Better Trained Ants for Genetic Programming

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

Technical Report: CSRP-98-12
2 April 1998

Abstract

The problem of programming an artificial ant to follow the Santa Fe trail has been repeatedly used as a benchmark problem in GP. Recently we have shown performance of several techniques is not much better than the best performance obtainable using uniform random search. We suggested that this could be because the program fitness landscape is difficult for hill climbers and the problem is also difficult for Genetic Algorithms as it contains multiple levels of deception.

Here we redefine the problem so the ant is (1) obliged to traverse the trail in approximately the correct order, (2) to find food quickly. We also investigate (3) including the ant's speed in the fitness function, either as a linear addition or as a second objective in a multi-objective fitness function, and (4) GP one point crossover.

A simple genetic programming system, with no size or depth restriction, is shown to perform approximately three times better with the improved training function.

1 Introduction

The problem of programming an artificial ant to follow the Santa Fe trail has been repeatedly used as a benchmark problem [Jefferson *et al.*, 1992, (John Muir trail)], [Koza, 1992; Lee and Wong, 1995; Chellapilla, 1997; Harries and Smith, 1997; Luke and Spector, 1997; Ito *et al.*, 1998; Kuscu, 1998]. Recently we have shown performance of several techniques is not much better than the best performance obtainable using uniform random search [Langdon and Poli, 1998]. We suggested that this was because the program fitness landscape is difficult for hill climbers and the problem contains multiple levels of deception which also makes it difficult for Genetic Algorithms.

Analysis of high scoring non-optimal programs suggests many reach high rewards even though they exhibit poor trail following behaviour. Typically they achieve high scores by following the trail for a while and then losing it at a corner or gap. They then execute a random search until they stumble into the trail at some later point and recommence following it (see Fig. 1). The random search may give them a better score than a competing program which successfully navigated the same bend or gap but lost the trail later (see Fig. 2). In Figs. 1 and 2 the ant's path is shown by the bending line with arrows, uneaten food pellets by black squares, eaten food by squares with horizontal crosses, shaded squares show gaps in the trail.

Here we redefine the problem. The same trail is used but we only place food onto the grid as the ant following along the trail nears it. This makes it difficult for an ant which moves away from the trail to find another part of it by random search. This changes the fitness landscape. We anticipate that almost all optimal points within it will retain their scores and many previously high scoring non-optimal points will be given reduced fitness by the new training regime. (Of the 3916 solutions to the Santa Fe trail problem found by exhaustive search, uniform random and ramped half-and-half random search [Langdon and Poli, 1998], all but at most 2 completed the trail. These solutions are available via anonymous ftp from `ftp.cs.bham.ac.uk` directory `pub/authors/W.B.Langdon/gp-code` file `antsol.tar.gz`). We also

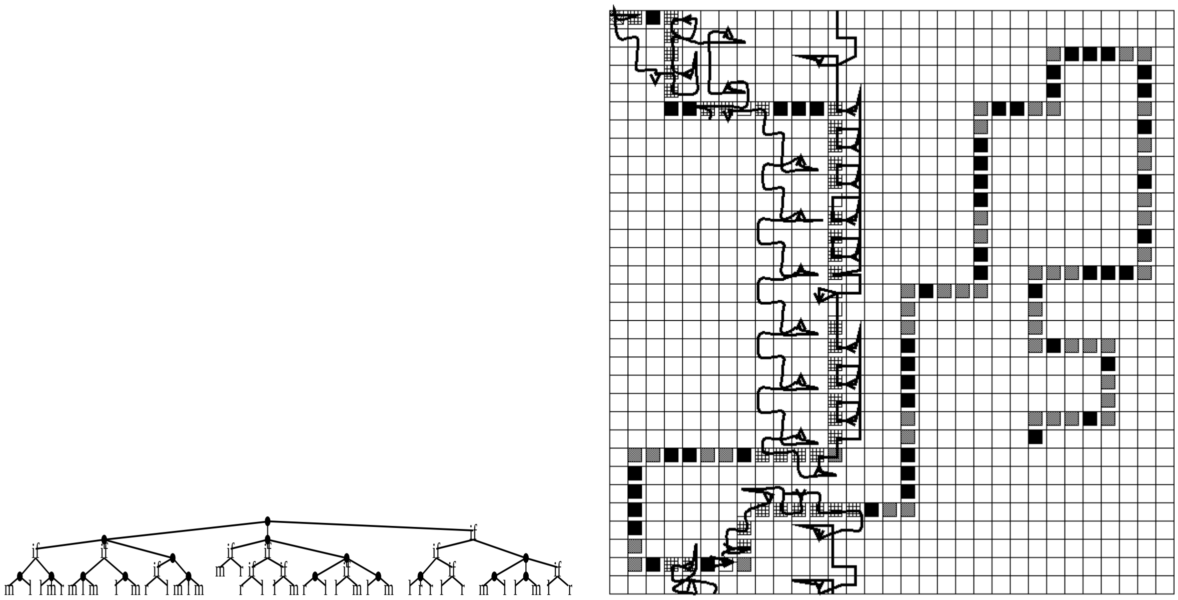

Figure 1: Example program from 1^{st} generation, eats 40 pellets

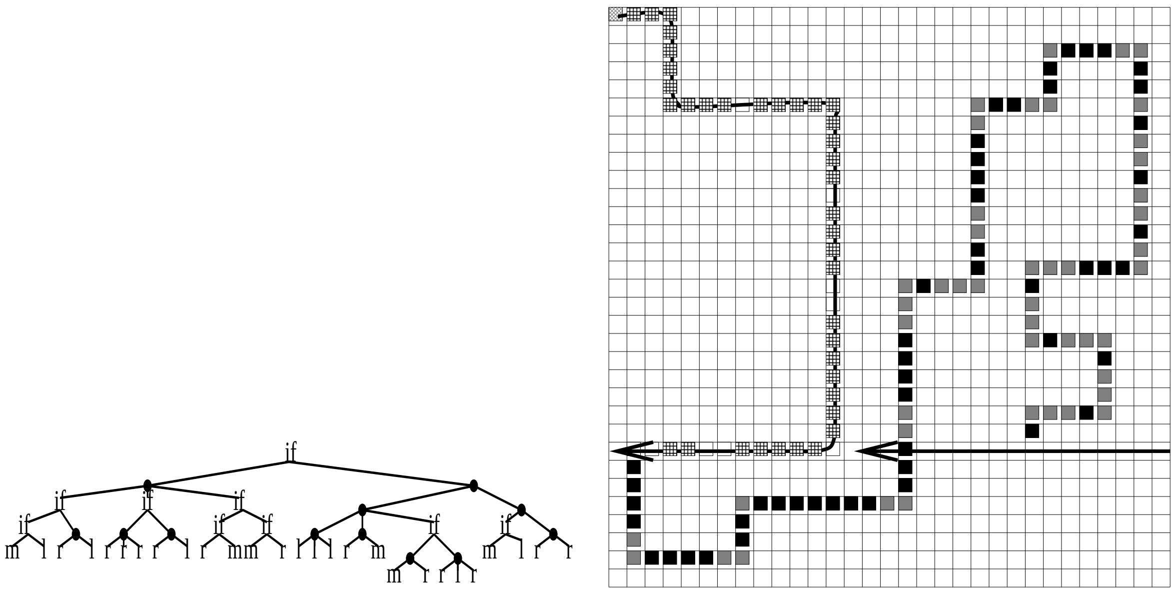

Figure 2: Example program from 1^{st} generation, eats 38 pellets

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
Initial population:	Created using "ramped half-and-half" with a max depth of 6 (duplicates allowed)
Parameters:	90% crossover, 10% reproduction, no size limit
Termination:	Maximum number of generations G = 50

extend interim results presented in [Langdon, 1998a] by requiring the ant to find food quickly, by including its speed in the fitness function and by solving the problem using one point crossover. The changes to the fitness function are intended to make the landscape less deceptive and so easier for genetic algorithms. Removal of false peaks may also benefit hill climbing techniques.

The ant problem and the GP system we use to evolve solutions to it are briefly described in Sect. 2, our results are given and discussed in Sect. 3, and in Sect. 4 we give our conclusions.

2 The Artificial Ant Problem

The genetic programming system and the artificial ant problem [Koza, 1992, pages 147–155] were set up identically to [Langdon and Poli, 1998] except the new training technique was used (see Table 1).

The program primitives are the same as [Koza, 1992, pages 147–155] except now food items on the trail are numbered in the order we expect the ant to eat them. The shortest distance along the trail to each food item is also calculated. Only the first x are placed on the grid initially. As the ant moves and eats them new food pellets are added to the grid. When it eats food pellet n then we ensure all the uneaten food pellets up to pellet $n + x$ are on the grid. At least 50 independent runs where carried out in each experiment.

3 Results

3.1 Limiting Food Ahead

As expected limiting the amount of uneaten food in front of the ant along the trail made the task of evolving suitable programs easier for GP. This can be seen in Fig. 3 which plots the estimated minimum number of individuals that GP with a population size of 500 needs to create in order to have a probability of at least 99% of finding at least one solution. (This is known as "effort" required, cf. [Koza, 1992, page 194], see also Table 2). We estimate of the variability of the effort data by calculating the minimum effort assuming the proportion of successful runs at each generation is increased or decreased by one standard deviation. These are given as the (max) and (min) values in Table 2 and by error bars in Figs. 3 and 4.

Figure 4 plots effort with various restrictions on the maximum program size. Data points are calculated from at least 50 runs using the original Santa Fe trail, only placing five food pellets ahead of the ant (i.e. $x = 5$) and allowing the energy available to the ant to increase as it moves along the Santa Fe trail (see next section). No run found any of the very smallest solutions (of length 11).

Considering first the original problem (crosses and long dashed line in Fig. 4). The minimium estimate for the effort is 189,000 which occurs with a size limit of 100. Values for 25–100 are similar at about two thirds of the effort required when no size limit is imposed. Also effort values for size limits of 200–500

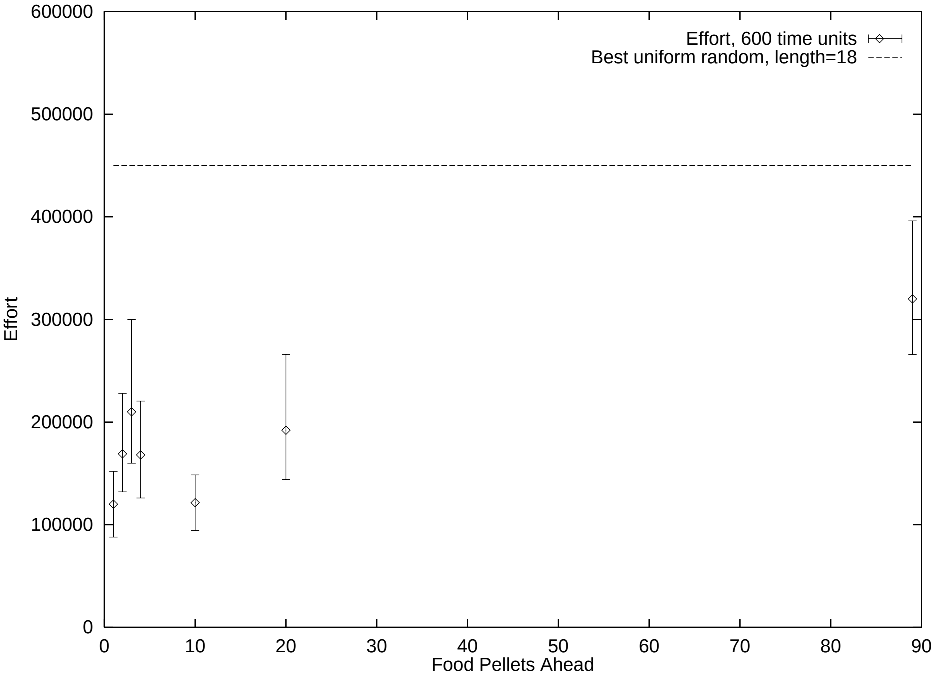

Figure 3: Effort v. no. food pellets ahead the ant can see and eat. Error bars indicate estimates given by one standard deviation above and below minimum measured figure. Original Santa Fe trail always allows all 89 food pellets to be seen.

appear to be much the same as when there is no size limit. The error bars are large as most data points are based on about 10 runs (out of 50) which quickly found a solution.

Now consider the case where food is only placed on the grid as the ant approaches it in along the trail (i.e. $x = 5$, dimonds and solid line in Fig. 4). The minimium estimate for the effort is 104,000 and occurs with a size limit of 50. Values for 25–100 are similar at about 80% of the effort required when no size limit is imposed. Again effort values for size limits of 200–500 appear to be much the same as with no size limit. The effort bars are smaller as data are estimated from more succesful runs. Fig. 4 also gives data from [Chellapilla, 1997].

3.2 Limiting Initial Energy

In this section we introduce a new restriction on the ant programs, which we apply in addition to limiting the amount of food on the trail initially. However no significant increase in performance was found.

Instead of repeatedly executing each program until the ant has made 600 moves or turns we limit the energy initially available to the ant but reward its progress along the trail by increasing the energy available to it. As the trail is 144 squares long and has to be traversed within 600 time steps, programs have to be fairly efficient, consuming on average less than $600/144 \approx 4.2$ energy units per square of trail traversed. Figure 5 shows the energy consumption of the evolved solution given in [Koza, 1992, page 154] and the example programs from Figs. 1 and 2. It appears that the energy limit (solid horizontal line in Fig. 5) provides less incentive to be efficient to poor programs that only manage to follow the trail a short distance than it does to better programs. This suggests it provides less guidance at the begining of GP runs than later. However GP populations often appear to become more resistant to change (possibly associated with programs becoming longer). This in turn suggests the requirement for programs to be efficient should be make itself felt at the start of each GP run. We hoped that by penalising programs that consume much more than 4.2 energy units per trail square we would better guide the population. We implemented this by limiting the each ant's initial energy to 25, which we then increased each time the ant eats a food pellet, up to the original limit of 600. Each program is repeatedly executed until

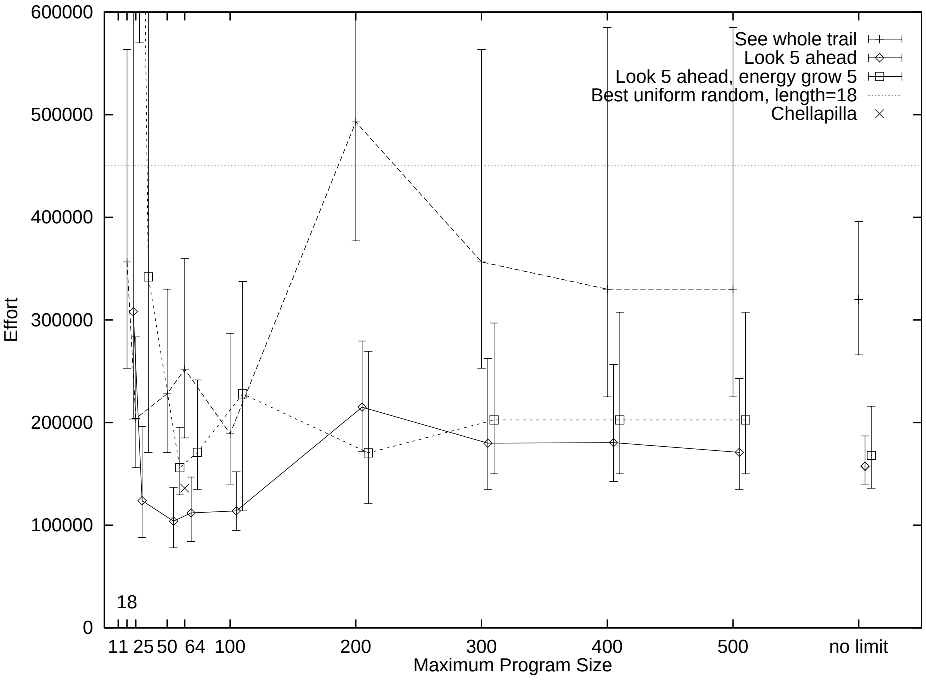

Figure 4: Effort v. maximium program size for original Santa Fe trail and ant restricted to look ahead of 5. Points offset slightly to improve presentation.

Table 2: Effort (in 1,000s) to solve Santa Fe Trail (no program size restriction)

Training Technique	Effort	(min)	(max)	ok/runs
Best uniform random, length=18	455	414	505	101/–
Original	320	266	396	35/200
Strict Hill Climbing 50%–150% mutation [Langdon and Poli, 1998]	188	118	433	8/50
[Chellapilla, 1997]	136			47/59
scalar speed	196	130	382	9/50
1 food ahead	120	88	152	19/50
1 food ahead, ramped energy	555	390	900	7/50
1 food ahead, ramped energy, scalar speed	230	170	349	13/50
5 food ahead	158	140	187	71/200
5 food ahead, ramped energy	168	136	216	32/100
5 food ahead, ramped energy, scalar speed	264	202	372	12/50
5 food ahead, ramped energy, Pareto, speed threshold 1.0, pop sample size 25	221	169	312	13/50
5 food ahead, ramped energy, Pareto, speed threshold 2.0, pop sample size 200	192	144	252	18/50
1ptXO, original fitness function	405	279	725	8/100
1ptXO, 5 food ahead,	204	143	315	11/50
1ptXO, 5 food ahead, ramped energy	279	198	420	11/50

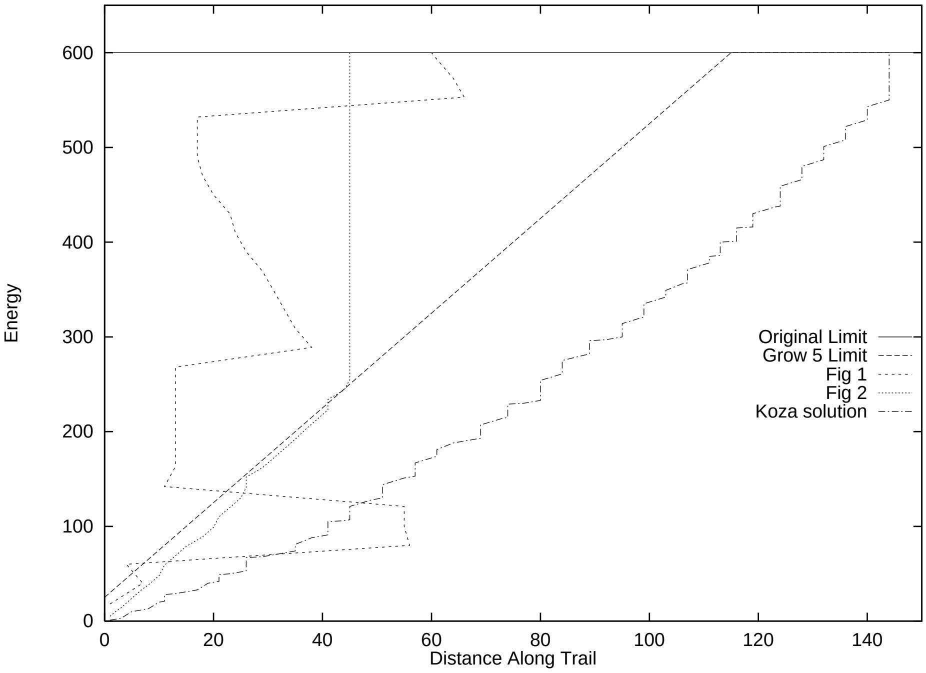

Figure 5: Energy limits and energy used as ants move along the trail. Plots for an evolved solution and the two partial solutions shown in Figs. 1 and 2.

its ant has used as much energy as it has been allocated. The food pellets give the ant a multiple of 5 energy units. With food pellets following gaps giving $5 \times (1 + \text{size of gap})$ energy units (cf. diagonal line Fig. 5).

Effort values when this guidance is used are given in Fig. 4 (plotted with squares) and Table 2 "ramped energy". Disapointingly these show no additional improvement compared to just limiting the amount of food ahead of the ant to five pellets.

3.3 Speed as part of Fitness

In this section we introduce the speed of the ant as a component of the controlling program's fitness. Disapointingly again no significant increase in performance was found.

We investigated two means of combining the ant's speed and the amount of food it eats. In the first speed is added to food eaten to yield a single scalar fitness measure. A threshold of unity is first applied. The threshold means very fast ants have no advantage over ants that are fast enough to complete the whole trail within the time limit and speed accounts for less than one food pellet in the fitness measure. As tournament selection is used the exact scaling co-efficent is unimportant.

We define the speed of the ant as the distance *along the trail* to the furthest pellet it eats divided by the energy it consumed to reach that food pellet (if eats none, its speed is zero). This is linearly scaled by dividing it by the speed required to complete the trail (i.e. 144/600).

The second mechanism was to treat the scaled speed as a second objective. Multi-objective Pareto tournament selection was used to decide which programs were selected for reproduction using the mechanism described in [Langdon, 1998b]. In multi-object tournaments it is possible that two or more members of the tournament are not dominated by any others. E.g. one eats less than the other but moves faster. Rather than decide at random between them, we conduct a secondary tournaments against a sample of the rest of the population. The winner being the one which is worse than the smallest number of other individuals from the (sample of the) rest of the population. (We treat having exactly the same fitness as being worse). Sample sizes of 25 and 200 were tried. In [Langdon, 1998b] use of such a comparsion set was found to be very effective, perhaps as it tends to avoid the population converging on a small number

Table 3: One Point Parameters (as Table 1 unless stated)

Crossover:	100% one point producing one child.
	90% of crossover children mutated
Mutation:	one terminal or function change per individual
Reproduction:	0%

of fitness values. Thresholds of 1.0 and 2.0 on the scaled speed where tried. Again, as Table 2 "Pareto" shows, no improvement above that obtained with simply limiting the amount of food ahead of the ant was seen.

3.4 One Point Crossover

In our previous work [Langdon and Poli, 1998] we suggested reasons why the original Santa Fe ant problem is difficult for Genetic Algorithms. However we hoped that the improved training techniques above would have rendered the search space less deceptive. In this section we investigate using one point crossover [Poli and Langdon, 1997] and point mutation. Performance of one point crossover with point mutation was found to be similar to standard crossover.

The parameters we used were the same as for standard GP crossover (as given in Table 1) except those shown given in Table 3. The principal difference, apart from replacing the crossover operator, is to use point mutation on the children produced by crossover. Normally point mutation is applied with a low probablity for each terminal or function (program node) in the program. When the probablity comes up, point mutation replaces the node with a randomly chosen terminal or function which takes the same number of arguments. Thus point mutation does not change either the mutated program size or shape. Usually there is no check the replacement is different and so there is a finite chance of selecting the same function or terminal, in which case the mutation makes no change. In the case of the ant problem both the terminal and function sets are small and in practice the chance of each point mutation making no change is about 50%. Rather than simply doubling the mutation rate, we provided a difference check. Using this we ensure point mutation is guaranteed to make exactly one change to mutated programs. This avoids doubling the mutation rate, which can lead to many programs suffering multiple changes while still leaving an appreciable fraction unchanged.

The results of one point mutation using both the improved training regime and the original are given in Table 2. From these it is clear that one point crossover with point mutation has a similar performance to standard GP on the Santa Fe problem and the mechanisms limiting initial amount of food or energy (described in the previous sections) have not been demonstrated to be beneficial with one point crossover and point mutation. Other experiments indicate that point mutation on its own is not sufficient and some crossover is needed (albeit at a low rate).

4 Conclusions

By making a modest change to the Santa Fe trail problem we have made it significantly easier for GP. In terms of the effort required (120,000) it is approximately three times easier and GP performs about as well as the best results previously obtained using variable length hill climbers (187,500) [Langdon, 1998c; Langdon and Poli, 1998] and size restricted EP search (136,000) [Chellapilla, 1997]. By enforcing an optimal size limit on the programs being evolved we might be able to do marginally better (104,000, size ≤ 50).

Surprisingly GP performance is only marginally affected by a size limit and is roughly constant for wide ranges in maximium size.

Using the improved fitness function on the original search space, GP is able to out perform the best that might be obtained by random search by about 4:1. If both search techniques are limited to programs of a maximum length of 50 (which best suits GP) this ratio expands to about 25:1.

Acknowledgements

This research was funded by the Defence Research Agency in Malvern.

References

[Chellapilla, 1997] Kumar Chellapilla. Evolutionary programming with tree mutations: Evolving computer programs without crossover. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 431–438, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[Harries and Smith, 1997] Kim Harries and Peter Smith. Exploring alternative operators and search strategies in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 147–155, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[Ito *et al.*, 1998] T. Ito, I. Iba, and S. Sato. Non-destructive depth-dependent crossover for genetic programming. In W. Banzhaf, R. Poli, M. Schoenauer, and T. C. Fogarty, editors, *Proceedings of the First European Workshop on Genetic Programming*, LNCS, Paris, 14-15 April 1998. Springer-Verlag. Forthcoming.

[Jefferson *et al.*, 1992] David Jefferson, Robert Collins, Claus Cooper, Michael Dyer, Margot Flowers, Richard Korf, Charles Taylor, and Alan Wang. Evolution as a theme in artificial life: The Genesys/Tracker system. In Christopher G. Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen, editors, *Artificial Life II*, volume X of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 549–578. Addison-Wesley, February 1992.

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

[Kuscu, 1998] Ibrahim Kuscu. Evolving a generalised behavior: Artificial ant problem revisited. In V. William Porto, editor, *Seventh Annual Conference on Evolutionary Programming*, Mission Valley Marriott, San Diego, California, USA, 25-27 March 1998. Springer-Verlag. to be presented.

[Langdon and Poli, 1998] W. B. Langdon and R. Poli. Why ants are hard. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann. To be presented at GP-98.

[Langdon, 1998a] W. B. Langdon. Better trained ants. In R. Poli, W. B. Langdon, W. Banzhaf, M. Schoenauer, and T. C. Fogarty, editors, *Late Breaking Papers at the First European Workshop on Genetic Programming*, Paris, 14-15 April 1998. Forthcoming.

[Langdon, 1998b] W. B. Langdon. *Data Structures and Genetic Programming*. Kulwer, 1998. Forthcoming.

[Langdon, 1998c] W. B. Langdon. The evolution of size in variable length representations. In *1998 IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, USA, 5-9 May 1998. Forthcoming.

[Lee and Wong, 1995] Jack Y. B. Lee and P. C. Wong. The effect of function noise on GP efficiency. In X. Yao, editor, *Progress in Evolutionary Computation*, volume 956 of *Lecture Notes in Artificial Intelligence*, pages 1–16. Springer-Verlag, Heidelberg, Germany, 1995.

[Luke and Spector, 1997] Sean Luke and Lee Spector. A comparison of crossover and mutation in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 240–248, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[Poli and Langdon, 1997] Riccardo Poli and W. B. Langdon. Genetic programming with one-point crossover. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag London, 23-27 June 1997.