Cost Based Operator Rate Adaptation: An Investigation

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. In the vast majority of genetic algorithm implementations, the operator probabilities are fixed throughout a given run. However, it may be useful to adjust these probabilities during the run, according to the ability of the operators to produce children of increased fitness. Cost Based Operator Rate Adaptation (COBRA) periodically re-ranks operator probabilities according to a measure of operator performance. The effect upon genetic algorithm performance of COBRA upon both well-studied theoretical and practical problems is examined.

1 Introduction

It has long been acknowledged that the choice of operator probabilities have a significant impact upon GA performance. However, finding a good choice is somewhat of a black art. The appropriate choice depends upon the other GA components, such as the population model, the problem to be solved, its representation, and the operators used.

The above also ignores the case for varying operator probabilities during the course of a GA run. Davis [3] advocates the use of a time-varying schedule of operator probabilities.

The problem lies in knowing how to vary — 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 of the operators.

This paper investigates one such approach: COBRA (COst Based operator Rate Adaptation) [1], a method that was originally devised for adapting operator probabilities in timetabling problems.

2 What is COBRA?

First, some terminology will be described. Each operator available to the GA has a probability of being fired — an *operator probability*. This study makes a distinction between this and any parameters associated with a given operator (an *operator parameter*). For example, a GA could use uniform crossover 70% (operator probability) of the time, along with mutation 30% of the time, with the mutation operator possessing a bitwise mutation rate of 0.02 (operator parameter). This study concerns itself only with the adaptation of operator probabilities.

COBRA is a member of the group of 'learning-rule' adaptation methods, such as the techniques devised by Davis [2], and Julstrom [7]. This approach explicitly collects information on operator performance and uses this to adjust the operator probabilities.

COBRA is implemented as follows. Given k operators $o_1, ..., o_k$, let $b_i(t)$ be the *benefit*, $c_i(t)$ the *cost* (the amount of computational effort to evaluate a child), and $p_i(t)$ the probability of a given operator, i at time t. We then apply the following algorithm:

1. The user decides on a set of initial probabilities p_i.
2. After G (the gap between operator probability readjustments) evaluations, rank the operators according their values of b_i/c_i, and assign the operators their new probabilities according to their rank (i.e. the highest probability to the operator with the highest value of b_i/c_i).
3. Repeat step 2 every G evaluations.

This study used *operator productivity* as the measure of benefit. This is defined as the average increase in fitness when a child was produced that was fitter than its parents (i.e. if the child is worse than the parent, then the productivity is zero) over a set interval (in this case all operations since the last probability re-ranking).

The variables in the adaptation method come from two sources: firstly the gap between operator probability readjustments G, and secondly the initial operator probabilities provided by the user. The effect that these variables have upon the genetic algorithm will be investigated.

3 The Test Problems

In order to study the effectiveness (or otherwise) of adaptation using COBRA, 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 familiarity. Each will be briefly described in turn.

- **Flowshop Sequencing**. A Taillard [11] benchmark problem: 20 jobs and 20 machines with RNG seed 47934044.
- **Max Ones** of length 100.
- **Order-3 Deceptive** [5] of length 30.
- **Royal Road** (R1) [4] of length 64.
- **Long Path** [6] of length 29.

The representation used for the flowshop sequencing problem is a permutation of the jobs to be placed into the flowline, numbered 1 to n. The operators used were 'Modified PMX' crossover [8], and shift mutation [9].

The operators used for the binary-coded problems were: binary mutation with a bitwise mutation probability of $1/l$ (where l is the length of the string) and uniform crossover.

4 The Investigation

A full description how the genetic algorithm used was implemented is given in [12]. Two population models were used as part of this study, in order to see if this had any effect upon the success (or otherwise) of COBRA. The first population model used was *steady-state* reproduction with a *kill-worst* replacement policy; the second model was *generational* replacement with *elitism*, keeping the single best chromosome.

A large number of experiments were performed in this study. A summary of the results are given here, and the interested reader is directed to [12] 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)

Samples of 50 GA runs were taken, for each of the two population models, with a population size of 100 and a rank-based selection pressure of 1.5. Where differences in performance are reported to be significant, a t-test was applied in order to ascertain if this was the case.

4.1 The GA With Fixed Operator Settings

The effect of varying crossover probability on a genetic algorithm with fixed operator probabilities was investigated. An exhaustive search was made of the operator probabilities: a GA was run for crossover probabilities 0.05 to 0.95 with steps of 0.05 (mutation was applied otherwise). This will provide a benchmark against which the performance of a GA using COBRA will be compared, and measure how sensitive this operator setting is to GA performance. This will give some indication of how hard the GA is to tune, and will allow later comparison of tuning difficulty when co-evolution is used.

The operator probabilities found to be suitable for each problem/population model are given in Table 1. For each entry, the standard deviation is given in parentheses. Note that for the flowshop sequencing problem the aim is to *minimise* the makespan (fitness).

When the trends in performance against crossover probability are examined, some general patterns were observed (reflected somewhat in Table 1). The choice of operator probabilities appear 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.

	Generational GA			Steady-State GA		
Problem	Xover	Fitness	Evaluations	Xover	Fitness	Evaluations
Flowshop	0.65	2444 (76)	8236 (1285)	0.35	2389 (69)	5064 (1616)
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 1. The Mean Performance of the Standard GA Used To Compare COBRA

4.2 The Effect of Varying The Gap Between Probability Re-Rankings

The previous study [1] did not investigate the effect of varying G. This investigation examined a range of values of G (from 200 to 2000 evaluations in steps of 200). The mean performance (in terms of fitness and speed) for each of these problems is given in Tables 2 and 3.

	Re-Ranking Interval (G)									
Steady-State	200	400	600	800	1000	1200	1400	1600	1800	2000
Flowshop	2398	2397	2398	2401	2399	2397	2398	2392	2394	2395
Max Ones	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Deceptive	287.72	287.52	287.44	288.04	287.96	287.92	288.08	288.32	288.40	288.44
Royal Road	40.00	39.36	40.80	40.48	40.32	40.48	40.80	40.64	40.96	40.64
Long Path	45675	43613	40568	41867	44704	44117	42998	43320	42055	42770
Generational	200	400	600	800	1000	1200	1400	1600	1800	2000
Flowshop	2445	2444	2444	2444	2444	2444	2444	2444	2444	2444
Max Ones	99.96	99.96	99.96	99.96	99.96	99.96	99.96	99.96	99.96	99.96
Deceptive	285.44	285.16	286.08	286.24	286.24	286.56	286.84	286.88	287.24	287.44
Royal Road	32.80	33.60	34.08	34.08	33.44	33.28	32.96	33.92	33.60	35.04
Long Path	47459	47843	48686	47887	47950	47696	48195	48687	48687	48687

Table 2. Mean Fitness Against Re-Ranking Interval (G)

For three of the problems considered here, the choice of G had no significant effect upon GA performance. The exceptions were the deceptive and long path problems.

To interpret these results it is necessary for us to know what COBRA is doing during the course of a GA run. Therefore plots of the *operator productivities* were then obtained for a conventional GA with crossover probability 0.5, to see which operator was providing the most improvements at a given stage of a GA run. The

	Re-Ranking Interval (G)									
Steady-State	200	400	600	800	1000	1200	1400	1600	1800	2000
Flowshop	5767	5979	5437	6081	5271	5862	5100	5993	5597	5470
Max Ones	2118	2139	2172	2172	2172	2172	2172	2172	2172	2172
Deceptive	2255	2463	2025	2334	2323	2119	2233	2220	2361	2349
Royal Road	3888	3741	3922	3910	4012	3845	3826	3800	3920	3830
Long Path	5120	6378	6326	6958	5575	5044	5305	5567	5555	5260
Generational	200	400	600	800	1000	1200	1400	1600	1800	2000
Flowshop	8098	8236	8236	8236	8236	8236	8236	8236	8236	8236
Max Ones	7714	7714	7714	7714	7714	7714	7714	7714	7714	7714
Deceptive	4322	4422	4194	4140	4244	4314	4160	4534	4204	4488
Royal Road	7664	7580	7728	7446	7616	7152	7026	7670	7410	7810
Long Path	5306	5344	5534	5418	5516	5536	5632	5434	5312	5280

Table 3. Mean Evaluations Reqd. With Re-Ranking Interval (G)

plots shown in Figures 1 and 2 tend to be typical for the problems considered here — crossover is consistently the more productive operator.

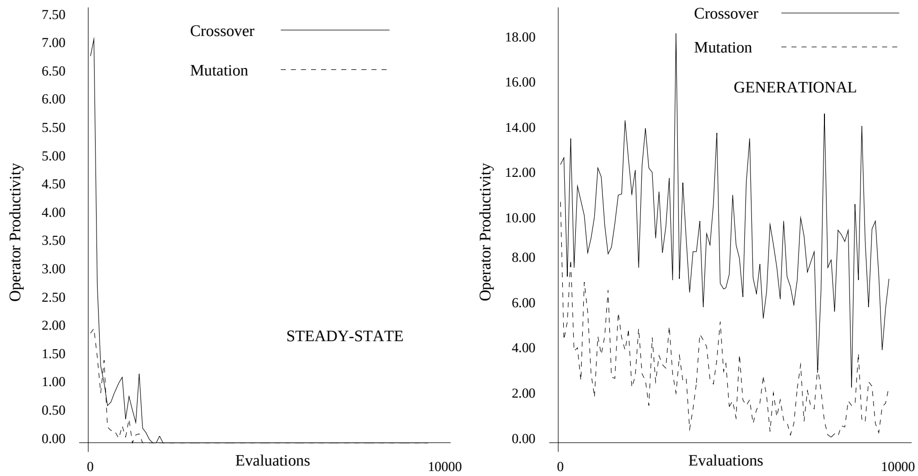

Fig. 1. Operator Productivities For The Deceptive Problem

In the case of the deceptive problem, a trade-off was observed — low values of G corresponding to an increased speed of search, although at the expense of solution quality: high values of G favouring a higher quality of solution. The reason for this behaviour is closely linked to the effect of the crossover probability

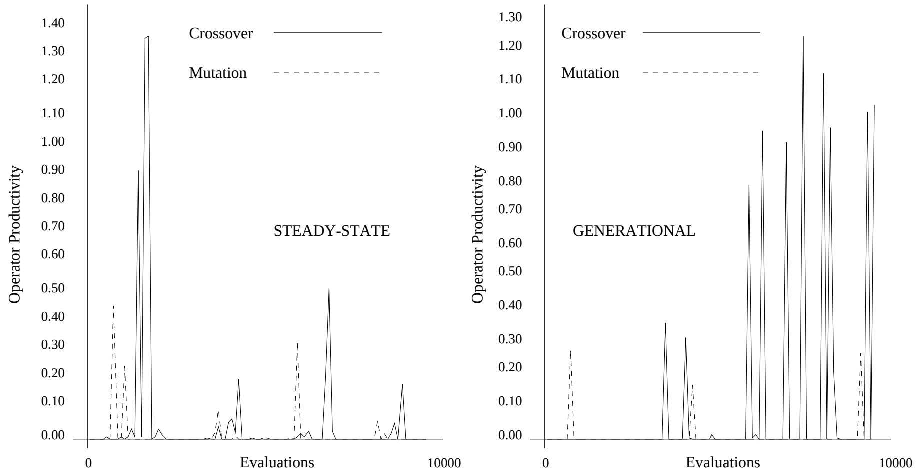

Fig. 2. Operator Productivities For The Long Path Problem

for the static GA — high crossover probabilities lead the GA to the deceptive optimum more quickly, leading to a faster speed of search whilst sacrificing quality. As crossover was consistently the more productive operator the GA will assume a high crossover probability. For low G this will occur earlier in a GA run and favour speed over quality.

For the long path problem, behaviour varied according to the population model used. When a generational model was used, solution quality was affected, rising with increased G. Presumably the larger samples prevent spurious re-ranking due to noise in the operator productivity information.

The trends observed with a steady-state GA were quite the opposite: solution quality was degraded when compared to a conventional GA, declining further with increasing G. This is a result of crossover being the more productive operator, most of the time (Figure 2). Unfortunately the preferred crossover probability is low (0.2). Therefore COBRA will mostly adopt a high crossover probability with the effect of degrading solution quality. However, when G is small, there is a greater chance that 'spurious' re-rankings to a lower (i.e. better) crossover probability will occur which reduces the adverse effect of COBRA somewhat.

4.3 The Effect of Varying The Initial Operator Probabilities

The relationship between the performance of COBRA and the initial crossover probability was investigated. For all problems, the performance of crossover probabilities from 0.05 to 0.95 (in steps of 0.05 with the exception of 0.5) was examined. Table 4 displays the range of performance attained, for all problems and population models, with and without the use of COBRA. It was apparent, in most cases, that COBRA was less sensitive to the initial crossover probability, than a conventional GA. COBRA appears to mitigate the effect of bad choices somewhat.

	Generational GA		Steady-State GA	
Problem	Static GA	GA w/COBRA	Static GA	GA w/COBRA
Flowshop: Fitness	2441-2453	2438-2454	2387-2420	2396-2411
Max Ones: Fitness	93.38-99.96	98.36-99.96	99.90-100.00	99.84-100.00
Deceptive: Fitness	284.20-289.68	284.92-288.04	286.28-289.28	286.28-288.48
Royal Road: Fitness	22.72-35.52	25.28-35.84	25.76-40.64	26.88-40.32
Long Path: Fitness	35824-48687	41407-49177	28287-48025	40955-44723
Flowshop: Evals	7714-8914	7824-8914	5064-7717	5045-6641
Max Ones: Evals	7714-9228	7714-8486	2172-5645	2172-6526
Deceptive: Evals	4484-6212	4244-6412	2192-4210	1979-3409
Royal Road: Evals	5626-7804	5596-7620	2876-3952	2607-4012
Long Path: Evals	3976-7848	4286-5702	5129-6736	5575-7437

Table 4. GA Performance With Crossover Probabilities In The Range 0.05-0.95

But does this affect performance? The best performance attainable appears, in most cases, to be unaffected by COBRA. The exceptions are: the deceptive problem (for both population models), and the long path problem.

In the case of the deceptive problem, the best solution quality attained is reduced, but with a corresponding increase in speed. This trade off was found to be controllable by the initial crossover probability. The reason for this is similar to the reason for the trends for G — the higher productivity of crossover means that a high crossover probability is adopted, which favours speed of search at the expense of quality.

The results for the long path problem, however, are disappointing. When a steady-state model was used, solution quality was significantly degraded. The reason for this is, as for the trend in G, largely due to the predominantly higher crossover productivity (Figure 2). This results in COBRA adopting a high crossover probability, causing a reduction in solution quality.

5 Conclusions

First, it was established that, by examining the effect of the crossover probability on a conventional GA, that the choice of crossover probability is dependant upon the problem to be solved, the population model, and the performance measure used.

Performance was found to be relatively robust towards the gap between operator probability re-ranking (most often a slight upward trend in performance was observed as a result of the better sampling of the operator productivities).

Instead it was found that the initial operator probabilities were the main factor affecting performance.

No improvement in performance was found to occur when COBRA was being used, however the GA was often made less sensitive to the operator probabilities provided when COBRA was used (it appears to reduce to effect of bad choices), which in some applications may be useful — it may well be easier than looking over a larger number of conventional GA runs to obtain equivalent performance. Also, the performance of COBRA (on the basis of the results for the deceptive problem) may be a technique that favours speed of search, at the possible expense of quality.

COBRA was found to be detrimental for some problems (such as long path with a steady-state population model). This appears to manifest itself in cases where the productivity of an operator is high (thus COBRA assigns it a high probability), but the preferred probability of the operator is low. In these cases COBRA should not be used.

This suggests that operator productivity is a poor basis with which to assign operator probabilities for a static GA, and possibly misleading for COBRA. Operator productivity is but one factor to consider; others include the maintenance of diversity lost by processes such as drift and premature convergence by the mutation operator.

Fortunately, one class of problems are known to exhibit improved performance when COBRA is used: timetabling problems. In addition, recent work [10] has shed some light upon what makes timetabling problems easy or hard for a genetic algorithm. Therefore it would be advantageous to use timetabling problems as a test-bed by taking examples for which COBRA was found to be effective and modifying them to see what effect it has.

To summarise, COBRA has promise as a means to remove some of the parameter tuning that can bedevil GA applications. But like many other such devices, there is the danger of being misled into a poor choice of operator probabilities by some problems.

6 Acknowledgements

Thanks to Dave Corne for his advice on COBRA. Thanks also to the Engineering and Physical Sciences Research Council for their support of Andrew Tuson via a research studentship with reference 95306458.

References

1. 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.
2. L. Davis. Adapting Operator Probabilites in Genetic Algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms and their Applications*, pages 61–69. San Mateo: Morgan Kaufmann, 1989.

3. L. Davis, editor. *Handbook of Genetic Algorithms.* New York: Van Nostrand Reinhold, 1991.
4. 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.
5. D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.
6. 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.
7. Bryant A. Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 81–87, San Francisco, Ca., 1995. Morgan Kaufmann.
8. G. F. Mott. Optimising Flowshop Scheduling Through Adaptive Genetic Algorithms. Chemistry Part II Thesis, Oxford University, 1990.
9. C. R. Reeves. A genetic algorithm for flowshop sequencing. *Computers & Ops. Res.*, 22:5–13, 1995.
10. P. Ross and D. Corne. The phase transition niche for evolutionary algorithms in timetabling. In *Proceedings of the First International Conference on the Theory and Practice of Automated Timetabling*, Edinburgh, 1995. Napier University.
11. E. Taillard. Benchmarks for basic scheduling problems. *European Journal of operations research*, 64:278–285, 1993.
12. A. L. Tuson. Adapting Operator Probabilities In Genetic Algorithms. Master's thesis, Department of Artificial Intelligence, Univeristy of Edinburgh, 1995.

This article was processed using the LaTeX macro package with LLNCS style