# Price's Theorem and the MAX Problem

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/
Tel: +44 (0) 121 414 4791, Fax: +44 (0) 121 414 4281

Technical Report: CSRP-97-4
13 January 1997

**Abstract**

**We present a detailed analysis of the evolution of GP populations using the problem of finding a program which returns the maximum possible value for a given terminal and function set and a depth limit on the program tree (known as the MAX problem). We confirm the basic message of** [Gathercole and Ross, 1996] **that crossover together with program size restrictions can be responsible for premature convergence to a sub-optimal solution. We show that this can happen even when the population retains a high level of variety and show that in many cases evolution from the sub-optimal solution to the solution is possible if sufficient time is allowed. In both cases theoretical models are presented and compared with actual runs.**

**Experimental evidence is presented that Price's Covariance and Selection Theorem can be applied to GP populations and the practical effect of program size restrictions are noted. Finally we show that covariance between gene frequency and fitness in the first few generations can be used to predict the course of GP runs.**

## 1 Introduction

While genetic programming (GP) has been demonstrated on a wide and increasing range of applications comparatively little work has been devoted to a theory of GP. However such a theory is of great importance to the field as a whole. Work so far has been principly split between extending the Schema Theorem [Holland, 1992] from Genetic Algorithms to GP [Koza, 1992; O'Reilly and Oppacher, 1995; Whigham, 1995; Poli and Langdon, 1997], studying benchmark problems (e.g. Royal Trees [Punch *et al.*, 1996]) and applying population genetics to GP [Altenberg, 1994; Altenberg, 1995]. In this paper we combine the later two approaches by applying Price's Theorem from theoretical biology to the MAX problem.

[Gathercole and Ross, 1996] introduce the MAX problem to GP to highlight deficiencies in the standard GP crossover operator which result from the practical requirement to limit the size of evolved programs. They concentrate on the case where program trees are restricted to a maximum depth but suggest similar effects will be seen when programs are restricted to a maximum number of nodes.

In the MAX problem "the task is to find the program which returns the largest value for a given terminal and function set and with a depth limit, D, where the root node counts as depth 0" [Gathercole and Ross, 1996, page 291]. In this paper we use the function set $\{+, \times\}$ and there is one terminal 0.5. For a tree to produce the largest value + nodes must be used

Table 1: MAX Problem

| | |
|---|---|
| Objective: | Find a program that returns the largest value |
| Primitives: | $+, \times, 0.5$ |
| Max depth<br><br>Init depth | 3 ... 8 (NB root node is depth 0)<br>5 or Max depth |
| Fitness: | Value of tree |
| Selection: | Tournament group size of 6, generational plus elitism |
| Parameters | Pop = 200, G = 500, 99.5% crossover, no mutation. Crossover points selected uniformly between nodes. |

with 0.5 to assemble subtrees with the value of 2.0. These can then be connected via either $+$ or $\times$ to give 4.0. Finally the rest of the tree needs to be composed only of $\times$ nodes to yield the maximum value of $4^{2^{D-3}}$.

Section 2 describes the GP used in our experiments, Section 3 gives the results obtained and provides a detailed analysis and comparison with [Gathercole and Ross, 1996]. Section 4 shows typically the population retains a high level of diversity and presents models of the variety in the initial population and its subsequent evolution. Section 5 describes Price's Theorem, presents experimental evidence on its applicability to GP and shows it can be used to predict the behaviour of subsequent generations. Some benchmarks for the GP system used in these experiments are contained in Appendix A.

## 2 GP Parameters

Our GP system was set up to be the same as given in [Gathercole and Ross, 1996]. 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.

[Gathercole and Ross, 1996] do not describe how the initial population was created. In this paper two sets of experiments are presented. In the first the usual ramped half-and-half method [Koza, 1992, page 93] which creates random trees with a depths between 2 and 6 (i.e. D = 1 ... 5) was used. (However the initial trees obeyed the problem specific height restriction). A second set of runs (with D > 5) where made with the maximum tree height in the initial population identical to the problem specific limit. (Unless otherwise stated the discussion refers to this second set of experiments).

## 3 Results

Figure 1 shows the mean number of generations taken by the GP to solve the MAX problem in the successful runs. The percentage of unsuccessful runs is given in Figure 2. These results are similar to those in [Gathercole and Ross, 1996] except [Gathercole and Ross, 1996, Figures 3 and 4] do not contain data for depths of 7 or 8.

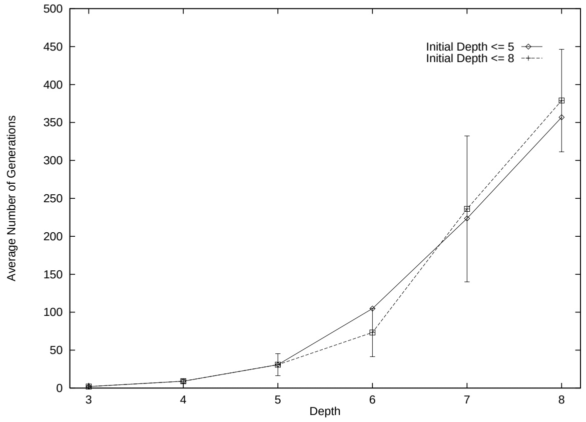


Figure 1: Mean number of generations need by the successful runs. Error bars indicate one standard deviation.

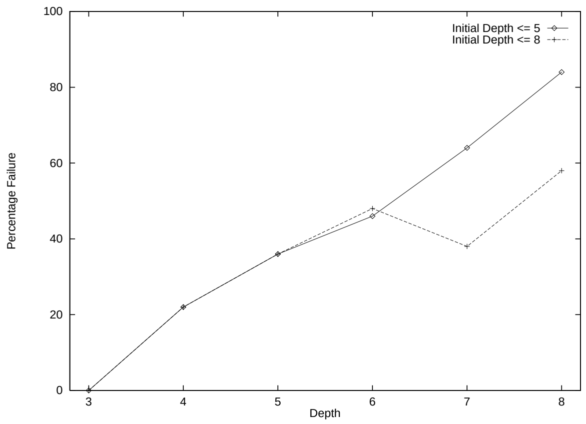


Figure 2: Percentage of runs ending in failure

### 3.1 Analysis of the Crossover Operator

The size of programs within the population grows quickly until almost all individuals are full trees. Due to the depth restriction crossover cannot make the trees any bigger and so crossover fragments are either the same size as the code they are replacing or smaller. In the case of full trees this means crossover can move code at the same level in the trees or higher but not lower.

In a typical unsuccessful run with D = 8 a third of crossovers are rejected because the subtree to be inserted would cause the offspring to violate the maximum depth. In these cases the roles of the two parents are reversed and a shorter subtree is inserted instead. Of the remaining two thirds half (i.e. $\frac{1}{3}$ of the total) result in replacing a subtree with one of the same height, so in total two thirds of crossovers replace a subtree with a shorter one. Only 0.74% of crossovers resulted in a taller subtree replacing a shorter one. These occur throughout most of the run, half occurring by generation 85 and the last in generation 441 (The mean generation is 105 and the standard deviation is 75).

[Gathercole and Ross, 1996, page 295] suggest the reason why GP finds the MAX problem hard is that the population quickly finds suboptimal solutions which contain + nodes near the root of the tree. To improve on such solutions crossover must replace them with $\times$ nodes. They suggest that "once the trees have reached the depth limit, the only way the higher levels are affected is through the promotion of lower level subtrees, which contain no $\times$ nodes, and the movement of subtrees within the same level". However the lack of $\times$ nodes in lower levels was not observed in our populations. For example at the end of all but one of the 50 runs with D = 8 the population contains thousands of $\times$ nodes at levels 3 and 4, even in runs where levels 0, 1 or 2 contain large numbers of + nodes.

Crossover readily moves such nodes to higher levels. To explain why the suboptimal + nodes are able to remain in the population we need to consider the fitness of the resulting offspring. Suppose both parents are near optimal (i.e. they consist mainly of $+, \times$ and 0.5 at the right level of the tree). The value of each subtree with a $\times$ as its root node is $4^{2^{D_\times - 3}}$ (where $D_\times$ is its depth) while if rooted with a + node it is $2 \times 4^{2^{D_+ - 4}}$. If an otherwise optimal subtree with a + as its root node is replaced with a optimal subtree from a lower level (i.e. $D_+ > D_\times$) then the value of this part of the offspring is reduced. Given the other subtrees at the same level are also near optimal (i.e. have values $\geq 1$) the value (and hence fitness) of the offspring as a whole will be less than that of its parent by a factor of at least $\frac{1}{2}$. As a high selection pressure is used (NB tournament size of 6) the offspring is unlikely to have children in the next generation. In contrast a crossover that moved subtrees at the same level is likely to produce an offspring with same fitness as its parent and thus is likely to have children itself. Thus high selection pressure and the problem representation conspire with the depth restriction to preserve sub-optimal programs.

[Gathercole and Ross, 1996, page 295] says "if there are no $\times$ nodes in any tree at a particular high level, it [is] now impossible for crossover to introduce a $\times$ node to this level; the population has converged to being duplicates of a sub-optimal tree, and no further improvement is possible." However (taking the example with D = 8 again) in only 7 cases does the population at the end of the run contain zero $\times$ nodes at any level from 0 to 3, i.e. in 22 of the runs that failed the population did contain $\times$ nodes in all of the higher levels. (The impossibility of the 7 runs succeeding was confirmed experimentally by running them again but this time to 5,000 generations. None succeeded, however in one case the population did improve finding better suboptimal solutions by replacing + nodes with $\times$ nodes where these were available at the same level).

The reason why the 22 runs failed (and why the successful runs took so long) despite having $\times$ nodes available is in part due to the low level of crossover activity near the root of the trees.

That is crossover is able to improve suboptimal trees but it takes a long time. We can estimate how long with the following model.

Assume the population has converged to a suboptimal tree containing $n_{+d}$ + and $n_{\times d}$ $\times$ nodes at level $d$ ($d < D - 3$) (so $n_{+d} + n_{\times d} = 2^d$). Each individual will be a full (or nearly full) binary tree of height D and so contain $2^{D+1} - 1$ nodes. Therefore the chance of selecting one of the + nodes at level $d$ to be the crossover point is

$$\frac{n_{+d}}{2^{D+1} - 1}$$

and the chance of replacing it with a $\times$ node from the same level is

$$\frac{n_{\times d}}{2^{D+1} - 1}$$

Firstly we note that the chance of improvement is much higher with large $d$. I.e. the nearer to the root the + node is the harder it will be for crossover to shift (in the case of the root node, it is impossible as there can be no $\times$ nodes also at the root). Secondly improvement is easiest when the number of + nodes is equal to the number of $\times$ nodes and the last + node at each level is the most difficult to remove. Also note that if the number of $\times$ nodes is small it will be difficult to increase it.

As replacing a + node at any of the higher levels will produce an improvement, by only considering crossovers which find improved solutions by moving subtrees at the same level as the + nodes we can form a lower bound on the overall chance of the offspring being an improvement:

$$\sum_{d=0}^{D-4} \frac{n_{+d} n_{\times d}}{(2^{D+1} - 1)^2} = \sum_{d=0}^{D-4} \frac{n_{+d}(2^d - n_{+d})}{(2^{D+1} - 1)^2}$$

Crossover closer to the root can also find improved solutions by replacing trees containing + nodes with others where they are replaced by $\times$ nodes, including where the inserted subtree contains + nodes further from the root. In general where there are + nodes at different levels deciding how many crossover points will yield an improved solution becomes complex. However if there are few incorrectly positioned + nodes we need only consider crossovers at the same level. For the special case of a single + node at level $d$ then crossover at any node connecting it to the root with a different node at the same level will yield the optimal solution. There are

$$\sum_{i=1}^{d} 2^i - 1 = 2^{d+1} - d - 2$$

such crossover point pairs. Note in this case the exact figure is about twice the approximation of $n_{\times d}$.

As each crossover is independent, the number required to replace a + node with a $\times$ node will be Poissonly distributed with a mean of

$$\frac{(2^{D+1} - 1)^2}{\sum_{d=0}^{D-4} n_{+d}(2^d - n_{+d})}$$

Thus the expected number of generations until the next improved solution is found in a population which has converged towards a good but sub-optimal solution is

$$\frac{(2^{D+1} - 1)^2}{p\ p_c \sum_{d=0}^{D-4} n_{+d}(2^d - n_{+d})}$$

Table 2: Generations at which improved solutions were found in an extended run (D = 8). The estimated number of crossover pairs which give improved solutions and the corresponding expected number of generations between improved solutions are given, for both the simple $\sum n_+(2^d - n_+)$ model and for the detail counting model.

| depth | Gen | $\Delta$Gen | $\sum n_+ \ldots$ | Pred | Count | Pred |
|---|---|---|---|---|---|---|
| 4 | 523 | 23 | 55 | 24 | 73 | 18 |
| 4 | 542 | 19 | 46 | 29 | 65 | 20 |
| 4 | 569 | 27 | 35 | 37 | 51 | 35 |
| 3 | 584 | 15 | 21 | 62 | 34 | 39 |
| 4 | 611 | 27 | 15 | 87 | 26 | 50 |

where $p$ is the population size and $p_c$ is the crossover probability (NB no mutation).

To test this model one unsuccessful run (D = 8) where the population had converged towards a tree with one + node at level 3 and four at level 4 was run on past generation 500 and the generations where improved solutions where founds was noted. The results are shown in Table 2. In the last but one column of Table 2 the exact count of crossovers which would generate an improved solution is given, the right most column gives the corresponding expected number of generations to find an improved solution. There is reasonable agreement with the actual number of generations, recorded in column 3, whereas the simpler model, columns 4 and 5, consistently overestimates the time required to find the next solution.

# 4 Variety

While the populations converge to the extent that most of the trees in the population are similar to each other, the population does not converge to the extent that all the population becomes the same. Typically (cf. Figure 3) after a 100 generations about two thirds of the population are unique. Recall from Section 3.1 that about two thirds of children are of a different length to their host parent and about one third are the same (assuming most trees are reasonably full by this stage). With a relatively small population it seems reasonable to assume that children which are different from their parent are also likely to be different from every other child in the population. Children produced by crossovers which swap subtrees at the same level are likely to be the same as one parent and so likely to be the same as another child of the same parent or an identical individual in the previous generation.

## 4.1 Variety in the Initial Population

Figure 3 highlights another problem with GP when it is applied to the MAX problem: in contrast with normal GP practice the initial populations contains a large number of duplicates. This is inherent in the ramped half-and-half method when used in a problem with such small function and terminal sets. Recall that with the ramped half-and-half method random trees are created with a maximum depth evenly distributed between 1 (counting the root node as zero) and the maximum depth allowed in the initial population. Half are created as full trees of this maximum depth and half are created using a grow method which creates trees of different shapes. ([Iba, 1996] and [Bohm and Geyer-Schulz, 1996] discuss alternative means of generating random trees for use as the initial population in GP runs).

With D = 8 about $\frac{1}{8}$ of trees will be created with a maximum depth of 1 and therefore

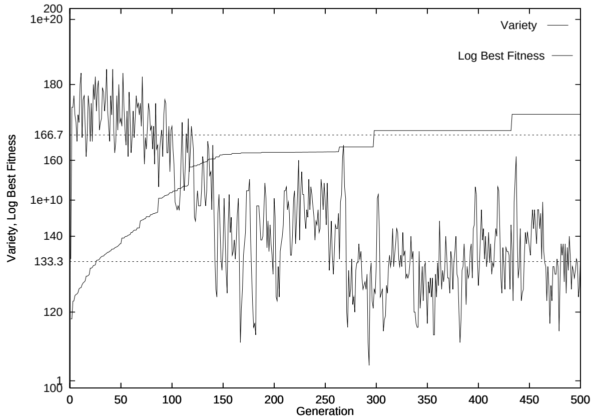


Figure 3: Number of different trees and best fitness in a typical run (D = 8)

will contain a single function. I.e. about 25 trees will consist of a single + node and two 0.5 terminals or a $\times$ node and two 0.5 terminals, and therefore they are very likely to be identical to at least one other member of the population. $\frac{1}{8}$ of trees are created with a maximum depth of 2 and so will contain no more than two or three functions. As there are 16 possible trees with two or three functions it is likely most trees with a depth of two will not be unique. With larger trees the number of possible trees grows rapidly and so most will be unique. $\frac{1}{4}$ of trees generated by the grow method will not grow either branch from the root and so a further $\approx \frac{1}{2}\frac{8-2}{8}\frac{1}{4}$ (i.e. 18.75) trees will be very small and so unlikely to be unique.

In summary we expect 68.75 duplicates in the initial population. This agrees very well with the 66 duplicate individuals found in a typical initial population (D = 8). In problems with larger function or terminal sets the chance of randomly generating duplicates will be much smaller, but a high proportion of small programs can still be expected.

### 4.2 Selection Pressure

In the MAX problem we see a rapid convergence of the population towards solutions that are beneficial in the first few generations. However early solutions may not readily evolve to acceptable solutions. We suspect this is common to many GP problems. In the case of the MAX problem early high fitness individuals may use + functions to form comparatively large values. Later generations can then yield still larger values by joining subtrees composed of + nodes with $\times$ functions. However, as [Gathercole and Ross, 1996] point out, the depth restriction and the mechanics of the crossover operator make the insertion of $\times$ nodes near the root of large trees difficult. This section was motivated by the suggestion that the selection pressure was too high (i.e. the tournament size was too big) and this was responsible for driving the initial generations towards convergence too quickly.

The chance of an individual $i$ being selected by a tournament is given by its rank $r_i$ within the population of size $p$ and the tournament size $t$ by the formula [Blickle and Thiele, 1995; Langdon, 1996]

$$\frac{(r_i/p)^t - (r_{i-1}/p)^t}{r_i - r_{i-1}}$$

In a large and diverse population this can be approximated by

$$t(r_i/p)^{t-1}$$

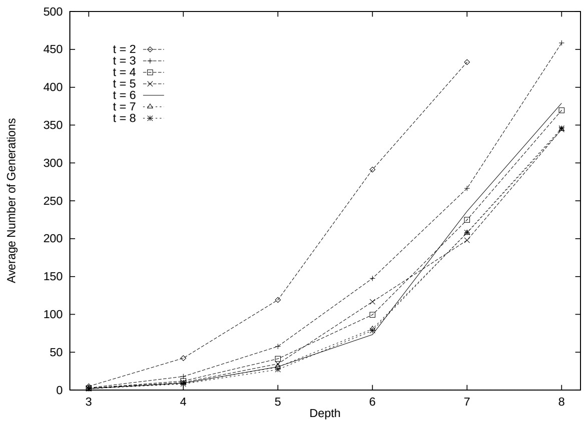


Figure 4: Mean number of generations need by the successful runs with tournament sizes from 2 to 8

The expected number of children produced by individuals of rank R or less is approximately

$$p \sum_{i=1}^{R} t(r_i/p)^{t-1} \approx p(R/p)^t$$

I.e. the worst $\approx \sqrt[t]{1/2}$ of the current generation produce half the children of the next generation, as do the best $1 - \sqrt[t]{1/2}$. With a tournament size of 6 this means on average the best 11% of the population (i.e. 22 individuals) produce half the next generation (i.e. 100 children). Given one third of these are likely to be identical to their parents, the elite members of the population can readily pass identical copies of themselves to the next and succeeding generations.

Figures 4 and 5 give the average number of generations in successful runs and proportion of runs that failed respectively with different tournament sizes. Each point represents the mean of 50 independent runs. For clarity only the data for the runs where the maximum height in the initial population is the same as that in later generations are displayed. We see performance is essentially independent of selection pressure except for tournament sizes of two and three. A possible explanation is with lower selection pressure the fitter members of the population have on average only two or three (i.e. $t$) children, since only one third of these will be identical it becomes impossible for (all but the very best of them) to pass on identical copies of themselves to later generations. (The GP remains elitist and so passes on one copy of the best individual to the next generation without crossover).

# 5 Price's Covariance and Selection Theorem

Price's Covariance and Selection Theorem [Price, 1970] from population genetics relates the change in frequency of a gene 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 produced by individuals in that population (see Equation 1). The theorem holds "for a single gene or for any linear combination of genes at any number of loci, holds for any sort of dominance or epistasis (non-linear interaction between genes), for sexual or asexual reproduction, for random or non-random mating, for diploid, haploid or polyploid species, and even for imaginary species with more than two sexes" [Price, 1970]. In particular it applies to genetic algorithms (GAs) [Altenberg, 1994; Langdon, 1996].

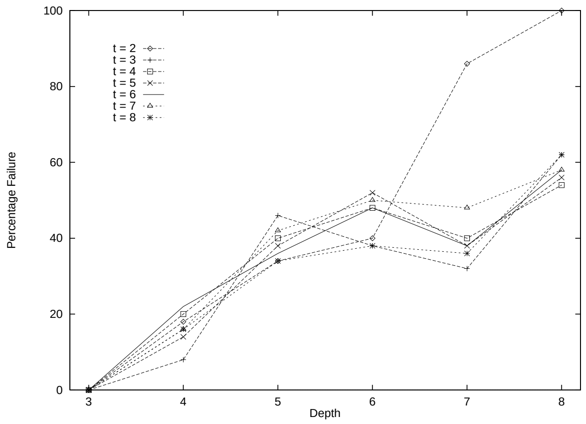


Figure 5: Percentage of runs ending in failure with tournament sizes from 2 to 8

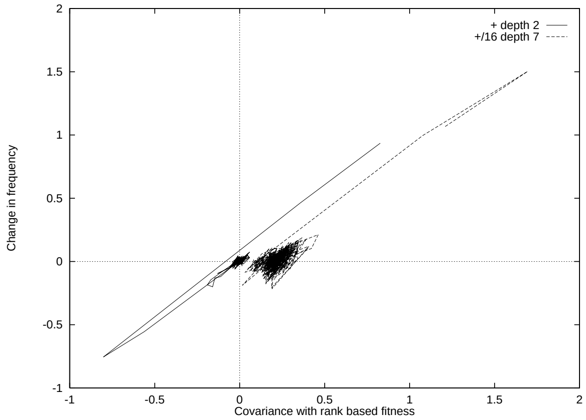


Figure 6: Change in Frequency v. Covariance for + in a typical run (+ at depth 7 rescaled by dividing by 16).

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

$q$ = Frequency of gene
$z$ = Number of offspring
$\overline{z}$ = Mean number of children
Cov = covariance

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 we expect the change in frequency of a gene to be given by $\text{Cov}(q, t(r_i/p)^{t-1})$ as long as crossover is random. Figure 6 shows good agreement between theory and measurement for + nodes near the root but at the maximum allowed depth for + nodes, the crossover depth restriction acts to depress the change in frequency. After three generations the number of + nodes no longer increases dramatical and instead clusters near zero even though the covariance is positive. Similar effects are seen with $\times$ nodes and the opposite with 0.5 nodes where the change in frequency tends to be larger than predicted later in the run when most trees have reached near the maximum size.

Figure 7 considers 3 runs, a typical unsuccessful run (which would find a solution if run for long enough), a successful run and one which can never find the optimal solution. The

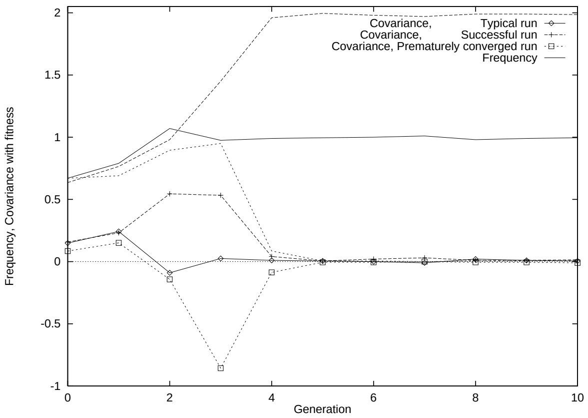


Figure 7: Covariance with fitness and frequency for $\times$ in the second level of the tree in a typical run, a successful run and a prematurely converged run (D = 8).

three are principally separated by the number of $\times$ nodes at the second level of the tree. In the successful run this rises rapidly so by generation four there are on average nearly 2.0 such nodes per individual in the population. This remains true throughout the rest of the run.

In the typical run, the number of $\times$ nodes at level 1 rises in generations 1 and 2 as it does in the successful run but then its covariance with fitness drops to near zero and the its frequency converges towards 1.0 where it remains until the end of the run. The unsuccessful run starts like the other two runs but then in generation three the covariance becomes negative and frequency falls to near zero in the next generation. The population eventually converges to zero i.e. there are no $\times$ nodes at level 1 anywhere in the population and (as explained above) it cannot escape from this trap. NB the eventual outcome of these three runs 500 generations later can be predicted from the covariance of gene frequency with (rank based) fitness in generations two and three.

# 6 Conclusions

We have extended the analysis of the difficulties crossover experiences presented in [Gathercole and Ross, 1996] to include a quantitative model of the later evolution of MAX problem populations. Comparison with experimental results indicate the model gives an indication of the likely rate of improvement but numerical values are not as close as we would like.

While [Gathercole and Ross, 1996] suggest that MAX problem populations converge, measurements indicate after many generations two thirds of the population are unique. A model based on the interaction between crossover and the depth restriction has been presented which correctly predicts this and the variety in the early generations. Additionally a model of the number of duplicate individuals in the initial populations has been presented which highlights a potential problem with the standard technique for generating the initial random population.

In the last section we used the MAX problem to demonstrate the applicability of Price's Covariance and Selection Theorem of gene frequencies to GP populations, but noted GP's depth restriction influences crossover and the consequent implications on the changes in gene frequencies. Gene covariance was analysed to help explain why GP populations get locked into suboptimal solutions in the first few generations.

# A Benchmarks

A number of CPU and memory usage measurements were made using the MAX problem running on a 400 MHz Digital Alphastation 500/400 with a 2MB cache and a SUN SPARCstation 2 fitted with an 80MHz CPU Weitek PowerUP CPU and floating point processor giving a performance of about 1.6 $\times$ that of a standard SPARCstation 2.

## A.1 Memory

In GP-QUICK the memory occupied by the GP process is normally dominated by the space required to store the population. When run as a steady state GA there is a single population but when run as a generational GA the new population is created separately from the current one and so (in this implementation) twice as much memory is required.

GP-QUICK uses fixed length objects to store the population. The memory required for each GP individual is one byte times the maximum allowed program length (up to 511 in the MAX problem) plus a fixed overhead. The overhead is 182 bytes on the SUN and 304 bytes on the Alpha. To estimate the total memory, the number of bytes per GP individual is multiplied by the size of the population (and doubled if using a generational GA). With small populations, the memory occupied by the code can also be important.

## A.2 Speed

A number of trials on both machines where made using the MAX problem. In this problem the population will be dominated by + and 0.5 primitives, as both of these require only trivial floating point calculations we expect the run time to be dominated by the time to interpret the GP programs and so Table 3 gives an indication of the performance of GP-QUICK on the two machines. The top two rows of figures are obtained by dividing the total time by the number of individuals or primitives evaluated.

GP-QUICK supports a FASTEVAL pre-processing step which is aimed as increasing performance when GP individuals are evaluated multiple times. In the MAX problem individuals are only evaluated once so a modest saving might have been made by removing the FASTEVAL step.

An estimate of program evaluation time was made by evaluating individuals multiple times. The bottom two rows of Table 3 were obtained by dividing the increase in total time by the increase in number of individuals and primitives evaluated. The lower two rows can be used to estimate the extra runtime caused by multiple evaluations, e.g. due to multiple fitness cases.

Due to the cache structures on the two machines individuals can be re-evaluated very quickly, leading to very fast times in the lower part of Table 3. On problems with large numbers of (or complex) functions or terminals the cache may not perform as well and longer run times are expected.

# Acknowledgements

This research is supported by the Defence Research Agency in Malvern and the British Council. The first author thanks Andy Singleton for the original version of GP-QUICK on which his code is based.

Table 3: GP-QUICK MAX Problem $\mu$S CPU

| Total time | SPARC | Alpha |
|---|---|---|
| per individual | 680 | 92 |
| per primitive | 6.3 | 0.8 |
| Evaluation only | | |
| individual | 37 | 4.8 |
| primitive | 0.3 | 0.042 |

## References


[Altenberg, 1994] Lee Altenberg. The evolution of evolvability in genetic programming. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 3, pages 47–74. MIT Press, 1994.

[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.

[Blickle and Thiele, 1995] Tobias Blickle and Lothar Thiele. A mathematical analysis of tournament selection. In Larry J. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference*, pages 9–16, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[Bohm and Geyer-Schulz, 1996] Walter Bohm and Andreas Geyer-Schulz. Exact uniform initialization for genetic programming. In Richard K. Belew and Michael Vose, editors, *Foundations of Genetic Algorithms IV*, University of San Diego, CA, USA, 3–5 August 1996.

[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.

[Holland, 1992] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, 1992. First Published by University of Michigan Press 1975.

[Iba, 1996] Hitoshi Iba. Random tree generation of genetic programming. 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 144–153, Berlin, Germany, 22-26 September 1996. Springer Verlag.

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

[O'Reilly and Oppacher, 1995] Una-May O'Reilly and Franz Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In L. Darrell Whitley and Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 73–88, Estes Park, Colorado, USA, 31 July–2 August 1994 1995. Morgan Kaufmann.

[Poli and Langdon, 1997] Riccardo Poli and W. B. Langdon. A new schema theory for genetic programming with one-point crossover and point mutation. Technical Report CSRP-97-3, School of Computer Science, The University of Birmingham, B15 2TT, UK, January 1997. Submitted to GP-97.

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

[Punch *et al.*, 1996] William F. Punch, Douglas Zongker, and Erik D. Goodman. The royal tree problem, a benchmark for single and multiple population genetic programming. In Peter J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 15, pages 299–316. MIT Press, Cambridge, MA, USA, 1996.

[Whigham, 1995] P. A. Whigham. A schema theorem for context-free grammars. In *1995 IEEE Conference on Evolutionary Computation*, volume 1, pages 178–181, Perth, Australia, 29 November - 1 December 1995. IEEE Press.