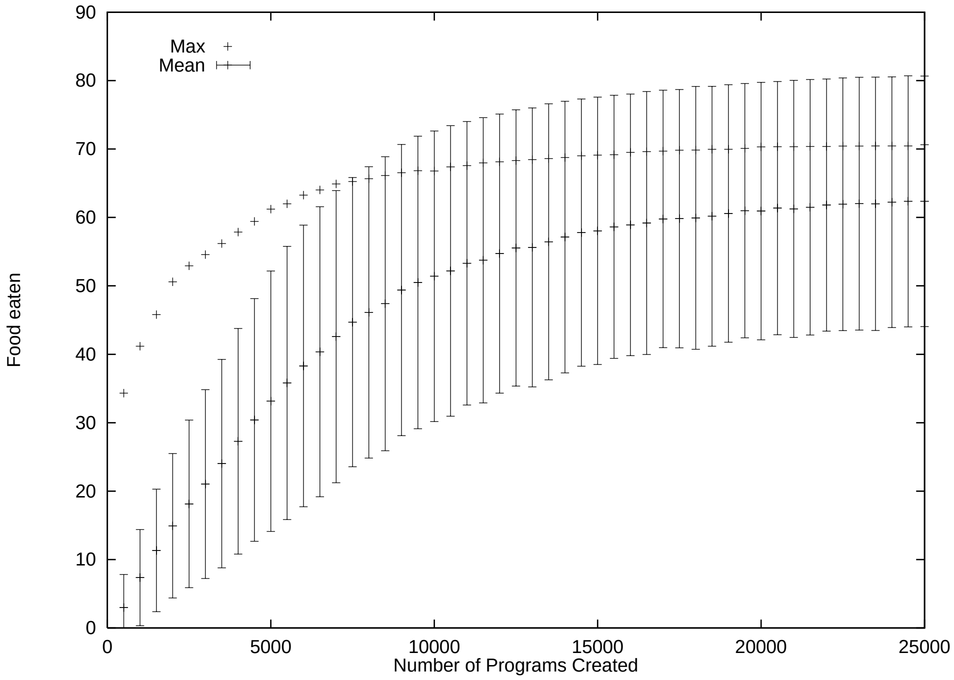

Figure 1: Evolution of maximum and population mean of food eaten. Error bars indicate one standard deviation. Means of 50 runs.

Figure 2 shows the evolution of maximum and mean program size averaged across all 50 runs. We see in the first four generations the average program length grows rapidly from an initial size of 23.5 to 73.5. Program size remains fairly static from generation 4 to generation 11, when it begins to grow progressively towards the maximum allowed program size of 500.

Surprisingly the mean program length grows faster than the size of the best program within the population and between generations 2 and 12 it exceeds it (cf. Section 5.1.1). In the later generations the mean program length and length of the best program on average lie close to each other. The size of the best program refers to that of a single individual. In generations with more than one individual program having the top score, one such program is chosen at random and labelled the "best". This random choice leads to random fluctuations in apparent program size which can be seen in Figure 2 after generation 27 despite averaging over 50 runs.

5.1.1 Price's Theorem Applied to Program Length

Price's Covariance and Selection Theorem [Price, 1970] from population genetics relates the change in frequency of a gene Δq in a population from one generation to the next, to the covariance of the gene's frequency in the original population with the number of offspring z produced by individuals in that population (see Equation 1). We have used it to help explain the evolution of the number of copies of functions and terminals in GP populations [Langdon, 1996a; Langdon and Poli, 1997].

$$\Delta q = \frac{\mathrm{Cov}(z, q)}{\overline{z}} \tag{1}$$

In our GP the size of the population does not change so $\overline{z} = 1$ and the expected number of children is given by the parent's rank so in large populations the expected change is approximately $\mathrm{Cov}(t(r/p)^{t-1}, q)$ as long as crossover is random. (t is the tournament size and r is each programs rank within the population of size p).

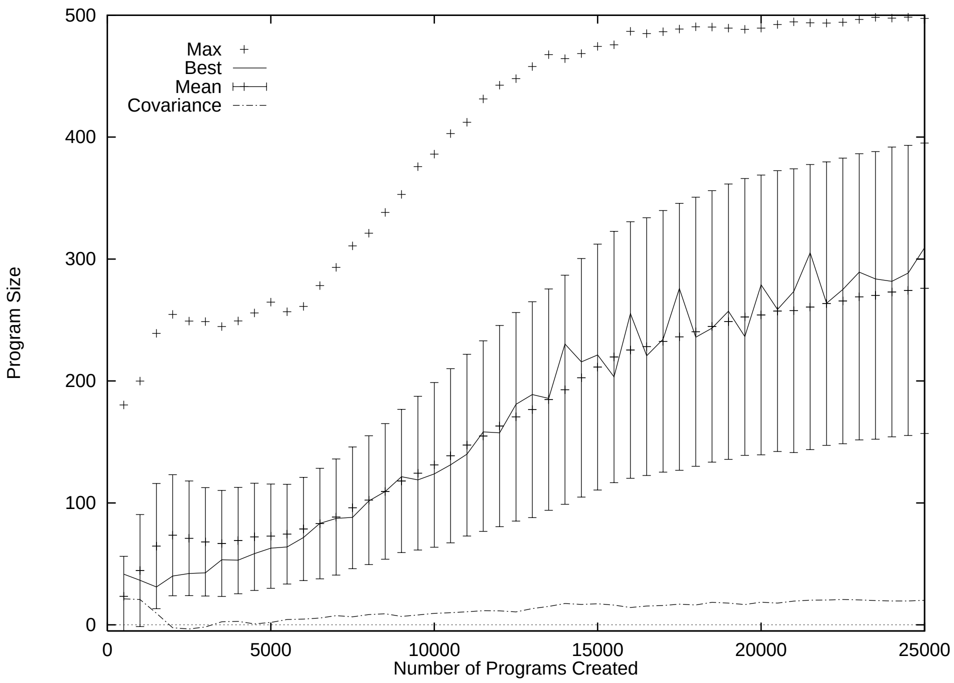

Figure 2: Evolution of maximum and population mean program length. Error bars indicate one standard deviation. Solid line is the length of the "best" program in the population, covariance of length and normalised rank based fitness shown dotted. Means of 50 runs.

In GP a program's length is inherited and so Equation 1 should hold. More formally Price's theorem applies (provided length and genetic operators are uncorrelated) since program length is a *measurement function* of the genotype [Altenberg, 1995, page 28]. If it held exactly a plot of covariance and change in frequency would be a diagonal line. The plot in Figure 3 shows good agreement between theory and measurement until generation 20. After Generation 20 the rise in program length is smaller than predicted. This is explained by the restriction on program length. From Figure 2 we can see after generation 20 (i.e. after 10000 programs have been created) the largest programs in the population are approaching the size limit and so further program growth is impeded.

Essentially Equation 1 gives a quantitative measurement of the way genetic algorithms (GAs) search. If some aspect of the genetic material is positively correlated with fitness then, other things being equal, the next generation's population will on average contain more of it. If it is negative, then the GA will tend to reduce it in the next generation.

For a given distribution of program lengths Equation 1 says the change in mean program length will be linearly related to the selection pressure. This provides some theoretical justification for [Tackett, 1994, page 112] claim that "average growth in size ... is proportional to selection pressure". Which is based upon experimental measurements with a small number of radically different selection and crossover operators on his "Royal Road" problem. (Solutions to the Royal Road problem are required to have prespecified syntactic properties which mean "that the optimal solution has a fixed size and does not admit extraneous code selections". Such solutions are not executable programs.)

Referring back to Figure 2 and considering the length of the best solution and the covariance in the first four generations. We see the covariance correctly predicts the mean length of programs will increase. In contrast if we assumed the GA would converge towards the individual with the best fitness in the population, the fact that the mean length is higher than the mean

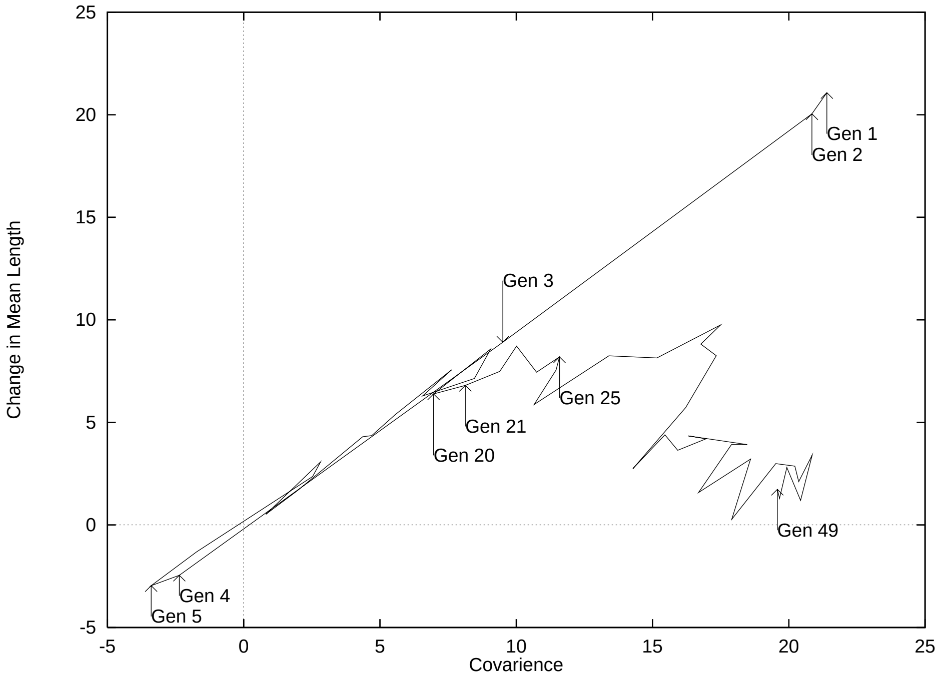

Figure 3: Covariance of program length and normalised rank based fitness v. change in mean length in next generation. Means of 50 runs.

length of the best individual would lead us to predict a fall in mean program length. That is Price's Theorem is the better predictor. This is because it considers the whole population rather than just one (albeit the best in the population).

Between generations 4 and 20 the mean program size can be reasonably described by a parabola, i.e. growth is quadratic. This corresponds with the covariance rising linearly (slope 0.76) with number of generations.

Where fitness selection is not used (as in the following sections), each individual in the population has the same fitness and so the covariance is always zero.

5.2 No Selection

A further 50 runs were conducted (using the same initial populations) and no fitness selection. Unsurprisingly no run found a solution and the maximum, mean and other fitness statistics fluctuate a little but are essentially unchanged (cf. Figure 4). Similarly program size statistics fluctuate but are essentially the same as those of the initial population (cf. Figure 5). This is in keeping with results with no selection reported in [Tackett, 1994, page 112]. Given crossover produces random changes in length we might have expected the spread of lengths to gradually increase. This is not observed. The slow fall in maximum program size can be seen in Figure 5. There is also a small fall in mean standard deviation from 32.8 in the initial population to 27.3 at the end of the runs.

5.3 Removing Selection

A final 50 runs were conducted in which fitness selection was removed after generation 25 (i.e. these runs are identical to those in Section 5.1 up to generation 25). Of the six runs which found ants that could eat all the food on the Santa Fe trail within 600 movements, four where found before generation 26 and of these three retained such programs until generation 50 while

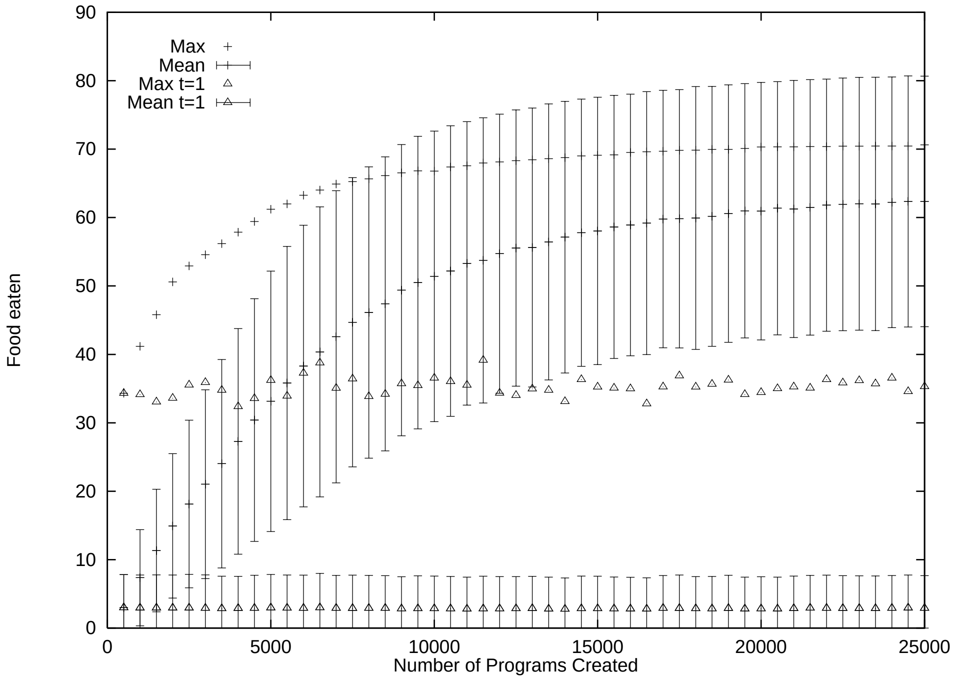

Figure 4: Evolution of maximum and population mean of food eaten. Error bars indicate one standard deviation. Means of 50 runs comparing tournament sizes of 7 and 1.

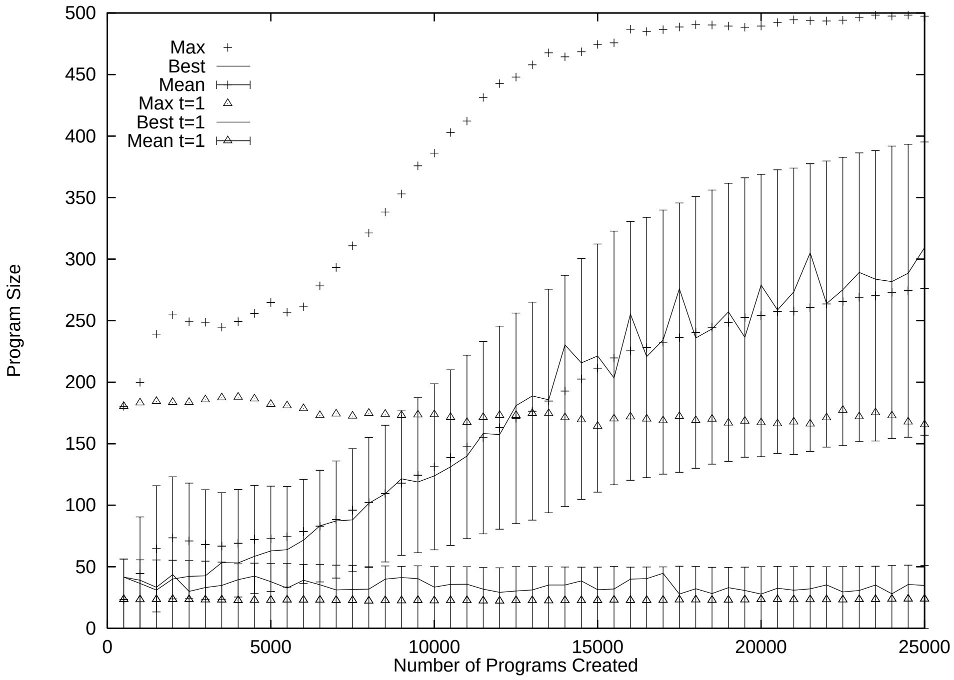

Figure 5: Evolution of maximum and population mean program length. Error bars indicate one standard deviation. Solid line is the length of the "best" program in the population. Means of 50 runs comparing tournament sizes of 7 and 1.

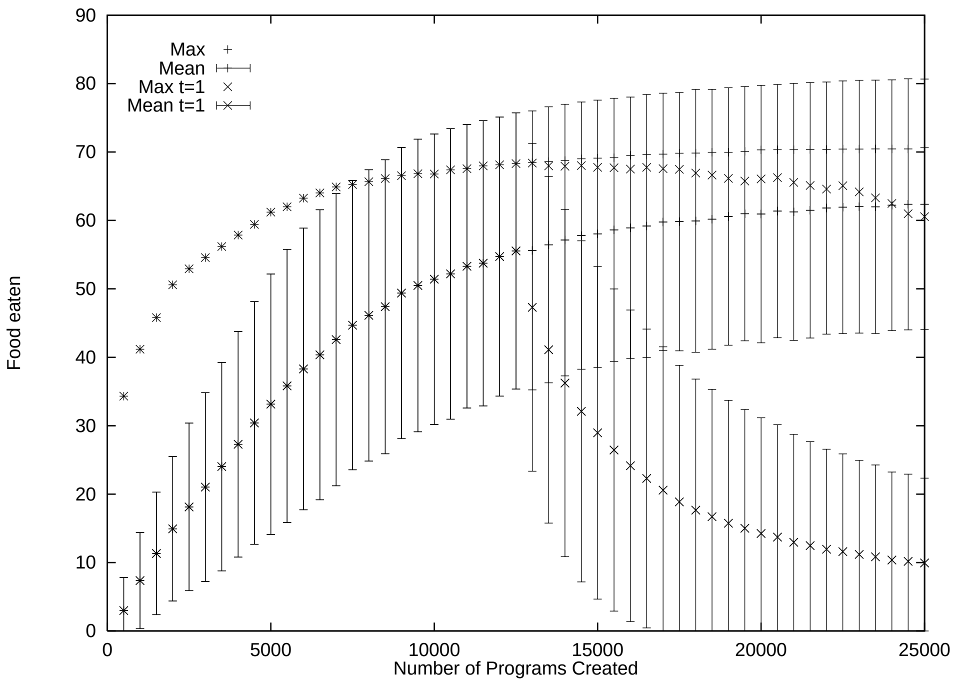

Figure 6: Evolution of maximum and population mean of food eaten. Error bars indicate one standard deviation. Means of 50 runs showing effect of removing fitness selection.

in one case the maximum fitness dropped below 89. In another run a solution was found in generation 41 but no other generations of this run contained solutions.

The evolution of maximum and mean fitness averaged across all 50 runs is given in Figure 6. As expected Figure 6 shows in the absence of fitness selection the fitness of the population quickly falls. (The average fitness in last generation is 9.9, i.e. 3.3 times the mean fitness of the initial populations).

After fitness selection is removed, if the crossover operator is unbiased, we expect the length of programs to fluctuate at random about the lengths of their parents. So the mean length should change little and the spread of lengths should increase. The length restriction means the crossover operator is biased because it must ensure the child it produces does not exceed the length limit. Where an unbiased choice of crossover points would cause the offspring to be too large, the roles of the two parents are reversed and a shorter (and legal) program is produced instead. This bias (which is more important as programs start to approach the length limit) explains the slow fall in program size seen in Figure 7. The spread of programs' lengths can also be seen. The average standard deviation rises from 92.3 in generation 26 to 107.6 at the end of the run. (The rise might be larger if lengths were not constrained to lie in the range 1 ... 500).

5.4 Correlation of Fitness and Program Size

Figure 8 plots to correlation coefficient of program size and amount of food eaten by the ant it controls. (Correlation coefficients are equal to the covariance after it has been normalised to lie in the range $-1 \ldots +1$. By considering food eaten we avoid the intermediate stage of converting program score to expected number of children required when applying Price's Theorem). Figure 8 shows in all cases there is a very strong correlation (typically $+0.3$) between program performance and length of programs. Where selection is not used longer programs

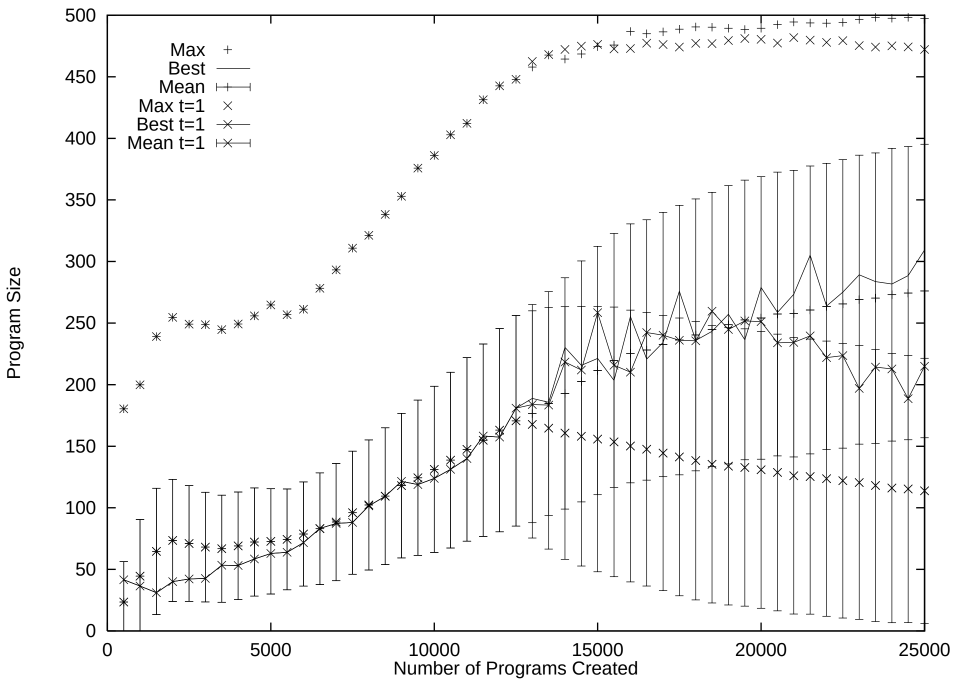

Figure 7: Evolution of maximum and population mean program length. Error bars indicate one standard deviation. Solid line is the length of the "best" program in the population. Means of 50 runs showing effect of removing fitness selection.

may be more fit simply because they are more likely to contain useful primitives (such as Move) than short programs. When selection is used the correlation may be because when crossover reduces the length of a program it is more likely to disrupt the operation of the program and so reduce its fitness.

5.5 Effect of Crossover on Fitness

As expected in the initial generations crossover is highly disruptive (cf. Figure 9) with only 19.5% of crossovers producing a child with the same score as its first parent (i.e. the parent from which it inherits the root of its tree). However after generation 4 this proportion grows steady, by generation 18 more than half have the same fitness. At the end of the run this has risen to 69.8%.

The range of change of fitness is highly asymmetric; many more children are produced which are worse than their parent than those that are better. By the end of the run, no children are produced with a fitness greater than their parent. Similar behaviour has been reported on other problems [Nordin *et al.*, 1996] [Rosca and Ballard, 1996b, page 183] [Langdon, 1996a, Chapter 7].

5.6 Non-Disruptive Crossover and Program Length

In Section 2 we argued that there are more long programs with a given performance than short ones and so a random search for programs with a given level of performance is more likely to find long programs. We would expect generally (and it has been reported on a number of problems) that crossover produces progressively fewer improved solutions as evolution proceeds and instead in many problems selection drives it to concentrate upon finding solutions with the same fitness. Apart from the minimum and maximum size restrictions crossover is random so

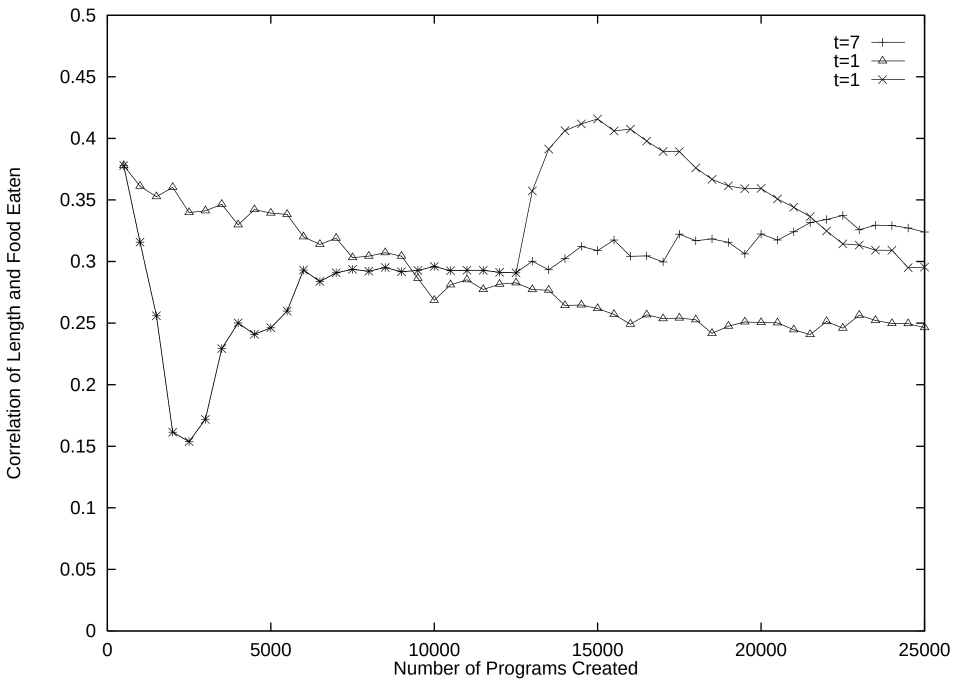

Figure 8: Correlation of program length and fitness, Normal runs, runs without selection and runs with selection removed halfway through. Means of 50 runs.

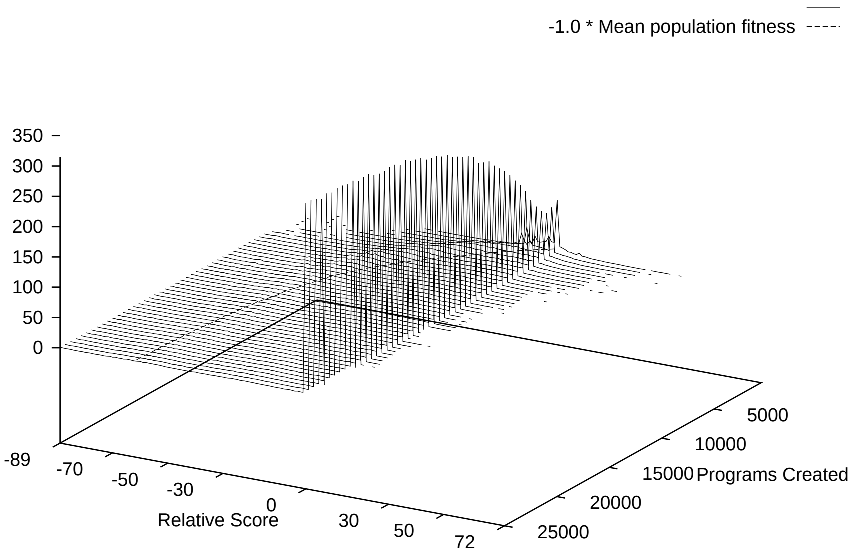

Figure 9: Fitness relative to First Parent, Normal runs, Means of 50 runs.

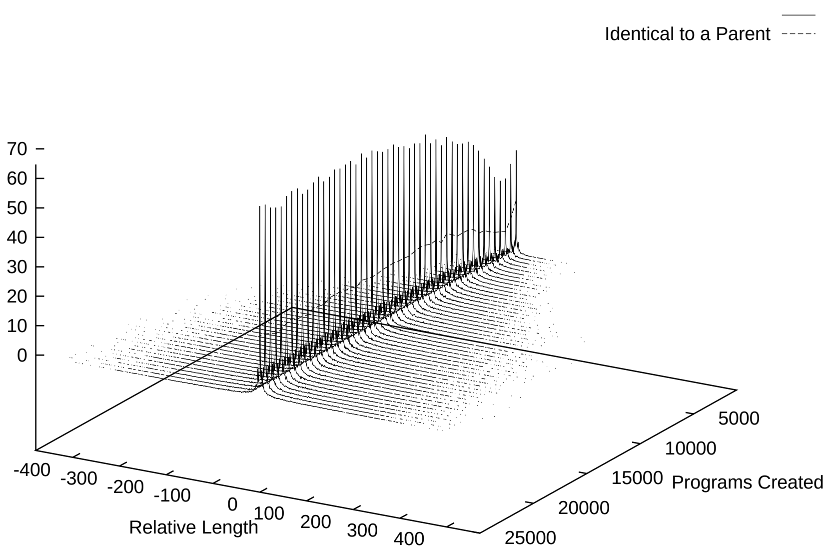

Figure 10: Change in length of offspring relative to first parent where score is the same. Normal runs, means of 50 runs.

we would expect to see it finding more long solutions than short ones. The change in program length of programs produced by crossover which have the same fitness as their first parent is plotted in Figure 10. At first sight Figure 10 is disappointingly symmetric with a large central spike of programs with the same length and rapidly falling tails either side of zero. (Initially 8% of crossovers produce a child which has the same length and fitness as its first parent. This proportion falls to a minimum of 6% in generation 4 and then rises progressively to 15% at the end of the run. 5% of crossovers produce a child which is identical to one or other of its parents, cf. dashed line on Figure 10).

Closer examination of the data in Figure 10 reveals that on average (cf. Figure 11) there is a bias towards longer programs. On average programs with the same fitness as their first parent are always longer than it. From generation 4 to 23 they are on average about 3 program nodes longer. Presumably due to the average program length approaching the upper limit, the bias falls to about 1 after generation 33 but remains positive.

Part of the reason for the central peak of Figure 10 is a result of another aspect of GP crossover; almost all the genetic material is inherited from the first parent. Crossover points are typically drawn from close to the leafs of both parents simply because for large bushy trees most parts of a tree are close to its leafs. While the expected size of crossover fragments depends in detail upon the trees selected as parents and the relative weighting applied to functions and terminals cf. Table 1, typically both the inserted subtree and the subtree it replaces consist of a function and its leafs. Since these subtrees are short together they produce a small change in total size. It seems likely that small changes are less likely to effect fitness than big ones, hence the spike in Figure 10. Inheriting principally from one parent is in sharp contrast to more traditional GAs (and indeed sexual reproduction in "higher organisms" in nature) where offspring receive about the same amount of genetic material from each parent.

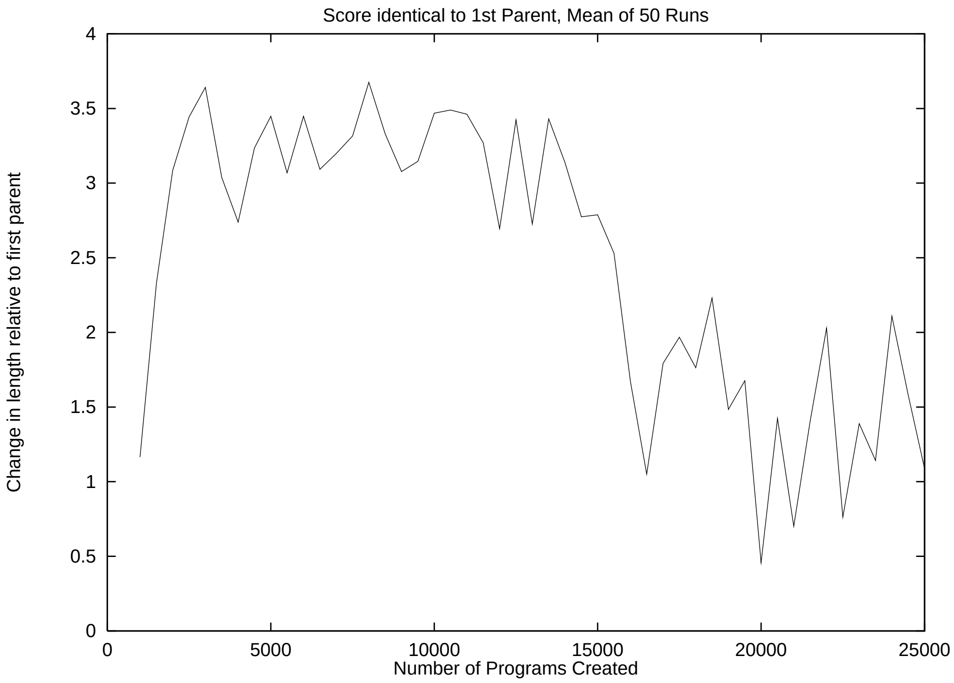

Figure 11: Mean change in length of offspring relative to first parent where score is the same. Normal runs, means of 50 runs.

6 Discussion

In the artificial ant problem every leaf that is executed has a potential detrimental impact on the programs fitness because it uses up $1/600^{th}$ of the time available to solve the problem. This might prevent some other food eating action from happening later. Since longer programs tend to contain more leafs this might introduce a parsimony pressure. However the programs are repeatedly executed until the number of time steps is reached. Thus every program no matter what its size will execute the same number of leafs. It is possible to complete the trail in less than 600 time steps. This potentially allows programs additional freedom to explore the grid around the trail with out attracting additional penalties for missing food units. I.e. multiple paths may have the same fitness.

It is possible for a IfFoodAhead branch instruction to always execute the same branch, in which case leafs in the other branch are never executed (i.e. they are in an intron). Changes made by crossover to that branch of the program tree are never executed and have absolutely no effect on the ants behaviour. This gives a ready and easily visualised mechanism for crossover to produce new representations with identical performance to the parent they where produced from. In other variable length representations the mechanisms for producing representations with identical fitness may not be so obvious but, in general, multiple solutions with the same cost will be possible.

6.1 Do we Want to Prevent Bloat?

From a practical point of view the machine resources consumed by any system which suffers from bloat will prevent extended operation of that system. However in practice we may not wish to operate the system continually. For example it may quickly find a satisfactory solutions or better performance may be achieved by cutting short its operation and running it repeatedly with different starting configurations [Koza, 1992, page 758].

In some data fitting problems growth in solution size may be indicative of "over fitting", i.e. better matching on the test data but at the expense of general performance. For example [Tackett, 1993, page 309] suggests "parsimony may be an important factor not for 'aesthetic' reasons or ease of analysis, but because of a more direct relationship to fitness: there is a bound on the 'appropriate size' of solution tree for a given problem".

By providing a "defence against crossover" [Nordin *et al.*, 1996, page 118] bloat causes the production of many programs of identical performance. These can consume the bulk of the available machine resources and by "clogging up" the population may prevent GP from effectively searching for better programs.

On the other hand [Angeline, 1994, page 84] quotes results from fixed length GAs in favour of representations which include introns, to argue we should "not ... impede this emergent property [i.e. introns] as it may be crucial to the successful development of genetic programs". Introns may be important as "hiding places" where genetic material can be protected from the current effects of selection and so retained in the population. This may be especially important where fitness criteria are dynamic. A change in circumstance may make it advantageous to execute genetic material which had previously been hidden in an intron.

In complex problems it may not be possible to test every solution on every aspect of the problem and some form of dynamic selection of test cases may be required [Gathercole and Ross, 1994]. For example in some cases Co-evolution has been claimed to be beneficial to GP. If the fitness function is sufficiently dynamic, will there still be an advantage for a child in performing identically to its parents? If not will we still see such explosive bloat?

6.2 Three Ways to Control Bloat

Three methods of controlling bloat have been suggested. Firstly, and most widely used (e.g. in these experiments), is to place a universal upper bound either on tree depth [Koza, 1992] or program length. ([Gathercole and Ross, 1996; Langdon and Poli, 1997] discuss unexpected problems with this approach).

The second (also commonly used) is to incorporate program size directly into the fitness measure (often called parsimony pressure) [Koza, 1992; Zhang and Mühlenbein, 1993; Iba *et al.*, 1994]. [Rosca and Ballard, 1996a] give an analysis of the effect of parsimony pressure which varies linearly with program length. Multi-objective fitness measures where one objective is compact or fast programs have also been used [Langdon, 1996b].

The third method is to tailor the genetic operations. [Sims, 1993, page 469] uses several mutation operators but adjusts their frequencies so a "decrease in complexity is slightly more probable than an increase". [Blickle, 1996] suggests targeting genetic operations at redundant code. This is seldom used, perhaps due to the complexity of identifying redundant code. [Soule *et al.*, 1996] showed bloat continuing despite their targetted genetic operations. Possibly this was because of the difficulty of reliably detecting introns. I.e. there was a route whereby the GP could evolve junk code which masqueraded as being useful and thereby protected itself from removal. While [Rosca and Ballard, 1996a] propose a method where the likelihood of potentially disruptive genetic operations increases with parent size.

7 Conclusions

We have generalised existing explanations for the widely observed growth in GP program size with successive generations (*bloat*) to give a simple statistical argument which should be generally applicable both to GP and other systems using discreet variable length representations and static evaluation functions. Briefly, in general simple static evaluation functions quickly drive search to converge, in the sense of mainly finding trial solutions with the same fitness as previously found trial solutions. In general variable length allows many more long representations of a given solution than short ones of the same solution. Thus with an unbiased random search we expect longer representations to occur more often and so representation length tends to increase. I.e. fitness based selection leads to bloat.

In Sections 3, 4 and 5 we have taken a typical GP problem and demonstrated with fitness selection it suffers from bloat whereas without selection it does not. We have demonstrated that if fitness selection is removed, there is a slight bias in common GP crossover (caused by the practical requirement to limit bloat) which causes a slow reduction in program size. NB fitness causes bloat inspite of a small crossover bias in favour of parsimony. Detailed measurement of crossover confirms after an extended period of evolution, most crossovers are not disruptive (i.e. most children have the same fitness as their parents). It also shows children with the same fitness as their parents are on average consistently longer than them.

In Section 5.1.1 we apply Price's Theorem for the first time to program lengths within GP populations. We confirm experimentally that it fits GP populations unless restrictions on program size have significant impact. In Section 6 we discussed the merits of using the ant problem as a test bed, the circumstances in which we need to control bloat and and current mechanisms which do control it.

Acknowledgements

This research is supported by the Defence Research Agency in Malvern and the British Council.

References

[Altenberg, 1995] Lee Altenberg. The Schema Theorem and Price's Theorem. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 23–49, Estes Park, Colorado, USA, 31 July–2 August 1994 1995. Morgan Kaufmann.

[Angeline, 1994] Peter John Angeline. Genetic programming and emergent intelligence. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 4, pages 75–98. MIT Press, 1994.

[Blickle and Thiele, 1994] Tobias Blickle and Lothar Thiele. Genetic programming and redundancy. In J. Hopf, editor, *Genetic Algorithms within the Framework of Evolutionary Computation (Workshop at KI-94, Saarbrücken)*, pages 33–38, Im Stadtwald, Building 44, D-66123 Saarbrücken, Germany, 1994. Max-Planck-Institut für Informatik (MPI-I-94-241).

[Blickle, 1996] Tobias Blickle. *Theory of Evolutionary Algorithms and Application to System Synthesis.* PhD thesis, Swiss Federal Institute of Technology, Zurich, November 1996.

[Gathercole and Ross, 1994] Chris Gathercole and Peter Ross. Dynamic training subset selection for supervised learning in genetic programming. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature III*, pages 312–321, Jerusalem, 9-14 October 1994. Springer-Verlag.

[Gathercole and Ross, 1996] Chris Gathercole and Peter Ross. An adverse interaction between crossover and restricted tree depth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings* *of the First Annual Conference*, pages 291–296, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[Iba *et al.*, 1994] Hitoshi Iba, Hugo de Garis, and Taisuke Sato. Genetic programming using a minimum description length principle. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 12, pages 265–284. MIT Press, 1994.

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

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

[Langdon and Poli, 1997] W. B. Langdon and R. Poli. Price's theorem and the MAX problem. Technical Report CSRP-97-4, University of Birmingham, School of Computer Science, January 1997.

[Langdon, 1995] W. B. Langdon. Evolving data structures using genetic programming. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 295–302, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[Langdon, 1996a] W. B. Langdon. *Data Structures and Genetic Programming*. PhD thesis, University College, London, 27 September 1996.

[Langdon, 1996b] William B. Langdon. Data structures and genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 20, pages 395–414. MIT Press, Cambridge, MA, USA, 1996.

[McPhee and Miller, 1995] Nicholas Freitag McPhee and Justin Darwin Miller. Accurate replication in genetic programming. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 303–309, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[Nordin and Banzhaf, 1995] Peter Nordin and Wolfgang Banzhaf. Complexity compression and evolution. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[Nordin *et al.*, 1995] Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Justinian P. Rosca, editor, *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 6–22, Tahoe City, California, USA, 9 July 1995.

[Nordin *et al.*, 1996] Peter Nordin, Frank Francone, and Wolfgang Banzhaf. Explicitly defined introns and destructive crossover in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 6, pages 111–134. MIT Press, Cambridge, MA, USA, 1996.

[Price, 1970] George R. Price. Selection and covariance. *Nature*, 227, August 1:520–521, 1970.

[Rosca and Ballard, 1996a] Justinian P. Rosca and Dana H. Ballard. Complexity drift in evolutionary computation with tree representations. Technical Report NRL5, University of Rochester, Computer Science Department, Rochester, NY, USA, December 1996.

[Rosca and Ballard, 1996b] Justinian P. Rosca and Dana H. Ballard. Discovery of subroutines in genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 9, pages 177–202. MIT Press, Cambridge, MA, USA, 1996.

[Sims, 1993] K. Sims. Interactive evolution of equations for procedural models. *The Visual Computer*, 9:466–476, 1993.

[Soule *et al.*, 1996] Terence Soule, James A. Foster, and John Dickinson. Code growth in genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 215–223, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[Tackett, 1993] Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[Tackett, 1994] Walter Alden Tackett. *Recombination, Selection, and the Genetic Construction of Computer Programs.* PhD thesis, University of Southern California, Department of Electrical Engineering Systems, 1994.

[Tackett, 1995] Walter Alden Tackett. Greedy recombination and genetic search on the space of computer programs. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 271–297, Estes Park, Colorado, USA, 31 July–2 August 1994 1995. Morgan Kaufmann.

[Wu and Lindsay, 1996] Annie S. Wu and Robert K. Lindsay. A survey of intron research in genetics. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving From Nature IV. Proceedings of the International Conference on Evolutionary Computation*, volume 1141 of *LNCS*, pages 101–110, Berlin, Germany, 22-26 September 1996. Springer-Verlag.

[Zhang and Mühlenbein, 1993] Byoung-Tak Zhang and Heinz Mühlenbein. Evolving optimal neural networks using genetic algorithms with Occam's razor. *Complex Systems*, 7:199–220, 1993.