Co-evolution of Operator Settings In Genetic Algorithms

Andrew Tuson and Peter Ross
Department of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, U.K.
{andrewt,peter}@aisb.ed.ac.uk
Tel: +44 (0)131 650 2715

Abstract

Typical genetic algorithm implementations use operator settings that are fixed throughout a given run. Varying these settings is known to improve performance — the problem is knowing how to vary them. One approach is to encode the operator settings into each member of the GA population, and allow them to evolve. This paper describes an empirical investigation into the effect of co-evolving operator settings, for some common problems in the genetic algorithms field. The results obtained indicate that the problem representation, and the choice of operators on the encoded operator settings are important for useful adaptation.

1 Introduction

It has long been acknowledged that the choice of operator settings (such as the mutation rate) has a significant impact upon GA performance. However, finding a good choice is somewhat of a black art. The difficulty is that the appropriate settings depend upon the other GA components. The large number of combinations usually precludes an exhaustive search of the space of operator settings.

In addition the appropriate operator settings may vary during the course of a GA run. For instance, [Davis 91] found that performance is improved by the use of a time-varying schedule of operator settings.

The problem lies in devising such a schedule — this is harder that finding a good set of static operator probabilities. It may be advantageous, therefore, to employ a method that adjusts the operator probabilities according to a measure of the performance of each operators. An investigation of one approach to operator adaptation — co-evolution — will be described in this paper.

2 Adaptation by Co-evolution

Operator adaptation methods based on the co-evolutionary metaphor encode the operator settings onto each member of the population and allow them to evolve. The rationale behind this is as follows: solutions which have encoded operator settings that tend to produce fitter children will survive longer and so the useful operator settings will spread through the population. The original work in this area originated from the 'Evolution Strategy' community [Bäck *et al.* 91].

This paper makes a distinction between an *operator probability* — the probability of an operator being invoked, and any parameters associated with an operator (*operator parameters*), such as the bitwise mutation rate. For example, a GA could use uniform crossover 70% of the time, along with mutation 30% of the time, with the mutation operator possessing a bitwise mutation rate of 0.02. The term *operator setting* will be taken to mean both of the terms above.

Co-evolution is implemented as follows. For any given problem there is a predetermined crossover operator and a mutation operator A chromosome contains the usual problem-specific

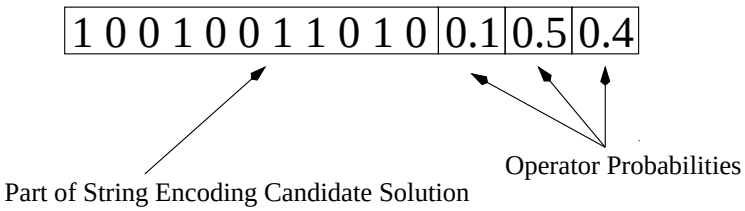

Figure 1: Representing Operator Probabilities

solution encoding, and also a vector of real numbers (Figure 1). In the simplest case, where only operator probabilities are to be co-evolved, the vector encodes the operator probabilities(p_{op}), which are used as follows. A chromosome is chosen by rank-based selection. The encoded p_{op} is used to decide whether a given operator is to be applied. If necessary, a second parent is chosen and the operator applied. Either way, a single child is produced.

If operator parameters are to be co-evolved, the vector would contain these. For example, in parameterised uniform crossover, a probability p_f determines the chance that a gene is copied from the first parent rather than the second. In genewise mutation, a probability p_m gives the probability of randomly changing any given gene.

How do these vectors get changed in the child? They undergo crossover and mutation themselves. We explored two variants: using either *strongly disruptive* or *weakly disruptive* operators. The former uses Radcliffe's random respectful recombination if there are two parents; each probability is randomly chosen from the range delimited by the parents' values. It uses uniform random mutation if there is one parent. The latter variant uses parameterised uniform crossover with parameter 0.1 if there are two parents, so the child's vector is likely to be close to its first parent's. It uses a biased mutation if there is only one parent: the new value is chosen from a Gaussian distribution of SD 0.05 whose mean is the parent's value.

This general idea raises various questions:

- **Should we evolve operator parameters or probabilities?** In this study we consider both, applying these ideas to five well-known problems. We try a steady-state GA and a generational GA with elitism in each case. Population size is kept fixed at 100.
- **What operators should be used on the encoded settings?** In particular should they be *strongly*, or *weakly* disruptive?
- **What level of co-evolution?** An easy approach fixes the co-evolution crossover probability (the probability of applying crossover to the vector of operator settings), but there is a risk that the chosen value will be poor. A higher level approach encodes this probability onto the string too (termed *meta-learning*).

3 The Test Problems

In order to properly study the effectiveness (or otherwise) of adaptation by co-evolution, a set of test problems need to be chosen. The first member of the test suite is a hard scheduling/operations research problem. The other members have been selected on account of their theoretical interest.

All but the flowshop problem use uniform crossover (parameter 0.5) and genewise mutation with probability $1/l$ (l=length) for the solution part of the chromosomes; for the flowshop problem, 'Modified PMX' crossover [Mott 90] and shift mutation [Reeves 95] are used.

Each problem will be briefly described in turn.

- **The Flowshop Sequencing Problem**

	Generational GA		Steady-State GA	
Problem	Solution Quality	Evaluations	Solution Quality	Evaluations
$n/m/P/C_{max}$	2441 (71) [0.95]	7714 (1573) [0.15]	2387 (71) [0.05]	5064 (1616) [0.55]
Max Ones	99.96 (0.20) [0.8]	7714 (1025) [0.8]	100.00 (0.00) [0.0-0.90]	2172 (702) [0.80]
Deceptive	289.68 (2.41) [0.05]	4484 (1883) [0.90]	289.28 (3.22) [0.1]	2095 (1552) [0.95]
Royal Road	35.52 (6.02) [0.35]	5626 (2254) [0.95]	40.64 (7.65) [0.95]	2876 (2260) [0.25]
Long Path	48687 (3441) [0.55]	3976 (1717) [0.95]	48024 (2907) [0.05]	5129 (4088) [0.2]

Table 1: Best Results Obtained Using a GA With Fixed Operator Probabilities

This is an important problem in Operations Research. In the flowshop sequencing, or $n/m/P/C_{max}$, problem jobs are dispatched to a string of machines joined in a serial fashion. The task is to find an ordering of jobs for the flowshop to process, so as to minimise the *makespan* — the time taken for the last of the jobs to be completed. One of the Taillard [Taillard 93] benchmark problems was used: a benchmark RNG seed of 479340445 was used to generate a completion times matrix for a flowshop with 20 jobs and 20 machines.

- **The Counting Ones Problem**

  The simplest of the problems considered here: for a string of binary digits, the fitness of a given string is the number of ones the string contains. The aim is to obtain a string containing all ones. A string length of 100 was used.

- **Goldberg's Order-3 Deceptive Problem**

  The problems that *deception* can present to a GA has been well-studied. A classic problem is such work is the order-3 'tight' deceptive problem devised by Goldberg [Goldberg *et al.* 89]. The problem used is of string length 30 bits.

- **The Royal Road Problem**

  Work by Forrest and Mitchell [Forrest & Mitchell 93] provides the seminal study of this problem. The Royal Road function used in this study (R1) is of length 64 bits

- **The Long Path Problem**

  Recent work [Horn *et al.* 94] has presented a class of problems where a genetic algorithm convincingly outperformed a range of hill-climbing algorithms. The problem was designed to be hard for hill-climbers — not because of local minima (there are none) — but due to the extreme length of the path to the optimum. This study examined the 'Root2Path' function, represented by a binary string of length 29.

4 Results

A large number of experiments were performed in this study; too many, to give here. Therefore a summary of the results are given, and the reader is directed to [Tuson 95] for full results. Two measures of performance were used: the quality of solution obtained after 10000 evaluations, and the number of evaluations after which improvements in solution quality were no longer obtained (or 10000 evaluations — whichever is smaller). Fifty runs were taken in each case.

4.1 The GA With Fixed Operator Settings

The effect of varying crossover probability on a genetic algorithm with fixed crossover probability (p_c) was investigated. An exhaustive search was performed: a genetic algorithm was run for values of p_c in the range 0.0 to 1.0 with steps of 0.05. This provides a benchmark against which the performance of a GA using co-evolution can be compared, and will measure how sensitive this

	Generational GA		Steady-State GA	
Problem	Solution Quality	Evaluations Reqd.	Solution Quality	Evaluations Reqd.
$n/m/P/C_{max}$	2444 (67)	7758 (1688)	2394 (72)	5063 (1628)
Max Ones	9.90 (0.30)	8374 (839)	100.00 (0.00)	2478 (418)
Deceptive	288.28 (2.80)	5692 (2159)	288.60 (2.31)	2518 (2124)
Royal Road	35.36 (6.01)	6794 (1805)	32.80 (9.23)	2759 (1961)
Long Path	49164 (107)	5150 (1797)	46649 (6831)	4622 (4165)

Table 2: Results Obtained Using Co-evolution With Strongly Disruptive Operators

	Generational GA		Steady-State GA	
Problem	Solution Quality	Evaluations Reqd.	Solution Quality	Evaluations Reqd.
$n/m/P/C_{max}$	2444 (73)	7946 (1795)	2397 (71)	5572 (2000)
Max Ones	99.82 (0.43)	8070 (911)	100.00 (0.00)	2475 (842)
Deceptive	288.56 (2.91)	5814 (2275)	289.00 (3.18)	2436 (1592)
Royal Road	35.20 (5.77)	6898 (1907)	31.68 (8.758)	2663 (2202)
Long Path	48687 (3441)	4768 (1870)	44509 (9753)	4933 (4067)

Table 3: Results Obtained Using Co-evolution With Weakly Disruptive Operators

operator setting is to GA performance. This will give some indication of how hard the GA is to tune.

The best results obtained for each problem/population model with crossover probability 0.05-0.95 (extreme values are rarely useful) are given in Table 1. For each entry, the mean, the standard deviation (in parentheses), and the crossover probability at which the sample was taken (in *square* brackets) are given.

When the trends in performance against crossover probability are examined, some general patterns were observed. The choice of operator probabilities appears to depend upon the problem to be solved, the population model (interestingly, the steady-state GA gave consistently better performance), and the performance criterion being used. The final point is illustrated by the deceptive problem: a low crossover probability gives high quality results, a high crossover probability exchanges solution quality for a higher speed of search.

4.2 Co-evolution Of Operator Probabilities

For both strongly and weakly disruptive co-evolution operators, an extensive search was carried out. A genetic algorithm was run 50 times for each of 21 different co-evolution crossover probabilities from 0.0 to 1.0 with steps of 0.05. Experiments were also performed using meta-learning.

The best average performances attained for each problem/population model are given in Tables 2 and 3, with standard deviations given in parentheses. The underlined table entries indicate a significant difference (by t-test) in performance when compared to the conventional GA.

For both operator sets, performance of the GA was found, in all cases, to be insensitive to the choice of co-evolution crossover probability. The effect of using meta-learning was found to be insignificant.

Comparison with a conventional GA shows a drop in performance when co-evolution is used, in many cases. The cases in which performance remained comparable were those that had suitable crossover probabilities of around 0.5; or were insensitive to the value of p_c anyway. The operator set used made little impact.

Why the drop in performance? A more detailed study, which examined the evolution of p_c during the course of the GA run, showed that with strongly disruptive operators, the encoded p_c tended to stay around 0.5; adaptation was having little effect (Figure 2). With weakly disruptive operators, adaptation was found to occur only in the flowshop sequencing and max ones problems and only with a steady-state model (Figure 3). In the other cases adaptation was not seen to

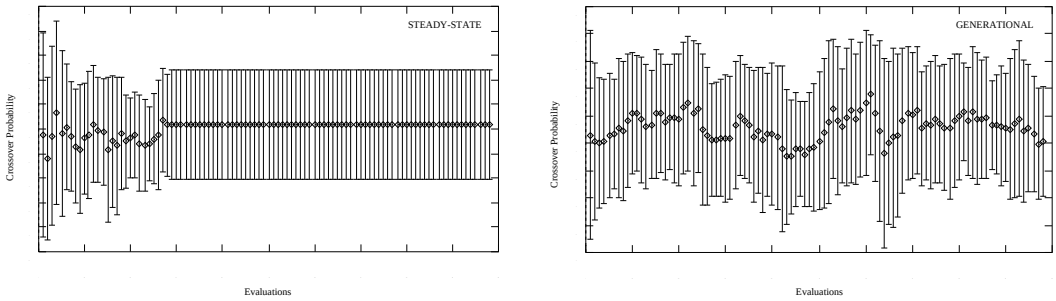

Figure 2: Plots For The Counting Ones Problem Using Strongly Disruptive Co-evolution

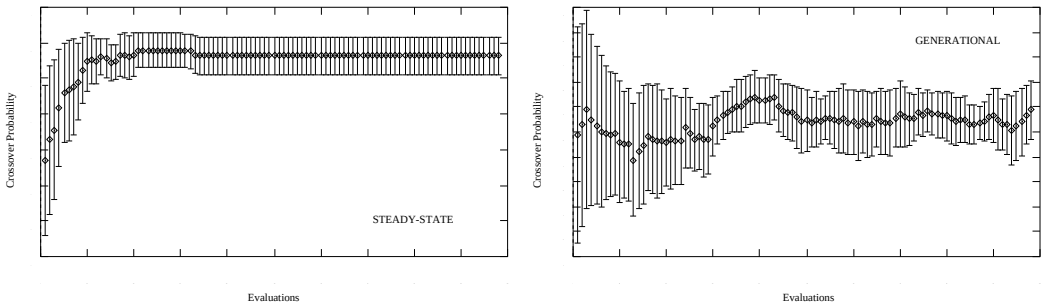

Figure 3: Plots For The Counting Ones Problem Using Weakly Disruptive Operators

occur. Therefore the choice of co-evolutionary operators is important for adaptation.

However, performance was worse even when adaptation was observed. Could this be due to the time taken to evolve a suitable crossover probability? By the time that the GA has found a 'good' crossover probability, the population can be close to the optimum or to convergence anyway, allowing little scope for a positive impact on performance. This requires further investigation. Observing the effect upon GA performance of initialising the crossover probability closer to a known suitable value may provide some answers.

4.3 Co-evolution of Operator Parameters

What happens if the crossover and mutation *parameters* are encoded, rather than their probabilities of being used? The modified PMX crossover operator used in the flowshop sequencing problem has no such parameter, so this problem is omitted here. On the basis of the earlier results, it was decided to only use weakly disruptive operators, without meta-learning. The operator probabilities were fixed throughout the GA run — only the operator parameters evolved.

As before, an exhaustive search was made: a genetic algorithm was run for co-evolution crossover probabilities 0.0 to 1.0 with steps of 0.05. The results obtained for a GA with fixed operator probabilities and default operator parameters are given in Table 4; these will be used as a basis for comparison.

First, it was examined whether adaptation does take place. Plots of the evolved crossover probability against the number of evaluations made so far were obtained (Figure 4).

As can be seen, for the counting ones problem adaptation does take place (to $1/l$ — the theoretical optimum [Bäck 93]), as well as the long path problem. However this was only observed for a generational GA. It is not apparent why — further investigation is needed.

No adaptation was observed at all for either the deceptive or royal road problems. In the case

	Generational GA			Steady-State GA		
Problem	p(Xover)	Solution Quality	Evaluations	p(Xover)	Solution Quality	Evaluations
Max Ones	0.80	99.96 (0.20)	7714 (1025)	0.80	100.00 (0.00)	2172 (702)
Deceptive	0.05	289.68 (2.41)	5960 (1965)	0.20	289.12 (3.08)	3506 (2523)
Royal Road	0.35	35.52 (6.02)	7804 (1390)	0.95	40.64 (7.65)	3786 (1741)
Long Path	0.55	48686 (3441)	5370 (2138)	0.20	47312 (4765)	5129 (4088)

Table 4: The Fixed Operator Probabilities Used To Compare Co-evolution of Operator Parameters And Their Performance

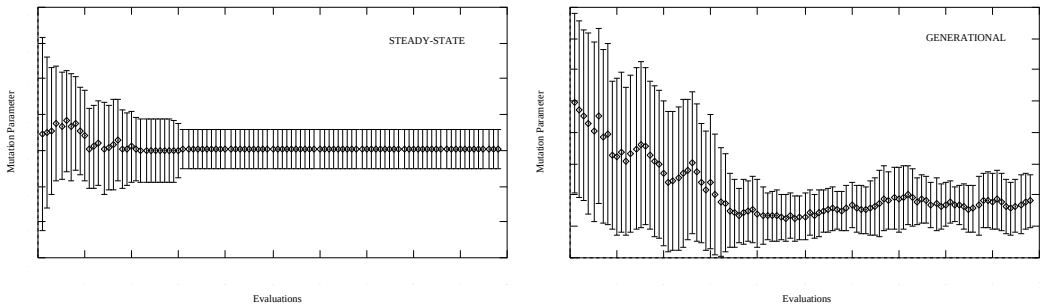

Figure 4: Plots of Mutation Parameter For The Counting Ones Problem

of the deceptive problem, this may be due to the fact that the optimal value for the mutation parameter $(3/l)$ lies in the middle of the range of the encoded mutation parameter. An experiment where the mutation parameter is initialised towards one end of the range should resolve this question. For the royal road problem adaptation may be made difficult by the stepwise fitness function, a result of which would be to make information on mutation parameter performance intermittent.

The performance of the GA was found, in all cases, to be insensitive to the choice of co-evolution crossover probability. When the quality of results (Table 5) is examined for the cases for which adaptation took place, GA performance was found to be degraded. As suggested earlier, this may be a result of the time it takes for the GA to adapt — good choices of the operator parameters are more important earlier in the GA run than later.

The speed to solution for the deceptive problem was improved in this case, presumably due an average mutation parameter close to the theoretically optimal value. In the case of the royal road problem, an exchange of decreased solution quality, in favour of increased speed of search was observed, an effect of the increased mutation parameter.

In any case, a study of the effect of the operator parameters upon GA performance may resolve many of the questions raised here.

	Generational GA		Steady-State GA	
Problem	Solution Quality	Evaluations Reqd.	Solution Quality	Evaluations Reqd.
Max Ones	99.52 (0.64)	8438 (857)	100.00 (0.00)	2791 (1011)
Deceptive	289.12 (2.83)	5306 (1918)	289.32 (2.61)	2555 (2124)
Royal Road	31.36 (6.16)	6086 (2505)	29.76 (8.32)	2428 (1937)
Long Path	46906 (7342)	5892 (2461)	46375 (5523)	5671 (3768)

Table 5: Results Obtained When Co-evolving Operator Parameters

5 Comparison With Previous Work

The results indicate that, at worst, adaptation by co-evolution of operator settings is of little practical use. However, much of the previous work (e.g. [Bäck *et al.* 91]) is more positive. Why is there a difference?

First, virtually all of the work performed previously examined the adaptation of operator parameters. The results obtained for operator parameters in this are more optimistic than for probabilities. Is this a possible indication that the adaptation of operator probabilities should be tackled by a different approach? [Tuson 95] describes a study of a technique called COBRA (COst Based operator Rate Adaptation) proposed in [Corne *et al.* 94], which periodically re-ranks operator probabilities according to some defined measure of operator performance. The results obtained using COBRA were more promising than for co-evolution.

Part of the reason also lies in the emphasis that some of the work has placed upon adaptation being an end in itself — with the implicit assumption that adaptation is a 'good thing'. This work strongly questions this assumption.

Finally, work by [Hinterding 95] looked at adapting operator parameters by co-evolution. The mutation parameter he adapted was the standard deviation of a Gaussian mutation, for problems that have a search space of real numbered parameters (in a similar fashion to the 'Evolution Strategy' community). The paper concluded that for many problems, the adaptation of the mutation parameter lead to improved GA performance. This suggests that binary-encoded problems have less potential for this approach than real-coded problems; further investigation is required to confirm this.

6 Conclusions

This investigation showed that the choice of operator probabilities has a marked effect upon GA performance, and that this choice can change dramatically with the other components of the GA.

The choice of co-evolution operators was found to have a dramatic effect: disruptive operators were found to remove the ability to adapt as they destroy any information gained by selection. The use of operators of low disruption improved matters somewhat, but the occurrence of adaptation was found to be problem dependent and unreliable.

Needless to say, when no adaptation was found to take place the effect upon performance was often found to be detrimental. However, performance was seen to decline even when adaptation took place. Part of the reason for this is that it takes time for the crossover probability to evolve to the right value, by which time much of the useful search has already been performed and the impact of the evolved crossover probability is much reduced — getting the operator probabilities right at the start of the GA run appears to be important. No trends in the externally set co-evolution crossover probability were found, but as adaptation does not occur reliably, if at all, this is not particularly surprising.

Encoding of operator parameters was found to be more successful: adaptation was observed more often, however performance was still found to be degraded — the time taken to adapt precludes efficient search at the early stages of the GA run. Genetic algorithm performance was found to be sensitive to the operator parameters used. Comparison with similar work suggests that it may be the case that binary-encoded problems have less potential for this approach than real-coded problems.

7 Acknowledgements

Thanks to the Engineering and Physical Sciences Research Council (EPSRC) for their support of Andrew Tuson via a studentship with reference 95306458.

References

[Bäck 93] T. Bäck. Optimal Mutation Rates In Genetic Search. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Mateo: Morgan Kaufmann, February 1993.

[Bäck *et al.* 91] T. Bäck, F. Hoffmeister, and H. P. Schwefel. A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9. San Mateo: Morgan Kaufmann, 1991.

[Corne *et al.* 94] D. Corne, P. Ross, and H.-L. Fang. GA Research Note 7: Fast Practical Evolutionary Timetabling. Technical report, University of Edinburgh Department of Artificial Intelligence, 1994.

[Davis 91] L. Davis, editor. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[Forrest & Mitchell 93] S. Forrest and M. Mitchell. Relative Building Block Fitness and the Building Block Hypothesis. In L. Darrell Whitely, editor, *Foundations of Genetic Algorithms 2*. San Mateo: Morgan Kaufmann, 1993.

[Goldberg *et al.* 89] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

[Hinterding 95] R. Hinterding. Representation and Self-adaption in Genetic Algorithms. In *Proceedings of the First Korea-Australia Joint Workshop on Evolutionary Computation*, Sept 1995.

[Horn *et al.* 94] J. Horn, D. E. Goldberg, and K. Deb. Long Path Problems. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature, PPSN III*, pages 149–159. Springer Verlag, 1994.

[Mott 90] G. F. Mott. Optimising Flowshop Scheduling Through Adaptive Genetic Algorithms. Chemistry Part II Thesis, Oxford University, 1990.

[Reeves 95] C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Ops. Res.*, 22:5–13, 1995.

[Taillard 93] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of operations research*, 64:278–285, 1993.

[Tuson 95] A. L. Tuson. Adapting Operator Probabilities In Genetic Algorithms. Unpublished M.Sc. thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1995.