Fitness Sharing in Genetic Programming

R I (Bob) McKay

School of Computer Science
Australian Defence Force Academy
Canberra, ACT, Australia
rim@cs.adfa.edu.au
+61 2 6268 8169

Abstract

This paper investigates fitness sharing in genetic programming. Implicit fitness sharing is applied to populations of programs. Three treatments are compared: raw fitness, pure fitness sharing, and a gradual change from fitness sharing to raw fitness. The 6- and 11-multiplexer problems are compared. Using the same population sizes, fitness sharing shows a large improvement in the error rate for both problems. Further experiments compare the treatments on learning recursive list membership functions; again, there are dramatic improvements in error rate. Conversely, fitness sharing runs achieve comparable results to raw fitness using populations two to three times smaller. Measures of population diversity suggest that the results are due to preservation of diversity and avoidance of premature convergence by the fitness sharing runs.

1 INTRODUCTION

Genetic programming has been applied to a wide range of problems, including some where performance equal to or better than the best human performance has been demonstrated. Nevertheless, genetic programming, like all evolutionary algorithms, can suffer from the phenomenon of premature convergence, whereby variation is eliminated from a population of fairly fit individuals before a complete solution is achieved.

Premature convergence has been heavily studied in the evolutionary community, and a number of mechanisms for preserving and enhancing variation within populations have been proposed. One class of such mechanisms goes by the name of fitness sharing. Fitness sharing was introduced by Deb and Goldberg (1989). That form, known as explicit fitness sharing, relies on a distance metric to cluster population members. Members which are similar to each other are punished for this similarity by being required to share their fitness, while isolated individuals retain all the fitness value that they achieve.

Some years later, Smith, Forrest and Perlson (1992) introduced implicit fitness sharing for concept learning problems. Implicit fitness sharing differs from the explicit form in that no explicit distance metric is required. Instead, all population members which correctly predict a particular input/output pair share the payoff for that pair.

Implicit fitness sharing extends readily to many genetic programming approaches, but with the increased complexity of genetic programming search spaces there is a risk that the benefits of fitness sharing may be dissipated. This paper investigates fitness sharing for three genetic programming problems.

2 IMPLICIT FITNESS SHARING

Implicit fitness sharing makes the assumption that the overall (raw) fitness of an individual may be determined by summing its performance over a set of sub-problems, each sub-problem having only a small number of possible outcomes (often two). For example, with concept learning problems, the raw fitness of the individual is the sum of the reward for the individual cases, and the reward for an individual case is 1 if the individual correctly predicts that case, otherwise 0. Mathematically, for each program i

$$f_{raw}(i) = \sum_{c \in cases} reward(i(c))$$

The implicitly shared fitness of an individual is calculated instead by dividing the reward for each case by the number of individuals which make the same prediction for that case, before summing over cases:

$$f_{share}(i) = \sum_{c \in cases} \frac{reward(i(c))}{\sum_{i':i'(c)=i(c)} reward(i'(c))}$$

Implicit fitness sharing provides selection pressure for each individual to make different predictions from other individuals, modifying the simple pressure to make accurate predictions which is provided by raw fitness. So evolutionary search with fitness sharing gives a less eager search than with raw fitness; this can result in lower performance early in a run, but is more than outweighed, for these problems, by a delay in convergence of the algorithm, and better overall performance.

3 EXPERIMENTAL SETUP

3.1 EXPERIMENTAL PROBLEMS

Three simple prediction problems were used as test cases: the 6- and 11-multiplexer problems and the recursive list membership problem. The 6-multiplexer problem is to predict the output of a multiplexer having as inputs two address and four data lines; the 11 multiplexer problem extends this to three adress and eight data lines. The recursive list membership task is to learn a lisp-like expression for list membership.

Table 1: 6-Multiplexer Grammar

EXPR → BOOL
BOOL → TERM
BOOL → and BOOL BOOL
BOOL → or BOOL BOOL
BOOL → not BOOL
BOOL → if BOOL BOOL BOOL
TERM → a0
TERM → a1
TERM → d0
TERM → d1
TERM → d2
TERM → d3

Table 2: List Membership Grammar

S → M
M → if EXPN EXPN M
M → ''
EXPN → atom LST
EXPN → eq LST LST
EXPN → member x LST
EXPN → true
EXPN → false
LST → first LST
LST → rest LST
LST → x
LST → y

The software is based on Ross' (1999) DCTG-GP system, modified by the incorporation of implicit fitness sharing. DCTG-GP is a grammar-guided genetic programming system (Whigham, 1995), but the grammars used in these experiments simple encode typing, so the results extend to many forms of tree-based genetic programming. All three problems are taken from (Whigham, 1996).

The grammar used for (table 1) the 6- and 11-multiplexers are identical except for the addition of an extra address line terminal 'a2', and four extra data line terminals, 'd4',..,'d7' in the latter.

The grammar used for the list membership problem (table 2) permits the system to learn a recursive definition. The recursive call to member allows the possibility of infinite loops. To handle this, a count of the depth of looping was kept, and a depth greater than 20 caused the function to return the value 'loop', which was treated in fitness evaluation as an incorrect answer.

3.2 GENETIC PROGRAMMING PARAMETERS

The experiments used half-ramped initialisation, and tournament selection. Parameter settings are in Table 3.

Table 3: Multiplexer GP Parameters

PARAMETER	SPECIFICATION
Number of Runs	100
Generations per Run	100 / 400
Population Size	500
Max depth (initial pop)	8
Max depth (subsequent)	10
Tournament size	5
Crossover Probability	0.9
Mutation Probability	0.1

The raw fitness for the 6-multiplexer experiment was the proportion of the 64 cases correctly predicted. For the 11-multiplexer experiments, computation time prevented evaluation of all 2048 cases, so a random subset of 64 distinct cases, independently selected each generation (but the same for all individuals in each generation) was used.

The 6 multiplexer runs were terminated at 100 generations, or earlier if a complete solution was found. The 11 multiplexer runs were terminated at 400 generations or on finding a complete solution (this was taken to be a correct solution for the 64 test cases for that generation - the solution was not checked against the remaining 1984 cases. While some incorrect solutions may have been accepted, this does not affect the comparative evaluations which are the focus of this work).

Table 4: List Membership GP Parameters

PARAMETER	SPECIFICATION
Number of Runs	100
Generations per Run	200
Population Size	1000
Max depth (initial pop)	8
Max depth (subsequent)	10
Tournament size	5
Crossover Probability	0.9
Mutation Probability	0.1

For the list membership problem, a fixed set of membership examples was used: ten positive and ten negative cases, as shown in table 5:

Table 5: List Membership Cases

POSITIVE CASES	NEGATIVE CASES
member(1 [1])	member(1 [6])
member(1 [2 1])	member(1 [3 6])
member(1 [2 3 1])	member(1 [2 3 6])
member(1 [2 3 4 1])	member(1 [2 3 4 6])
member(1 [2 3 4 5 1])	member(1 [2 3 4 5 6])
member(1 [2 3 4 5 6 1])	member(1 [2 3 4 5 6 7])
member(1 [2 3 4 5 6 7 1])	member(1 [2 3 4 5 6 7 8])
member(1 [2 3 4 5 6 7 8 1])	member(1 [2 3 4 5 6 7 8 9])
member(1 [2 3 4 5 6 7 8 9 1])	member(1 [2 3 4 5 6 7 8 9 2])
member(1 [2 3 4 5 6 7 8 9 2 1])	member(1 [2 3 4 5 6 7 8 9 2 3])

Each experiment involved three separate treatments:

- raw fitness
- implicit fitness sharing
- fitness sharing for the first 25% generations, raw fitness for the last 25%, with a linear ramp between the two - raw and shared fitnesses normalised each generation to have equal range before apportioning

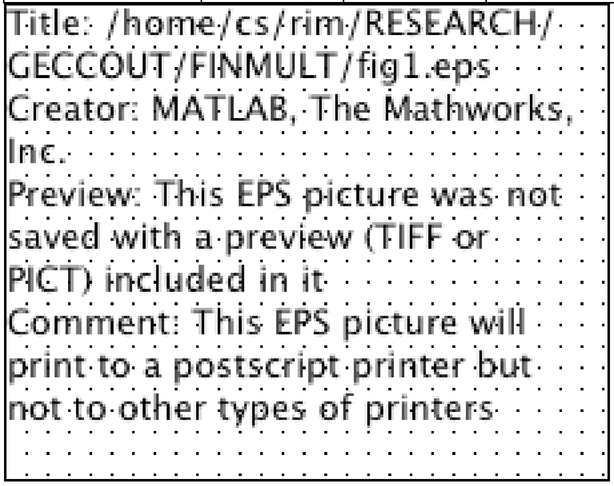
4 RESULTS

4.1 6-MULTIPLEXER RESULTS

As shown in table 3, treatments using fitness sharing found a complete solution reliably, but the raw fitness treatment failed to find a solution in 16% of runs.

Table 6: 6-Multiplexer Results

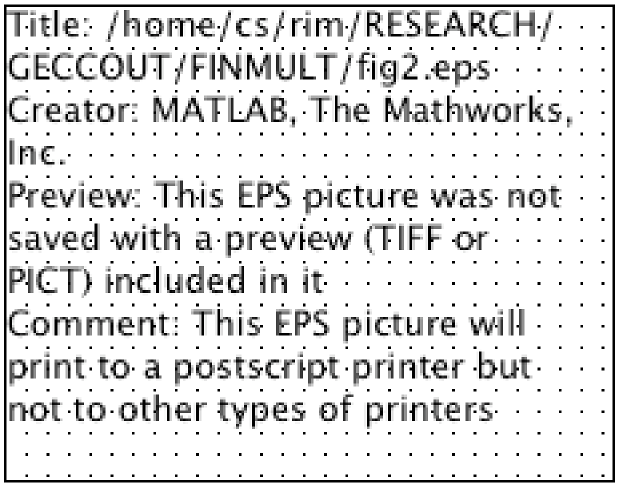
TREATMENT	Raw Fitness	Ramped Fitness	Shared Fitness
% SOLUTIONS	84	100	100



Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig1.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 1: 6-Multiplexer, Solutions Found

Figure 1 shows the percentage of complete solutions for the 6 multiplexer, by generation. The two fitness sharing treatments clearly outperform the raw fitness treatment at all generations after the first ten.



Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig2.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 2: 6-Multiplexer, Error of Best Individual

Figure 2 shows the error rate of the best individual, averaged over the 100 runs of each treatment Again, the two fitness sharing treatments yield consistently better results than that using raw fitness (the first ten generations are omitted to permit reasonable clarity in the graph).

The aim of fitness sharing is to preserve diversity in the population. If the population is highly diverse, then all cases in the dataset will be equally likely to be covered (indeed this is the principle behind implicit fitness sharing), so that a low variance in cover corresponds to a diverse population. Conversely, if the population has low diversity, some cases will be better covered than others, and the variance in cover will be high. Figure 3 shows the variance in cover by generation, and clearly illustrates the ability of fitness sharing to maintain population diversity (the variance falls to zero when all runs have converged).

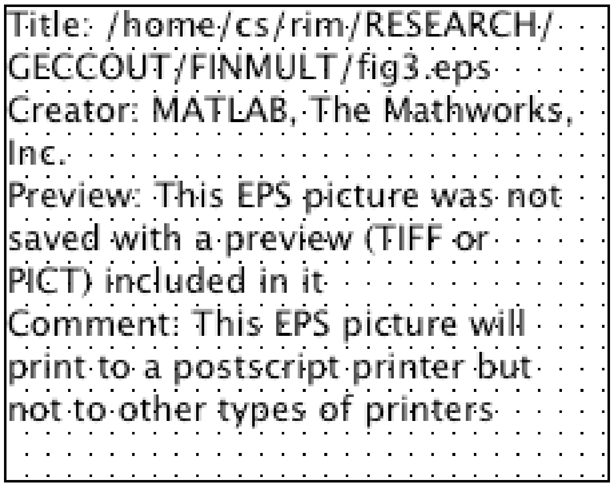

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig3.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 3: 6-Multiplexer, Cover Variance

4.2 11-MULTIPLEXER RESULTS

Table 7: 11-Multiplexer Results

TREATMENT	Raw Fitness	Ramped Fitness	Shared Fitness
% SOLUTIONS	50	99	99

With the 11 multiplexer, there is again a very large difference between the performance of treatments using fitness sharing, and that using raw fitness. 99 of 100 runs using fitness sharing, either throughout or ramped, found a solution, but only 50 of 100 runs using raw fitness did.

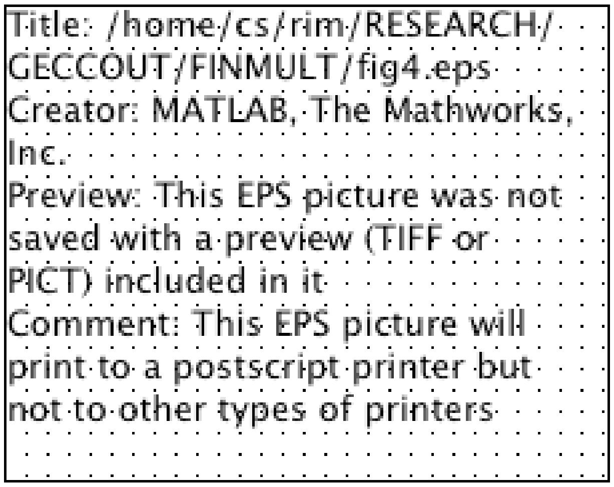

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig4.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 4: 11-Multiplexer, Solutions Found

The fitness sharing approaches clearly outperform the raw fitness treatment for all generations after generation 50. There is a hint that the increase in eagerness of ramped sharing after the ramping cuts in at generation 50 may give slightly better performance, but at the cost of two extra parameters (the start and end point of the ramping).

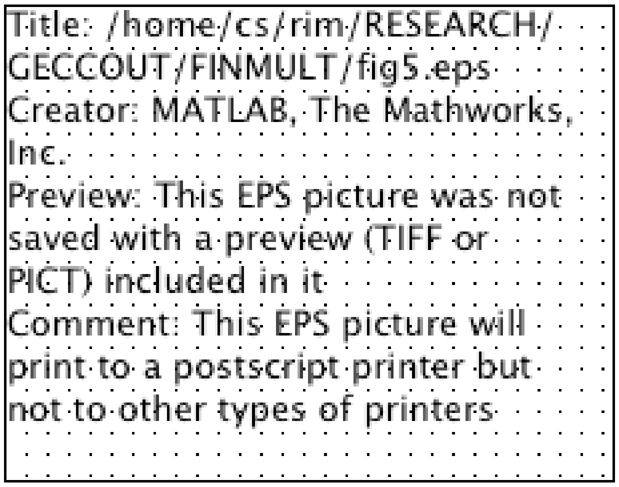

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig5.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 5: 11-Multiplexer, Error of Best Individual

Figure 5 shows the error rate of the best individual, averaged over the 100 runs of each treatment (the first ten generations are omitted to permit reasonable clarity).

Figure 6 shows again the ability of fitness sharing to maintain population diversity. In this case, we can also see the decrease in diversity (increase in variance) of the ramped runs as raw fitness takes over. There is a corresponding small improvement in fitness of the ramped treatment compared with the pure fitness sharing treatment (figure 5). This presumably arises from the increase in eagerness - late in a run, eagerness may be beneficial.

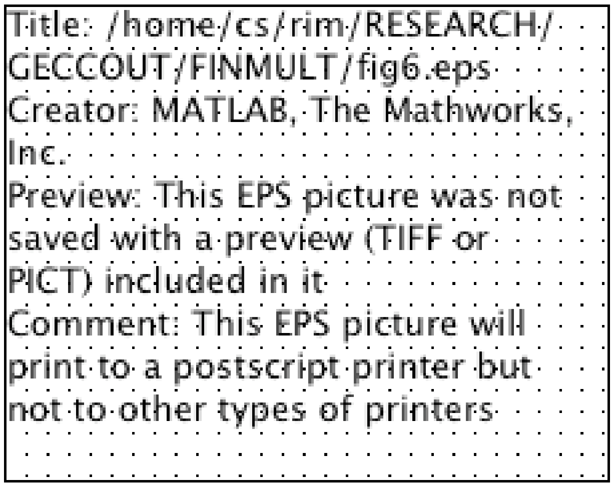

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig6.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 6: 11-Multiplexer, Cover Variance

4.3 LIST MEMBERSHIP RESULTS

The list membership problem yielded similar results: the two fitness sharing treatments clearly outperform the raw fitness treatment at all generations after the first ten.

Table 8: List Membership Results

TREATMENT	Raw Fitness	Ramped Fitness	Shared Fitness
% SOLUTIONS	63	79	79

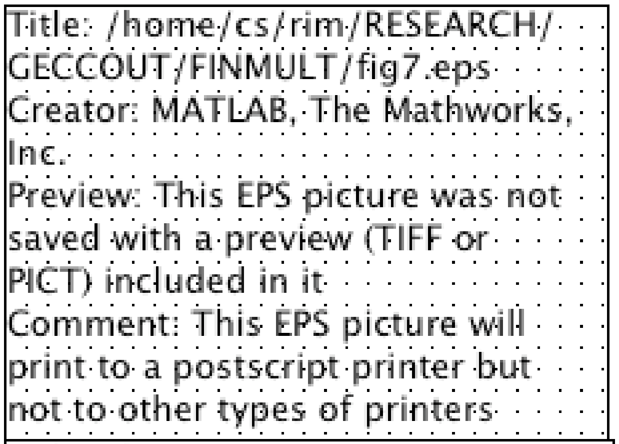

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig7.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 7: List Membership, Solutions Found

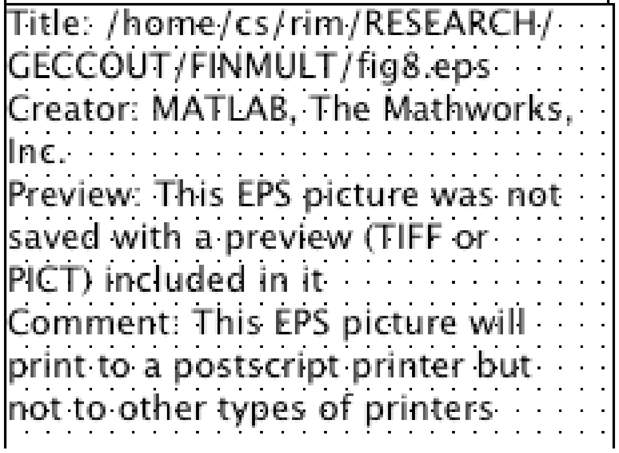

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig8.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 8: List Membership, Error of Best Individual

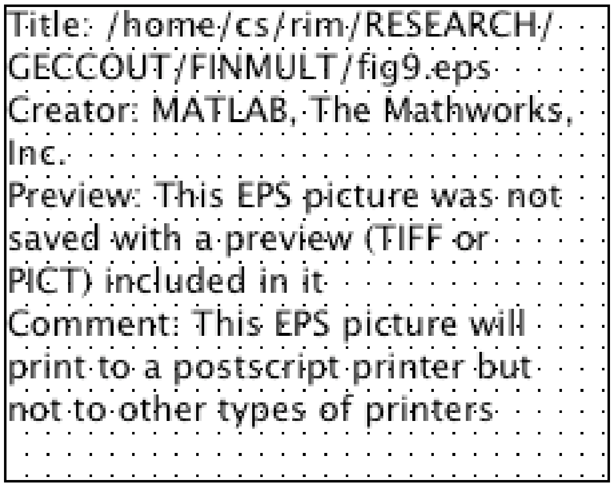

Title: /home/cs/rim/RESEARCH/GECCOUT/FINMULT/fig9.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 9: List Membership, Cover Variance

The error rate of the best individual, averaged over the 200 runs of each treatment, shows similar results (figure 8), though the oscillatory behaviour of the fitness is interesting. On detailed examination, the populations in many runs contain a group of individuals with low error rate (and hence high raw fitness), and another group of very different individuals with higher error rate, but covering a different subset of the test cases. Small fluctuations in the size of the second set are reflected in larger changes in the shared fitness of its members relative to the first set, and hence to the selection pressure toward it, resulting in the oscillations seen.

This oscillatory behaviour can also be clearly seen in the cover variance graph (figure 9)

5 FURTHER EXPERIMENTS: POPULATION SIZE

The results indicate that fitness sharing leads to improved predictive accuracy when other evolutionary parameters are kept the same. But computational requirements are an important issue in genetic programming. If fitness sharing allows us to obtain greater accuracy from similar sized runs, then presumably it will also allow us to obtain equivalent accuracy from smaller sized runs. Experiments were conducted with pure fitness sharing on different sized populations to determine what population size yielded similar predictive accuracy to raw fitness with the populations in the preceding runs.

Table 9: GP Parameters for Multiplexer Population Size Experiments

PARAMETER	SPECIFICATION
Number of Runs	100/50
Generations per Run	100 / 400
Population Size	0 - 500 (steps of 50/100)
Max depth (initial pop)	8
Max depth (subsequent)	10
Tournament size	5
Crossover Probability	0.9
Mutation Probability	0.1

For the raw fitness treatment, the population size was held, as before, at 500. The fitness sharing treatments reduced the population size in decrements of 50. Experiments were run for both the 6- and 11-multiplexer problems. As in the previous series of experiments, the 6-multiplexer experiments were run for 100 generations, since the runs appear to be at or near convergence by then. The 11-multiplexer experiments were run to 400 generations; even this does not show full convergence in all cases, but computational limitations have precluded longer runs, and also limited the population step sizes to 100 (50 for the 6-multiplexer) and the number of runs to 50 (100 for the 6-multiplexer).

Table 10: GP Parameters for List Membership Population Size Experiments

PARAMETER	SPECIFICATION
Number of Runs	100
Generations per Run	200
Population Size	0 - 1000 (steps of 100)
Max depth (initial pop)	8
Max depth (subsequent)	10
Tournament size	5
Crossover Probability	0.9
Mutation Probability	0.1

For the list membership problem, in the raw fitness treatment, the population size was held at 1000. The fitness sharing treatments reduced the population size in decrements of 100. As in the previous series of experiments, the experiments ran for 200 generations.

6 FURTHER RESULTS

6.1 6-MULTIPLEXER RESULTS

Table 11: 6-Multiplexer Size Comparisons

Raw	Shared									
500	500	450	400	350	300	250	200	150	100	50
84	100	100	98	99	99	97	91	89	74	52

The second row in the table shows the population size, the third the percentage of complete solutions over 100 runs.

Figure 10 shows the error rate of the best individual, averaged over the 100 runs of each treatment (the first ten generations are omitted to permit reasonable clarity in the graph). The results show that there is a trade-off between the number of generations in the run and the population size. Fitness sharing with a population of 150 outperforms raw fitness with a population of 500 after about generation 50, but to obtain earlier high performance a larger population is required (and as previously discussed, the very early performance of raw fitness is marginally better than for fitness sharing, even with similar populations).

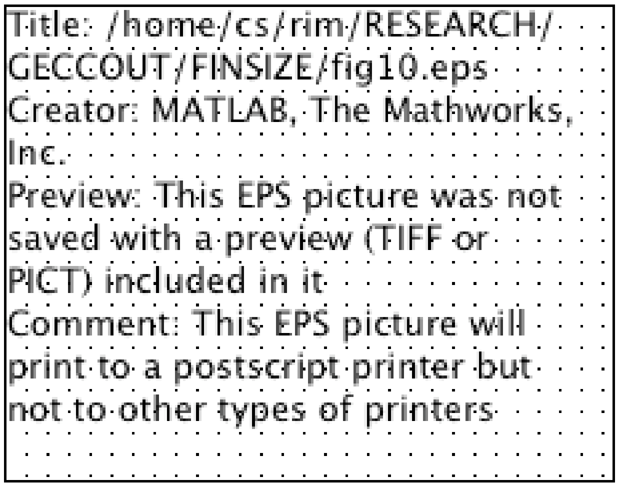

Title: /home/cs/rim/RESEARCH/GECCOUT/FINSIZE/fig10.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 10: 6-Multiplexer Size Comparisons

Table 12: 11-Multiplexer Size Comparisons

Raw	Shared				
500	500	400	300	200	100
60	100	100	98	90	26

The second row in the table shows the population size, the third the percentage of complete solutions over 50 runs. Stochastic noise has given slightly different values in columns 1 & 2 from those in table 7.

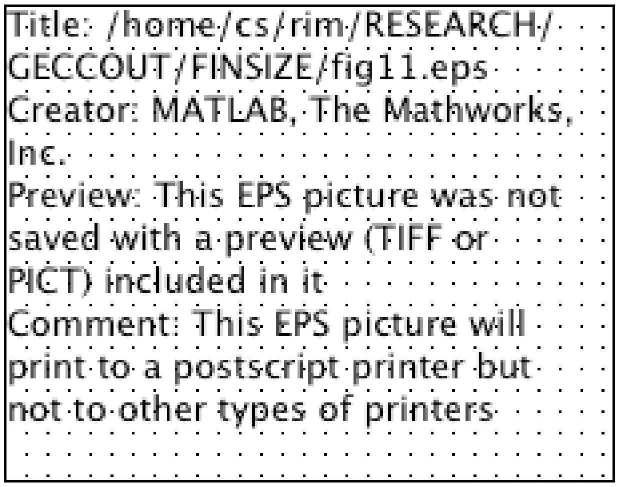

Title: /home/cs/rim/RESEARCH/GECCOUT/FINSIZE/fig11.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 11: 11-Multiplexer Size Comparisons

Figure 11 shows the error rate of the best individual, averaged over the 400 runs of each treatment (the first ten generations are omitted to permit reasonable clarity in the graph). There is a trade-off between number of generations in the run and population size. Fitness sharing with a population of 200 outperforms raw fitness with a population of 500 after about generation 150.

Table 13: List Membership Size Comparisons

Raw	**Shared**									
1000	1000	900	800	700	600	500	400	300	200	100
63	79	77	79	61	65	68	66	46	39	19

The second row in the table shows the population size, the third the percentage of complete solutions over 50 runs.

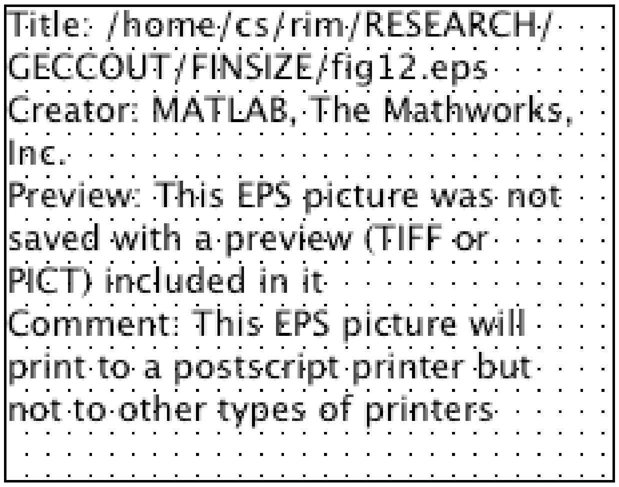

Title: /home/cs/rim/RESEARCH/GECCOUT/FINSIZE/fig12.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 12: List Membership Size Comparisons

Figure 12 shows the error rate of the best individual, averaged over the 200 runs of each treatment. Because of the oscillatory behaviour previously noted, the graph is more difficult to interpret, but it is clear that equivalent performance to the raw fitness runs is achieved by fitness shared runs with roughly half the population size.

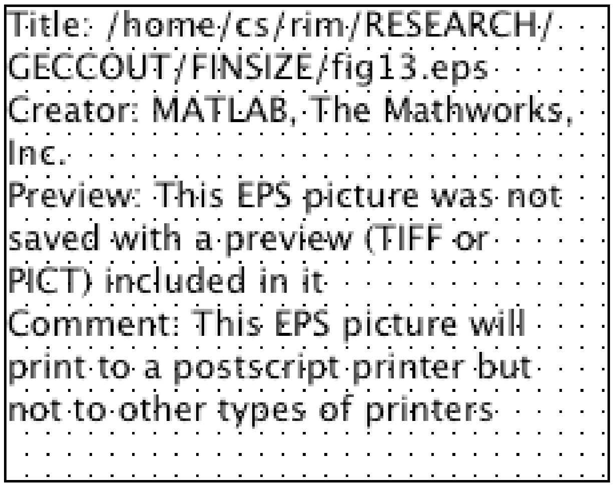

Title: /home/cs/rim/RESEARCH/GECCOUT/FINSIZE/fig13.eps
Creator: MATLAB, The Mathworks, Inc.
Preview: This EPS picture was not saved with a preview (TIFF or PICT) included in it
Comment: This EPS picture will print to a postscript printer but not to other types of printers

Figure 13: List Membership Size Comparisons

The 'runs incomplete' plot (figure 13) shows similar behaviour. There is a strong suggestion that the smaller population fitness shared runs have not yet converged after 200 generations, so that extended runs (which were not completed due to computational cost) might have reflected a stronger trade-off in favour of fitness sharing.

7 DISCUSSION

7.1 RELATED WORK

A wide variety of other diversity-promoting mechanisms have been studied in the evolutionary community, among them island models (Tanese, 1989), mating restrictions (Goldberg 1989), and De Jong's crowding (reported in Goldberg 1989). Of these, only island models appear to have widespread acceptance in genetic programming (Andre & Koza, 1995), with the emphasis being on their advantages for parallelism rather than for diversity preservation.

By comparison with island models, fitness sharing approaches to diversity directly promote variation within the population, rather than simply permitting sub-optimal populations to survive through isolation, hence they potentially provide greater opportunity for delaying convergence. On the other hand, implicit fitness sharing requires the computation of population-wide statistics in each generation, and hence is unattractive for parallel implementation. However fitness sharing and island models are not incompatible - fitness sharing may be attractive as an additional diversity-promoting mechanism within the individual demes of an island model.

7.2 FURTHER WORK

The experiments reported demonstrate that fitness sharing has a very significant impact on the performance of genetic programming on two classes of problems. Experiments are currently under way on real-World problems to confirm whether the effect extends beyond toy problems

As mentioned in earlier discussion, there is usually a short-term payoff in switching from fitness sharing to raw fitness, due to the increased eagerness of search. Thus mixed approaches, using fitness sharing early in a run and raw fitness toward the end, offer potential benefits. The ramped approach used here shows one simple algorithm to exploit this, but many other approaches are possible. Investigations of a number of alternative approaches are currently under way. Further experiments are also in progress on combining fitness sharing with populations of partial functions, and with committee methods of population evaluation.

The aim of fitness sharing is to maintain population diversity and thus delay convergence. But convergence is strongly associated with the phenomenon of bloat (Nordin et al, 1995). Thus one would expect delayed convergence to reduce bloat. Results from the above experiments do not support this expectation - the tree depth profiles of the raw fitness and fitness sharing runs are almost identical. Interestingly, preliminary experiments combining partial functions and fitness sharing do show a modest reduction in bloat.

8 CONCLUSIONS

Implicit fitness sharing applied to genetic programming for the 6- and 11-multiplexer problems and the recursive list membership problem results in dramatic improvements in the error rate of learned functions. This reduction appears to be due to the ability of fitness sharing to maintain population diversity.

Conversely, fitness sharing permits significantly smaller populations to achieve similar learning accuracy to larger populations using raw fitness. In the experiments reported here, reductions in population size of a factor of three were obtained, with a comparable reduction in computational cost - the incremental cost of computing the implicitly shared fitness was negligible.

Acknowledgments

The ideas in this paper have benefited greatly from discussions over the years with Paul Darwen, Peter Whigham, Xin Yao and Ko-Hsin Liang. The system has been implemented through modifications to Brian Ross' innovative DCTG-GP system, which contributed greatly to the rapidity with which a wide range of different algorithms could be prototyped in the development of this system. My thanks also to the reviewers of this paper for their contributions to the clarity of the exposition.

References

Andre, D and Koza, J R 'Parallel Genetic Programming on a Network of Transputers' in J Rosca (ed) *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Pp 111-120, Morgan Kaufmann, 1995

Deb, K and Goldberg, D E: 'An investigation of niche and species formation in genetic function optimization' in J D Schaffer (Ed) *Proceedings of the Third International Conference on Genetic Algorithms*, Pp 42-50, Morgan Kaufmann, 1989

Goldberg, D E *'Genetic Algorithms in Search, Optimisation, and Machine Learning'*, Addison-Wesley, 1989

Montana, D J: 'Strongly Typed Genetic Programming', Technical Report BBN 7866, Bolt Beranek and Newman, Inc, Cambridge MA, 1994

Nordin, P, Francone, F and Banzhaf, W: 'Explicitly defined introns and destructive crossover in genetic programming' in J Rosca (ed) *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Pp 6-22, Morgan Kaufmann, 1995

Ross, B J: 'Logic-based Genetic Programming with Definite Clause Translation Grammars', Technical Report #CS-99-02, Dept of Computer Science, Brock University, St Catharines Ontario, 1999

Smith, R E, Forrest, S and Perelson, A S: 'Searching for diverse, cooperative populations with genetic algorithms', *Evolutionary Computation* 1(2), Pp 127-149, 1992

Tanese, R: 'Distributed Genetic Algorithms' in D Schaffer (ed) *Proceedings of the Third International Conference on Genetic Algorithms*, Pp 434-439, Morgan Kaufmann, 1989

Whigham, P A: 'Grammatically-biased Genetic Programming' in J Rosca (ed) *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, Pp 33-41, Morgan Kaufmann, 1995

Whigham, P A: 'Grammatical Bias for Evolutionary Learning' PhD Thesis, University College, Australian Defence Force Academy, University of New South Wales, Canberra, 1996