Selective Crossover in Genetic Algorithms: An Empirical Study

Kanta Vekaria and Chris Clack

Department of Computer Science
University College London
Gower Street
London WC1E 6BT
United Kingdom
Email: {K.Vekaria, C.Clack}@cs.ucl.ac.uk

Abstract. The performance of a genetic algorithm (GA) is dependent on many factors: the type of crossover operator, the rate of crossover, the rate of mutation, population size, and the encoding used are just a few examples. Currently, GA practitioners pick and choose GA parameters empirically until they achieve adequate performance for a given problem. In this paper we have isolated one such parameter: the crossover operator. The motivation for this study is to provide an adaptive crossover operator that gives best overall performance on a large set of problems. A new adaptive crossover operator "selective crossover" is proposed and is compared with two-point and uniform crossover on a problem generator where epistasis can be varied and on trap functions where deception can be varied. We provide empirical results which show that selective crossover is more efficient than two-point and uniform crossover across a representative set of search problems containing epistasis.

1. Introduction

In the canonical genetic algorithm (GA) [Gold89], individuals are represented as fixed length binary vectors, recombination is implemented as a crossover operator, mutation is an additional operator to provide diversity in a population and the population is generational. Recombination is regarded as the driving force of a GA and is the process where segments (genes) of two individuals (parents) are exchanged to produce two new individuals (children). The number of crossover points is decided beforehand and is usually limited to 1 or 2 [Holl75].

Since Holland's work the field of GAs has grown tremendously. Along with this growth we now have a pool of genetic operators, where different permutations will yield alternative GAs (the mimicking of natural evolution is still retained). The alternative GAs have varying degrees of performance – for one kind of problem some will do extremely well but for another they will do poorly. One can argue that many factors contribute to a GA's performance such as population size, crossover/mutation rates, method of selection, representation and the recombination operator and hence

A.E. Eiben et al. (Eds.): PPSN V, LNCS 1498, pp. 438-447, 1998.

this makes it difficult to pinpoint which operators to choose for optimum performance. In this research we have isolated one operator, the recombination operator.

There are now many different methods for recombination [Spea97] but one-point, two-point and uniform crossover are generally those that are commonly used. Even with just three crossover operators it has been difficult to decide a priori which form of crossover to use. Users typically pick and choose the operators in the hope that they will give optimum performance. We know from "no free lunch" theorems [WoMa95] that an operator that is suited for all problems cannot exist, but is it possible to devise a form of crossover that is suited for most practical problems?

The motivation for this paper is to present a new form of adaptive crossover "Selective Crossover" which gives better performance (i.e. the number of evaluations required to find a solution) than two-point and uniform crossover. The reason we have measured the number of evaluations is because the vast majority of the computation involved in a GA is during evaluation. The GA used in this study only re-evaluates individuals if they change (by crossover or mutation).

2. Exploration and Exploitation

An issue that is of great concern in the GA community is the balance between exploration and exploitation. An efficient optimisation algorithm is one that uses two techniques: exploration to investigate new and unknown areas in a search space and exploitation to make use of knowledge acquired by exploration to reach better positions on the search space. Pure random search is good at exploration, but has no exploitation. Hill climbing is good at exploitation but has little exploration. Genetic algorithms combine both strategies, but crossover operators have varying degrees of exploration and exploitation [EsCaSc89].

The combination of exploration and exploitation is effective but it is difficult to know where the balance lies. Many adaptive techniques [ScMo87][Davi89][WhOp94] have been introduced to provide a balance between exploration and exploitation and attempt to solve the problem of finding optimal parameters. For this study we will assume the balance lies with a strategy which gives the best overall performance across a wide range of problems (of varying epistasis; most 'real world' problems contain some epistasis). The balance is a compromise between generalisation (a crossover operator which works well with many different problems) and specialisation (a crossover operator which is optimum for a single problem). This balance can provide a good general-purpose strategy and can also be a starting point for an adaptive GA to favour specialisation.

3. Selective Crossover - Inspiration from Nature

The inspiration for our new adaptive crossover operator ("selective crossover") comes from nature, specifically Dawkin's model of evolution and dominance characteristics in nature. Dawkin's model of evolution is based on the gene [Dawk89]. He presents his theory of the gene as the fundamental unit of natural selection. Chromosomes

have a life span of one generation but a genetic unit lasts for many generations, thus natural selection favours the genetic unit.

Dominance in nature is usually associated with genetic material presented using diploid chromosomes. In the diploid form a genotype carries one or more *pairs* of chromosomes, each containing information for the same functions. The genes contained in one set can be regarded as a direct alternative to the genes in the other set. When building the body the genes in one set compete with those in the other set. Genes that are dominant are expressed in the phenotype of an organism and those that are less likely to be expressed are recessive. The relationship between a dominant and recessive gene is complex: some genes that have been known to be dominant have become more recessive in successive generations and vice versa. These shifts in dominance are due to changes in fitness of an individual with respect to their environment change. Those genes that increased an individual's fitness became dominant. These dominance characteristics have evolved over generations.

Selective crossover is very much like "dominance *without* diploidy". It uses an extra vector that accompanies the chromosome to accumulate knowledge of what happened in previous generations and uses that to promote successful genes (individual bits) during crossover onto the next generation.

4. Implementation

To mimic the dominance characteristics of a gene, to bias genes during crossover, each individual (chromosome) has associated with it a real-valued vector, and thus each gene has an associated *dominance value*. Each recombination uses two parents to create two children. During recombination two parents are selected and their fitness is recorded. 'Parent 1' is considered as the *contributor*. The dominance value of each gene in both parents is compared linearly across the chromosome. The gene that has a higher dominance value contributes to 'Child 1' along with the dominance value. If both dominance values are equal then crossover does not occur at that position. Fig. 1 gives an example of selective crossover: the shaded genes have a higher dominance value than its competing gene. To keep diversity in the population 'Child 2' inherits the non-dominant genes. The need for this will become apparent later.

After crossover the two new children are evaluated. If a single child's fitness is greater than the fitness of either parent, the dominance values (of those genes that were exchanged during crossover) are increased proportionately to the fitness increase. This is done to reflect the genes' contribution to the fitness increase. Fig. 2 gives an example of the mechanism. It follows on from the selective crossover example given in Fig. 1. In Fig. 2, only 'Child 1' has an increase in fitness of 0.1 (compared with the fittest parent) hence its dominance values get updated. In Fig. 1 the bit values of 'Parent 1' and 'Parent 2' at loci 1 and 2 did not get exchanged during crossover and the bit values at loci 4 and 6 are the same. Thus, after selective crossover, the genes that effected the change in the chromosome are only those held at loci 3 and 5. Since the change of those genes at loci 3 and 5 resulted in an increase in fitness, only their dominance values get increased in 'Child 1' (shaded in Fig. 2).

Parent 1 – fitness = 0.36

0.4	0.3	0.01	0.9	0.1	0.2
1	0	0	1	0	0

Parent 2 – fitness = 0.30

0.01	0.2	0.4	0.2	0.9	0.3
0	1	1	1	1	0

Child 1

0.4	0.3	0.4	0.9	0.9	0.3
1	0	1	1	1	0

Child 2

0.01	0.2	0.01	0.2	0.1	0.2
0	1	0	1	0	0

Fig. 1. Selective Crossover

Child 1 – fitness = 0.46

0.4	0.3	0.4	0.9	0.9	0.3
1	0	1	1	1	0

Child 2 – fitness = 0.20

0.01	0.2	0.01	0.2	0.1	0.2
0	1	0	1	0	0

⇩ **Increase dominance values**

Child 1 – fitness = 0.46

0.4	0.3	0.5	0.9	1.0	0.3
1	0	1	1	1	0

Child 2 – fitness = 0.20

0.01	0.2	0.01	0.2	0.1	0.2
0	1	0	1	0	0

Fig. 2. Biasing Genes

On initialisation the dominance values are randomly generated, as is the population, but are restricted to be in the range [0,1]. By doing this we are allowing the GA to explore the search space by evolving the dominance values – to determine and promote those genes which are considered fit. 'Child 2' is needed so that important information is not lost in early generations when there is more exploration than exploitation. That way if 'Child 2' was to produce an increase in fitness to that of its parents then its genes will get promoted. Thus selection will bias the fitter individuals and lose the least fit and their dominance vectors.

The dominance values get increased when the fitness increases therefore it follows that one should decrease the dominance values when fitness decreases. We choose not to do this because we prefer not to introduce a strong bias during the early (highly explorative) generations. In our example (Fig. 2) 'Child 2' showed a fitness decrease. This child may be a prospective parent in the next generation. By not decreasing the dominance values we still allow the genes to compete with other genes (at the same locus in the population). If we were to decrease them they may never get chosen, hence introducing a strong bias in early generations: this form of bias is left for selection.

Unlike one-point or two-point crossover, selective crossover is not biased against schema with high defining length (as defined by Holland [Holl75]). Selective crossover propagates good schema regardless of their defining length – for example, if a schema consists of interacting genes at the two extremes of the chromosome, it can be propagated as easily as a schema which consists of interacting genes located adjacent to each other. Selective crossover can be considered as an extension of uniform crossover. With selective crossover the probability of crossing over at a position is dependent on what happened in previous generations whilst in uniform crossover the probability it fixed throughout (traditionally at 0.5).

5. Experiments

To determine whether a certain operator gives best overall performance we really should apply it to all problems. Since the problem domain is infinite we restrict our experiments to a small set of problems which display common and challenging characteristics: the "random L-SAT problem generator" [DePoSp97] and "deceptive trap functions" [DeGo92]. The random L-SAT problem generator allows epistasis (interaction between genes) to be varied and the deceptive trap function allows deception to be varied. Most complex problems contain some, and possibly a great degree of, epistasis and deception; these problems are thus a good representative set for practical problems.

Each experiment is averaged over 50 independent runs. In order to have a strict comparison between the operators each population is generated with the same seed (50 different seeds were used for the 50 runs). The probability of crossover (P_c) and mutation (P_m) for the GA were fixed for all runs at *0.6* and *0.001* respectively. The GA uses fitness proportionate selection (Stochastic Universal Sampling [Baker87]).

5.1. Random L-SAT Problem Generator

The random L-SAT problem generator [DePoSp97] is a boolean expression generator. It creates random problems in conjunctive normal form subject to three parameters V (number of boolean variables), C (number of disjunctive/conjunctive clauses) and L (the length of the clauses). Selecting L of the V variables uniformly randomly and negating each variable with probability 0.5 generates each clause.

The fitness function for the L-Sat Problem is:

$$f(chrom) = \frac{1}{C}\sum_{i=1}^{C} f(clause_i)$$

Where *chrom* consists of C clauses, $f(clause_i)$ is the fitness contribution of each clause and is 1 if the clause is satisfied or 0 otherwise. Since the problem generator randomly generates problems on demand, there is no guarantee that such an assignment to the expression exists. The difficulty of the problem increases as a function of the number of boolean variables and the complexity of the boolean expression. Increasing the number of clauses increases the epistasis and complexity.

The forms of epistasis that exist in nature are pleiotropy (a gene may influence multiple traits) and polygeny (a trait maybe influenced by multiple genes). L-Sat problems are encoded as binary bit chromosomes, where each bit represents a boolean variable. Hence each clause can be regarded as a trait and thus the polygeny is of order L and the pleiotropy can be estimated because on average each variable occurs in CL/V clauses. By varying parameters V, C, and L we can vary both the type and amount of epistasis.

In our experiments we used the same parameters as De Jong, Potter and Spears. We keep V and L fixed and we vary the number of clauses C to increase or decrease the amount of pleiotropic epistasis. The number of variables V is fixed to 100 and the

clause length L is set to 3. The number of clauses C is varied from 200 (low epistasis) to 1200 (medium epistasis) to 2400 (high epistasis). The chromosome length is V. The population size is 100. The GA was allowed to run for 600 generations and the number of evaluations administered was recorded at the end of each run or when a global solution was found. The results are shown in Section 6.1.

5.2. Deceptive Trap Functions

For this study we have only used a partially deceptive problem where no tight ordering exists and a single function, rather than subfunctions, is represented in the chromosome. Trap functions [Ackl87] are piecewise-linear functions of unitation [DeGo92]. A unitation *u* is defined as the number of 1s in a string and is considered to be the Hamming distance of the string from the local optimum. They depend only on the number of 1s in an individual and not on the positions of the 1s. A trap function divides the search space into two peaks in the Hamming Space; one leads to the global optimum and the other to a local optimum. An example of a trap function is given in Fig. 3.

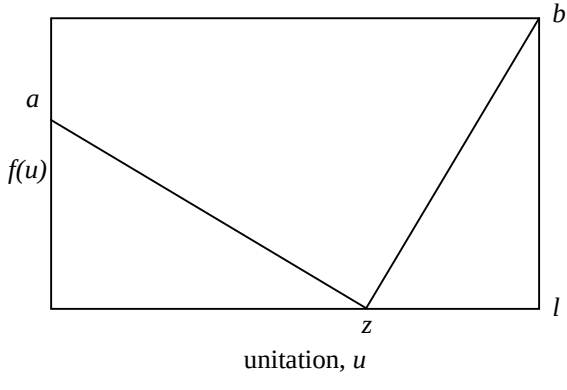

Fig. 3. A trap function with a global optimum at $u=l$.

The fitness function for a Trap Function is:

$$f(u) = \begin{cases} \frac{a}{z}(z-u), & \text{if } u \le z; \\ \frac{b}{l-z}(u-z), & \text{otherwise;} \end{cases}$$

Where *a* and *b* are constants, *l* is the length of the chromosome, *u* is the number of 1s in the string and *z* is the slope change location. Deb and Goldberg [DeGo92] analysed deception in trap functions. They defined a parameter *r* to be the ratio of the locally to the globally optimal function values ($r = a/b$) and thus delineated boundaries between functions that are not deceptive, partially deceptive and fully deceptive. “A function is defined to be partially deceptive to an order *k* if all schema partitions of order less

than *k* are deceptive" [DeGo92]. An order *k* trap function is one where *k* of the *l* bits in the chromosome (not necessarily situated together) are deceptive. Therefore the global optimum is when all *l* bits are 1s and the local optimum is when the order *k* schemata consists of 0's (*k* bits are 0s and *l*-*k* are 0s). In our experiments we use these boundaries to vary the amount of deception in a trap function.

The fitness value is scaled so that the maximum fitness is 1.0, thus $b = 1$. The parameters a and z are varied to increase or decrease deception. A fully deceptive trap function is one where all schemata with *l*-1 defined bits (the "order *l*-1" schemata) are misleading. Experiments were carried out on easy (no deception) and partially deceptive trap functions. Due to the size limitations of a conference paper we present only the results for partially deceptive trap functions with order 10, 15 and 20 misleading schemata. These show the typical behaviour as deception is increased.

The chromosome length *l* is 50. The population size is 1000. The results are shown in Section 6.2. For order 10, $r = a = 0.21$ and z=30. For order 15, $r = a = 0.35$ and z=30.

6. Results

For all results GA performance per evaluation is presented in tables. As mentioned earlier the GA only re-evaluates individuals if they underwent crossover or mutation.

6.1 Results for L-SAT Problems

Table 1. shows the number of evaluations (our performance measure) taken by selective, two-point and uniform crossover to find the best solution. Recall from section 5.1 that there is no guarantee such an assignment exists. The solution quality is given in square brackets and represents the average solution found in 50 runs.

	Selective	**Two-point**	**Uniform**
Low	24098 (6430) [0.997]	29206 (1064) [0.999]	29621 (5782) [0.999]
Medium	28214 (509) [0.937]	37188 (69) [0.932]	38306 (42) [0.931]
High	27573 (484) [0.916]	37275 (61) [0.911]	38310 (43) [0.911]

Table 1. Mean number of evaluations to find the best solution for low, medium and high epistasis. The standard deviation is shown in brackets. The solution quality is given in square brackets. These are averages over 50 runs.

During low epistasis all crossover operators work equally as well (selective crossover is slightly better with the least number of evaluations, Table 1). As epistasis is increased to medium and high, selective crossover finds a better solution and does this with the least number of evaluations at 28214 and 27573 respectively. The standard

deviation is large with low epistasis because the GA stopped when it found a solution of 1.0 and recorded the number of evaluations. If the GA did not always find a solution of 1.0, evaluation was terminated at 600 generations.

Tests for significance were carried out, to compare the three crossover operators, on the number of evaluations yielded. Using the *t-test* with a 5% significance level, selective crossover showed a significant difference, an improvement of up to 25%. Selective crossover works well with epistatic problems because it allows exploitation of genes which are not necessarily situated together in the chromosome and it does this consistently with the least number of evaluations, thus saving online computation. These striking results led us to try selective crossover with deceptive functions, to see how it compared with the other conventional reproductive operators.

6.2 Results for Deceptive Trap Functions

Table 2 shows the mean number of evaluations taken to find the solution for order 10 and 15 trap functions. Our initial expectation was that selective crossover was too exploitative (exploration is achieved solely through the generation of Child 2 and, of course, mutation) and would always be misled by the local optimum in all experiments. In Table 2, the number of evaluations taken to reach the solution show that selective crossover took the least number of evaluations on average at 10856, but this is not a significant amount compared with uniform crossover. In an order 15 trap function selective crossover again took the least number of evaluations as well as finding a solution in early generations. Uniform crossover did not always find the solution: misleading schemata of order 15 deceived it. Selective crossover, like two-point and uniform crossover, was finally fooled by deception on an order 20 trap function.

	Selective	**Two-point**	**Uniform**
Order 10	10856 (2058) [1.0]	12735 (1372) [1.0]	11313 (1059) [1.0]
Order 15	15033 (2909) [1.0]	15168 (3286) [1.0]	15833 (5317) [0.93]

Table 2. Mean number of evaluations completed for order 10, 15 deceptive trap functions. The standard deviation is shown in brackets. The solution quality is given in square brackets. These are averages over 50 runs. Order 20 trap functions successfully deceived all three forms of crossover.

Tests for significance were carried out at the 5% level and as suspected selective crossover did not show a significant difference on the mean number of evaluations for deceptive trap functions. We can safely conclude that selective crossover did not perform poorly but did as well as two-point crossover.

7. Conclusions

We have described a new adaptive crossover operator, *selective crossover*, for use with genetic search. Its design was motivated by intuition abstracted from Dawkin's theory of natural evolution to exploit and express good characteristics.

Selective crossover uses an extra real-valued vector to bias and promote, onto the next generation, genes that have increased an individual's fitness in previous generations. It uses this vector as a means of storing knowledge about what happened in previous generations. It allows exploitation of good schemata regardless of their defining length; hence if a schema consists of interacting genes at the two extremes of the chromosome, it can be propagated as easily as a schema with interacting genes located adjacent to each other.

Experiments indicate that selective crossover performs better than, or as good as, a traditional GA, which uses two-point or uniform crossover, on a set of test problems that contain characteristics common in practical problems. The problem sets used were the L-Sat problem generator and the deceptive trap function that allowed epistasis and deception to be varied respectively. The results show that selective crossover worked exceptionally well with problems of high epistasis where it found a better solution, than the conventional operators, and did this with the least number of evaluations (~25% performance increase) which is significant at the 5% level.

Due to the generational exploitation that drives selective crossover we assumed that its performance would be poor when applied to deceptive problems. The fitness increase exploits genes and thus can easily converge at the local optimum. In the case of the trap functions, selective crossover had performance comparable with two-point and uniform crossover. Like two-point and uniform crossover, selective crossover was fooled by deception in trap functions of order 20 and above.

We conclude that the initial results of this study indicate that selective crossover may be a good candidate for a crossover operator in which practitioners can have more confidence to use as a starting point for an adaptive GA system.

8. Future Work

Currently the dominance values are increased proportionately to the fitness increase of a child. Since all genes that were changed undergo this increase we do not know which gene influenced the increase in fitness. To overcome this problem of promoting false changes the dominance values of each changed gene can be updated by sharing the fitness increase amongst these changed genes. For example, when there is an increase in fitness as a result of changing many genes, the dominance increase for each gene would be less in comparison to the fitness increase due to the change of just one gene. Further experiments will be carried out to compare selective crossover with other adaptive crossovers [WhOp94], to see how alternative ways of initialising the dominance values affects performance, and to investigate other forms of epistasis. Other experiments will include non-normalised fitness functions (i.e. Royal Road functions) and problems with a finite cardinality alphabet.

We also intend to analyse schema creation, propagation and disruption to determine exactly why and how a GA benefits from selective crossover and use those findings towards more reliable and adaptive recombination operators. Specifically, we wish to develop a recombination operator that identifies and exploits epistasis.

Acknowledgements

Our thanks to David Goldberg for his valuable comments and William Spears for the original version of GAC, on which our code is based.

References

[Ackl87] Ackley, D. H. (1987) *A connectionist machine for genetic hillclimbing*. Boston, MA:Kluwer Academic Publishers.

[Baker87] Baker, J. E. (1987) Reducing Bias and Inefficiency in the Selection Algorithm. In J.J Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, 14-21. Lawrence Erlbaum Associates.

[EsCaSc89] Eshelman, L. J., Caruana, R. A. & Schaffer J. D. (1989) Biases in the Crossover Landscape. In David Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, 10-19. Morgan Kauffman

[Davi89] Davis, L. (1989) Adapting operator probabilities in genetic algorithms. In David Schaffer (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, 61-69. Morgan Kauffman

[Dawk89] Dawkins, R. (1989) *The Selfish Gene - New Ed*. Oxford University Press, UK.

[DeGo93] Deb K. & Goldberg D. E. (1993). Analyzing Deception in Trap Functions. In L. D. Whitley, (ed.), *Foundations of Genetic Algorithms 2*, 93-108. CA: Morgan Kauffman.

[DePoSp97] De Jong, Kenneth A., Potter, Mitchell A. & Spears, William M. (1997) Using Problem Generators to Explore the Effects of Epistasis. In Thomas Bäck (ed.), *Proceedings of the 7th International Conference on Genetic Algorithms*, 338-345. Morgan Kauffman

[Gold89] Goldberg, D. E. (1989) *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley.

[Holl75] Holland. J. H. (1975) *Adaptation in Natural and Artificial Systems*. MIT Press.

[ScMo87] Schaffer, J. & Morishima, A. (1987) An adaptive crossover distribution mechanism for genetic algorithms. In J.J Grefenstette, (ed.), *Proceedings of the 2nd International Conference on Genetic Algorithms*, 36-40. Lawrence Erlbaum Associates.

[Spea97] Spears, W. M. (1997), Recombination Parameters. In T. Baeck, D. Fogel and Z. Michalewicz (ed.), *The Handbook of Evolutionary Computation*, Oxford University Press.

[Sysw89] Syswerda, W. (1989) Uniform Crossover in Genetic Algorithms. In J. David Schaffer, (ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, 10-19. Morgan Kauffman.

[WhOp94] White, T. & Oppacher, F. (1994) Adaptive Crossover Using Automata. In Y. Davidor, H.-P Schwefel and R. Männer (eds.), *Proceedings of the Parallel Problem Solving from Nature Conference*, 229-238. NY:Springer Verlag.

[WoMa95] Wolpert, D. H. & Macready, W. G. (1995) No free lunch theorems for search. Technical Report 95-02-010, Santa Fe Institute.