Crossover, Macromutation, and Population-based Search

Terry Jones
Santa Fe Institute
1399 Hyde Park Road
Santa Fe, NM 87501, USA
terry@santafe.edu

Abstract

A major reason for the maintenance of a population in a Genetic Algorithm (GA) is the hope of increased performance via direct communication of information between individuals. This communication is achieved through the use of a crossover operator. If crossover is not a useful method for this exchange, the GA may not, on average, perform any better than a variety of simpler algorithms that are not population-based. A simple method for testing the usefulness of crossover for a particular problem instance is presented. This allows the identification of situations in which crossover is apparently useful but is actually only producing gains that could be obtained, or exceeded, with macromutation and no population.

1 INTRODUCTION

The features of a GA that distinguish it most from other search methods are its use of a population and a crossover operator. In a GA, crossover is employed to effect direct communication between individuals in a population. The usefulness of crossover, and therefore of maintaining a population, has been a contentious issue in research on GAs (Fogel & Atmar, 1990; Eshelman & Schaffer, 1993; Spears, 1993; Fogel, 1994; Fogel & Stayton, 1994; Culberson, 1995).

There are many levels at which one might attempt to assess the usefulness of crossover. This paper examines crossover in the context of single problem instances. More general and ambitious aims, such as determining the usefulness of crossover 1) on classes of problem instances, 2) on classes of problems, or even 3) in general, are not considered. This paper provides a simple test for determining the efficacy of crossover *given a single problem instance and choices of GA parameters*. In what follows, discussion of the usefulness of crossover is always meant in this context.

In the standard test for determining whether crossover is useful to a GA in a particular situation, a GA with crossover is compared to a GA without crossover. If the GA with crossover outperforms the GA without crossover, this is taken as evidence that crossover is useful in this situation (i.e., that it is facilitating the exchange of building blocks between individuals). Therefore, the maintenance of a population and the use of crossover seem justified because these are clearly an aid to the algorithm. This paper argues that this conclusion is not justified. A distinction is made between the *idea* of crossover and the *mechanics* by which crossover operators attempt to implement this idea. It is shown that even in the absence of the idea, the mechanics alone can be effectively used for search. A GA with crossover disabled has neither the idea nor the mechanics of crossover. If it is outperformed by a normal GA, it does not follow that the idea of crossover is therefore useful to the normal GA.

A *crossover hillclimbing* algorithm is presented that illustrates the power of the mechanics of crossover. An investigation into the performance of this algorithm suggests a simple new test that indicates whether the GA is making gains through crossover above those made by simpler algorithms that do not use crossover or a population. This test involves a comparison between a normal GA and a GA that uses *random crossover*. Using this test, it is possible to see what advantage the GA is gaining from the idea of crossover over and above what could be gained from the mechanics alone. In some cases, when well-defined building blocks are not present, the GA may actually perform worse with normal crossover than a GA with random crossover, apparently because of its use of a population. The test gives an indication of the existence of building blocks that are exploitable by a GA with a given crossover operator.

2 CROSSOVER HILLCLIMBING

This section considers the combination of the navigation strategy of a simple hillclimber with the landscape structure of crossover. The resulting crossover hillclimbing (CH) algorithm (Jones, 1995b) is very competitive with both the standard GA and the hillclimber, from whose pieces it is assembled. An investigation into the reason for this leads to a clearer understanding of the dual role of crossover in a GA.

The crossover landscapes described in (Jones, 1995b) consist of a family of hypercubes, each of whose vertices corresponds to a pair of individuals.[1] To conduct a hillclimb within a hypercube, randomly choose a starting pair (A_0, B_0). Perform crossover on this pair repeatedly until some limit (*max-attempts*) is reached or a pair (A_1, B_1) is found with $\max(f(A_1), f(B_1)) \geq \max(f(A_0), f(B_0))$, where f is the fitness function. If no such pair is found, halt the climb. Otherwise, continue, performing a similar search from (A_1, B_1). This process carries out a single climb within a hypercube of the landscape. However, once *max-attempts* crossovers have been tried and a new vertex containing a better individual has not been found, there is no need to discard both A_i and B_i to continue the search. Instead, the CH algorithm discards the less fit of A_i and B_i (ties are broken arbitrarily) and replaces it with a randomly chosen individual. This has the effect of (probably) transferring the search into another hypercube, from which the hillclimb may continue in identical fashion. This process is illustrated in Figure 1.

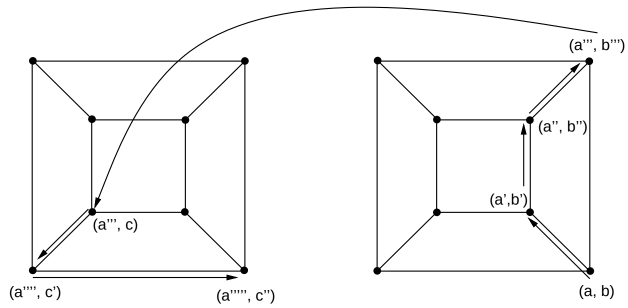

Figure 1: A fragment of a hillclimb under crossover. Three uphill steps are taken (using crossover) within the right hypercube, from (a, b) to (a''', b'''). At that point b''' (here assumed to be less fit than a'''), is discarded and replaced with the randomly generated c. Two more steps are then taken on the left hypercube. The jump into the left hypercube is either the result of a limit on the number of steps in a hypercube (*max-steps*) or a limit on the number of attempts to find an improving crossover (*max-attempts*).

A second parameter, *max-jumps* limits the number of times this jumping between hypercubes may occur. A third parameter, *max-steps* can be used to limit the number of steps that may be taken consecutively within a hypercube before a jump is forced. Equivalently, the number of pairs in the sequence of steps $(A_0, B_0), (A_1, B_1), \ldots$ within a hypercube is limited to *max-steps* + 1. A triple representing the values of *max-attempts*, *max-steps* and *max-jumps* will be used to describe particular CH algorithms. For example, CH(10,3,1000) represents the CH algorithm that examines up to 10 neighbors (under crossover), limits the number of steps that may be consecutively taken within a hypercube to 3, and makes at most 1000 jumps via discarding the worst individual of the current pair when no acceptable crossover is found or when 3 steps have been taken within a hypercube.

CH combines a simple hillclimbing navigation strategy with the landscape graph of crossover. The hillclimbing strategy is modeled on the random mutation hillclimbing algorithm used by Forrest and Mitchell (1993) and the crossover operator used is two-point. Other combinations are clearly possible, for instance a steepest ascent hillclimber with uniform crossover. Additionally, the requirement that

$$\max(f(A_{i+1}), f(B_{i+1})) \geq \max(f(A_i), f(B_i))$$

could be changed to

$$\max(f(A_{i+1}), f(B_{i+1})) > \max(f(A_i), f(B_i)).$$

Many other navigation strategies, such as A* and simulated annealing, could also be used on the crossover landscape graph.

2.1 EXPERIMENTS

CH(10,3,1000), which will be denoted simply as CH for the remainder of this paper, was compared to a bit-flipping hillclimber (BH), to a standard GA (GA-S) and to a standard GA with elitism (GA-E). In BH, one-bit mutants of the current individual are evaluated until an individual with equal or greater fitness is encountered and this becomes the new current individual. A maximum of $2l$ one-bit mutants are tried, where l is the number of neighbors.[2] If a climb does not result in an increase in fitness in 10,000 steps, it is terminated. This prevents BH from wandering indefinitely in a set of individuals of equal fitness none of which have a fitter neighbor. The GAs had population size 500 and were allowed to run 100 generations. If every locus of the population became more than 85% converged, the run was stopped. Mutation probability was 0.01, two-point crossover was performed with probability 0.75 and binary tournament selection was used with the fitter individual selected with probability 0.75. In the elitist GA, at least one copy of the best individual of a generation is guaranteed to survive into the subsequent generation.

The four algorithms run on three instances of six problems. These were: one max (Ackley, 1987) (strings of

[1] For crossover operators that act on and produce two individuals.

[2] No provision is made to avoid neighbors that have already been examined.

Table 1: Algorithm abbreviations and brief descriptions. The last column gives the section in which a fuller description can be found.

Abbreviation	Description	Section
GA-S	Standard GA.	§2.1
GA-E	Elitist GA.	§2.1
GA-RC	GA with random crossover.	§4
BH	Bit-flipping Hillclimbing.	§2.1
CH(a,s,j)	Crossover Hillclimbing, a attempts, s steps, j jumps.	§2
CH	CH(10,3,1000).	§2
CH-1S	CH(10,1,3000).	§2.3
CH-NJ	CH(10,3000,0).	§2.3
BH-MM	Hillclimbing with macromutations.	§5
BH-DMM	Hillclimbing with distributed macromutations.	§5

length 30, 60, 120), consecutive copies of a fully easy 6-bit problem (Deb & Goldberg, 1992) (5, 10 and 15 copies of the 6-bit problem), consecutive copies of a fully deceptive 6-bit problem (Deb & Goldberg, 1992) (5, 10 and 15 copies of the 6-bit problem), maximally distributed copies of the same 6-bit deceptive problem (5, 10 and 15 copies), 2-, 3- and 4-state busy beaver problems (Rado, 1962; Jones & Rawlins, 1993) and Holland's royal road problem (Holland, 1993; Jones, 1995a) (default parameter settings with k set to 2, 4 and 6, giving problems with 60, 240 and 960 bits).

2.2 SUMMARY OF CH RESULTS

Each of the four algorithms was run on three instances of six problems for at least 2000 runs. CH was very competitive with the other algorithms on these problems. It was certainly the most consistent algorithm. In no experiment did it have the worst performance. The hillclimber was clearly the worst algorithm on four of the functions (fully deceptive, distributed fully deceptive, fully easy and Holland's royal road), and the two GA variants were clearly the worst algorithms on the remaining two. On three of the functions (fully deceptive, fully easy and Holland's royal road), the CH algorithm was the best performer and was equal best on a fourth (busy beavers). Only on one problem of the six, one max, was it significantly beaten. The addition of elitism to the GA resulted in improvement in every case (though some were minimal). However, in no case was the performance difference large enough that the performance of one of the other algorithms (CH or BH) fell between the two GAs as a result.

Table 2 gives a rough overview of the performance of the algorithms. Each algorithm is ranked for each problem, with a rank of 1 being the best. Algorithms are give equal rank if it was unclear which was better. The two versions of the GA have been coalesced since they always ranked next to each other.

Table 2: Overall trends for the four algorithms on the six problems. Algorithms are roughly ranked according to performance. 1 = best, 3 = worst. The standard and elitist versions of the GA are represented as a single column.

	BH	CH	GA
Busy beavers	1	1	3
One max	1	2	3
Fully deceptive	3	1	2
Distributed deceptive	3	1	1
Fully easy	3	1	2
Holland's royal road	3	1	2

2.3 WHY DOES CH PERFORM SO WELL?

A first impression of these results is that the CH algorithm demonstrates that crossover is a powerful operator, at times more so than mutation. The algorithm was designed to isolate crossover and examine it in as simple a context as possible. To a certain extent this has been achieved, but a closer look shows there is more to the story. CH is creating new individuals in two ways: through the use of crossover, and at random. Initially, two random individuals are created and thereafter a random individual is created each time the algorithm jumps to a new hypercube. It is simple to examine the number of times a random individual is fitter than any individual seen so far. When this is done, the results are unsurprising. Random creation produces very few highly fit individuals.

To get an idea of the importance of the random individuals as a source of material for crossover to exploit, the CH algorithm used in the experiments above, CH(10,3,1000), was compared to CH(10,3000,0) and CH(10,1,3000). The two new algorithms represent extremes of the spectrum with respect to the random creation of new individuals. CH(10,3000,0) never jumps to a new hypercube when no acceptable crossover can be found from the current pair. Each search will therefore take place entirely within the hypercube determined by the initial choice of random individuals. When an improving (or equaling) crossover cannot be found from the current pair, the search is concluded. This algorithm will be denoted by CH-NJ (NJ = No Jumps). At the other extreme, CH(10,1,3000) allows at most 1 step to be taken by a pair within a hypercube before a jump to another hypercube is forced. This algorithm will be denoted by CH-1S, (1S = 1 Step). These parameter settings limit all three of these algorithms to 3000 steps via crossover.

Figures 2 to 7 show the results on the largest instance of each of the problems (with the exception of Holland's royal road, which shows the default problem). In these graphs the X axis, which indicates the mean number of evaluations taken, is shown on a log (base 10) scale. The Y axis indicates various levels of performance that could be achieved on the way to finding the global optimum. For example, a point at (5000, 3) on a graph of a problem containing 10 deceptive subproblems indicates that the algorithm used a mean of 5000 function evaluations to complete 3 of the 10 deceptive subproblems. On some problems, some algorithms never achieved the higher levels of performance. For example, in Figure 4, CH-NJ never completed more than 5 of the 15 deceptive subproblems. For this reason, there are no points on the graph indicating how long CH-NJ took to achieve these levels.

These graphs show that CH-1S is the best of the three algorithms on these instances of these problems. Experiments with other settings of the three parameters of the crossover hillclimbing algorithm (not shown) support the conclusion that the algorithm performs better the more often it jumps between hypercubes. CH-1S is at the extreme, jumping after every crossover. This is remarkable because the first crossover performed after a jump always involves a random individual. It is clear that the CH-1S algorithm is therefore simply performing macromutations that are distributed on the genome in the fashion of two-point crossover.

3 CROSSOVER: THE IDEA AND THE MECHANICS

It is useful to consider crossover as playing two roles. These can be called the *idea* and the *mechanics* of crossover. The idea of crossover is the hope that building blocks from two individuals may be combined into an offspring whose fitness exceeds either parent. The mechanics of crossover is the process by which an attempt is made to implement this idea. All forms of crossover share a similar idea, but the mechanics vary considerably (particularly for representations that are not fixed length strings).

The experiments with various forms of crossover hillclimbing indicate that crossover may be very useful even in the absence of the idea of crossover. That is, even when there is no reason to believe that both parents have an above average chance of contributing above average material to the offspring, crossover may still be useful simply through performing macromutation via its mechanics. When considering whether crossover is useful to a GA, we should attempt to distinguish the gains the algorithm is making using the idea of crossover from those made simply through the mechanics. If there is no additional gain due to the idea of crossover, we might do as well by simply employing macromutation and discarding the population.

4 THE HEADLESS CHICKEN

This section presents a simple experiment designed to test whether a GA is making progress through of the idea of crossover above that which could be made by performing macromutations. To do this, compare the GA with an identical GA that uses a *random crossover* operator. Given two parents, the operator generates two random individuals and uses these in crossovers with the parents. The action of this operator is illustrated in Figure 8. The use of random crossover is the only difference between the two algorithms.

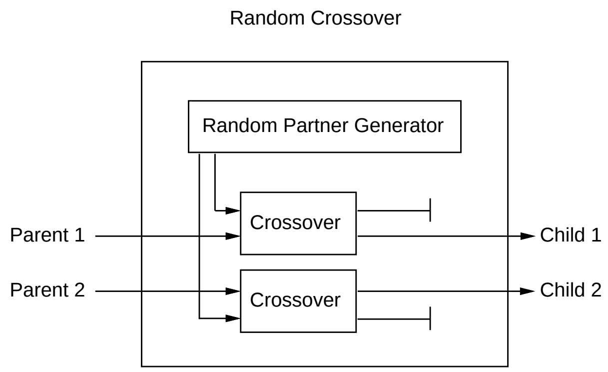

Figure 8: The random crossover operator used in the headless chicken test. Instead of recombining the parents as a normal crossover would do, two random individuals are created and crossed with them. One offspring from each of these crosses is returned. The crossover method used (one-point, two-point, uniform, etc.), is the one used in the GA that is being compared to the GA with random crossover.

In the algorithm with random crossover, there is no direct communication between population members as there is no crossover between population members.[3] When one of the individuals involved in a crossover is randomly chosen, the operation is purely mechanical and has nothing of the spirit of crossover. In this

[3]Unless, by chance, a randomly generated parent also happens to be present in the population.

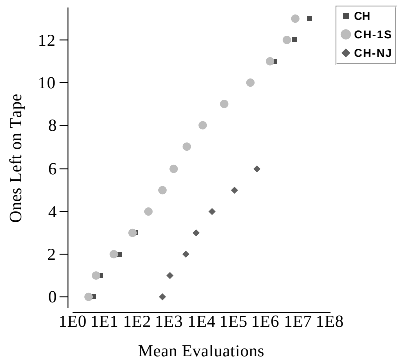

Figure 2: Mean evaluations to discover 4-state Turing machines writing different numbers of ones on the tape and then halting.

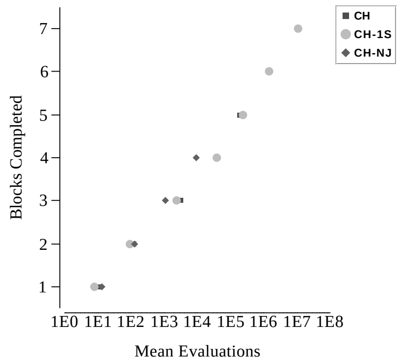

Figure 3: Mean evaluations to complete 15 6-bit maximally distributed fully deceptive subproblems.

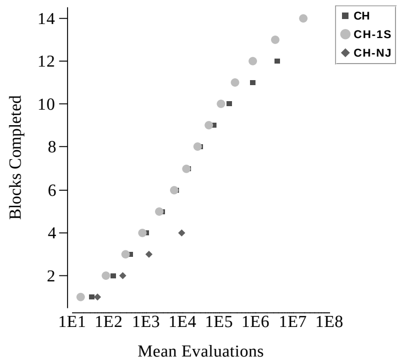

Figure 4: Mean evaluations to complete 15 contiguous 6-bit fully deceptive subproblems.

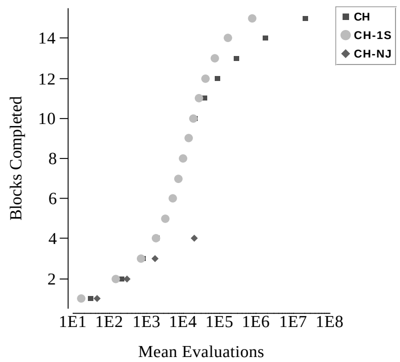

Figure 5: Mean evaluations to complete 15 contiguous 6-bit fully easy subproblems.

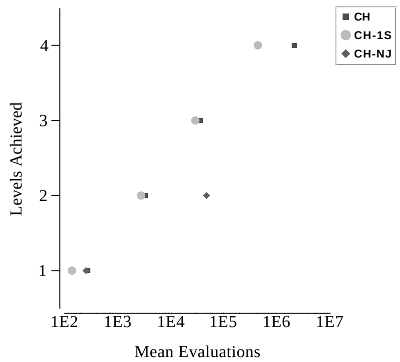

Figure 6: Mean evaluations to reach levels in Holland's royal road with default parameter settings (string length 240 bits).

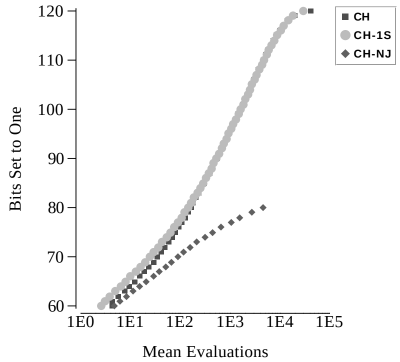

Figure 7: Mean evaluations to find ones in a one max problem of 120 bits.

case, despite the identical mechanical rearrangement, the operation is not crossover. That is, without the idea of crossover, you do not have crossover. To call this operation crossover is similar to calling a headless chicken running around the yard a chicken. Certainly it has many characteristics of a chicken, but an important piece is missing. A further argument that this operation is not really crossover is that two parents are not required. For example, for two-point crossover on strings, simply choose the crossover points and set all the loci between the points to randomly chosen alleles. This is an identical operation and is clearly just a macromutation.

This comparison was performed (1) between the GA used in this paper and an identical GA with random crossover (GA-RC) and (2) between the elitist GA of this paper and an elitist GA with random crossover. Figures 9 to 14 show the result of comparison between GA-S and GA-RC on the six problem instances shown in the previous sets of graphs. The comparisons between GA-E and GA-RCE (not shown) exhibit identical qualitative results. On all instances of all problems, GA-S is a clear winner over GA-RC when the problem contains well-defined building blocks. This is the case with the fully deceptive, fully easy, Holland's royal road and one max problems. On these problems, the idea of crossover is clearly aiding the search.

More interesting is the comparison on the problems where well-defined building blocks do not exist: the busy beaver and distributed fully deceptive problems. On the busy beaver problems, the performance of GA-S and GA-RC is hard to distinguish. On the distributed fully deceptive problem, GA-RC outperforms GA-S. In this case, GA-S is not only using an operator (two-point crossover) that, in an informal sense, has a bias that is orthogonal to the structure of the encoding, but is also drawing samples from a population. GA-RC uses the same operator but is able to explore more widely. When GA-RC is as good or better than the normal GA on a problem, it suggests that

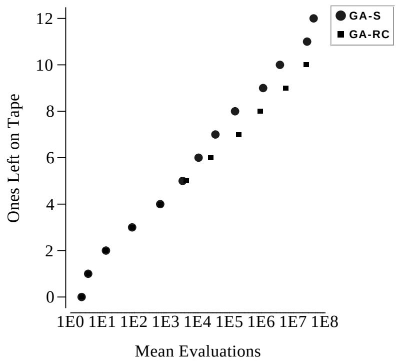

Figure 9: Mean evaluations to discover 4-state Turing machines writing different numbers of ones on the tape and then halting.

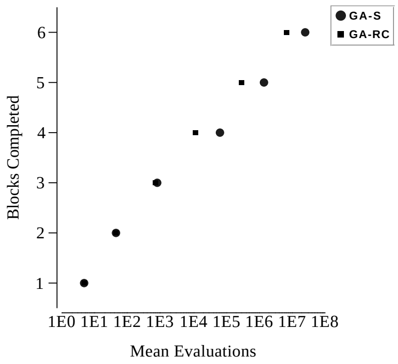

Figure 10: Mean evaluations to complete 15 6-bit maximally distributed fully deceptive subproblems.

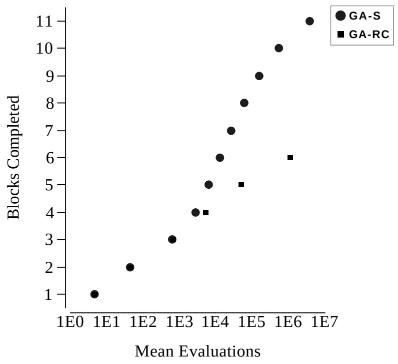

Figure 11: Mean evaluations to complete 15 contiguous 6-bit fully deceptive subproblems.

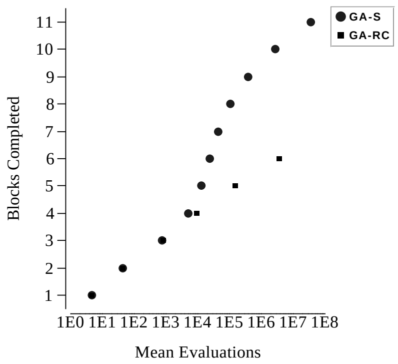

Figure 12: Mean evaluations to complete 15 contiguous 6-bit fully easy subproblems.

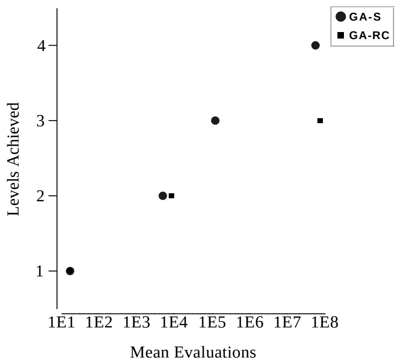

Figure 13: Mean evaluations to reach levels in Holland's royal road with default parameter settings (string length 240 bits).

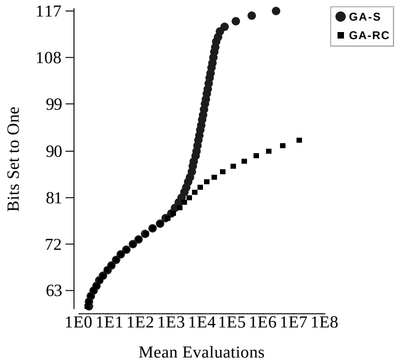

Figure 14: Mean evaluations to find ones in a one max problem of 120 bits.

the idea of crossover is doing nothing significant for the normal GA. In such cases it seems safe to draw two conclusions:

- The chosen combination of representation, fitness function, crossover operator and other GA parameters does not result in a problem that contains building blocks that the GA is managing to exploit using the idea of crossover.
- Equally good performance could be obtained from a GA that used normal mutation and macromutation (with mutated loci chosen so as to be distributed in the way the crossover operator does it). This is not really a GA, as there is no direct communication between population members.

Notice that these conclusions, even if correct, do not imply that there is no representation with exploitable building blocks, or that another crossover operator might not do better at exploiting the building blocks that do exist (if any), or that it is not possible to modify the GA in some way so that it does exploit the building blocks (e.g., by using a larger population, a different fitness function, or adopting measures to maintain diversity). These are conclusions about a single crossover operator, a single representation and a single algorithm, and may well not apply if any of these are altered.

The experiment suggests an explanation for GAs that appear to make good use of crossover early in a run, but not thereafter. It may be that the combination of the representation and fitness function does not have building blocks that can be exploited by the crossover operator in question. In this case, crossover may make improvements simply through macromutations at the start of the run and then, as is widely known, become of increasingly less worth as the population converges. Crossover *appears* to be useful (since disabling it results in worse performance), but in fact, its usefulness may never be more than a consequence of the macromutations it is performing.

5 MACROMUTATIONAL HILLCLIMBING

The success of CH-1S prompted the implementation of two hillclimbers that use a macromutational operator and the navigation strategy of BH. In the mutation operator of the first, (BH-MM), the genome is viewed as a ring. Two distinct points on the ring are selected uniformly at random and the loci in the smaller section are set to random alleles (including those that are already present). The smaller section is mutated as observation revealed that CH made the great majority of its improving steps when it modified less than half the genome. The action of this mutation operator is exactly what a crossover with a random individual would produce if the child inheriting most from the non-random parent was retained. The operator distributes the mutations in the same way that two-point crossover does. This hillclimber is very similar to CH(1,1,∞). In the second hillclimber (BH-DMM), the macromutation operator picks a number of loci to set randomly (as in BH-MM) but distributes these mutations at random through the string.

These hillclimbers were compared to CH-1S and GA-E. Except for the busy beaver problems, one of the macromutational hillclimbers always proved the best algorithm. Once again, the results depended on whether well-defined building blocks existed. When they did, BH-MM outperformed BH-DMM by orders of magnitude. When they did not (on the distributed fully deceptive problem), BH-DMM performed an order of magnitude better than BH-MM. On every problem, BH-MM achieved the highest levels of performance at least one order of magnitude faster than GA-E, and in one case (15 fully easy subproblems) required over 3 orders of magnitude less evaluations to achieve the highest level reached by the GA. Typically, the hillclimbers also reached levels of performance never achieved by the GA. The results for the six problems instances are shown in Figures 15 to 20.

6 SUMMARY

The traditional method used to assess whether crossover is useful to a GA in a given situation is to compare the GA with crossover to a GA without crossover. This paper argues that if the GA with crossover performs better, the conclusion that crossover is therefore useful (i.e., that it is facilitating the exchange of building blocks between individuals) is not justified. A more informative method of assessing the worth of crossover to a GA in a given situation is presented. This compares the normal GA with a GA that uses random crossover. Such a GA dispenses with the idea of crossover while maintaining the mechanics. This makes it possible to see what advantage the standard GA is gaining from the idea of crossover over and above what it could be gaining from the mechanics alone. In some cases, when well-defined building blocks are not present, the GA may actually perform *worse* with normal crossover than a GA with random crossover, apparently as a result of its use of a population. This test gives an indication of the existence of building blocks in a problem instance that are exploitable by a GA, given choices of GA parameters.

A major reason a GA maintains a population is to allow the communication of information between individuals via crossover. If the above comparison indicates that the idea of crossover is not producing gains, the use of a GA, in this configuration, is probably not warranted. In the observed cases, there are simpler algorithms that use the macromutational mechanics of crossover and do not maintain a population, and these easily outperform the GA. Several such algorithms, based on crossover hillclimbing or macromutational hillclimbing were presented. Even when building blocks do exist, these algorithms (especially those that use macromutation exclusively) outperformed the GA on virtually every instance of every problem addressed. These results are in accordance with Eshelman and Schaffer's belief that the "niche" of problems in which crossover gives a GA a competitive advantage may be quite small (1993). Actually, the situation appears worse than they feared: a macromutational hillclimber easily outperforms a standard GA on Holland's royal road, which has the properties that Eshelman and Schaffer ascribe to problems residing in crossover's niche. The niche becomes smaller if we insist that the idea of crossover, in addition to the mechanics, be responsible for good performance.

Acknowledgments

Thanks to Stephanie Forrest, Ron Hightower and especially Derek Smith for many helpful conversations about this work, and to David Fogel for comments on an early draft. This research was supported in part by grants to the Santa Fe Institute, including core funding from the John D. and Catherine T. MacArthur Foundations, the National Science Foundation (PHY-9021427), and the U.S. Department of Energy (DE-FG05-88ER25054).

References

Ackley, D. H. 1987. *A Connectionist Machine for Genetic Hillclimbing.* Boston, MA: Kluwer Academic Publishers.

Culberson, J. C. 1995. Mutation-Crossover Isomorphisms and the Construction of Discriminating Functions. *Evolutionary Computation*, **2**(3), 279–311.

Deb, K., & Goldberg, D. E. 1992. *Sufficient Conditions for Deceptive and Easy Binary Functions.* Tech. rept. University of Illinois, Urbana-Champaign. IlliGAL Report No 92001. Available via ftp from gal4.ge.uiuc.edu in pub/papers/IlliGALs/92001.ps.Z.

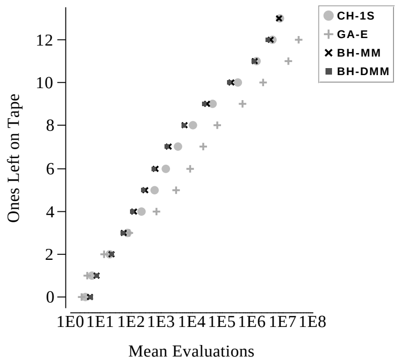

Figure 15: Mean evaluations to discover 4-state Turing machines writing different numbers of ones on the tape and then halting.

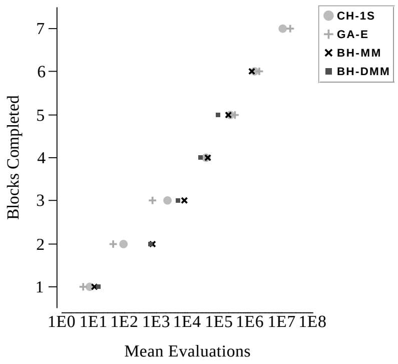

Figure 16: Mean evaluations to complete 15 6-bit maximally distributed fully deceptive subproblems.

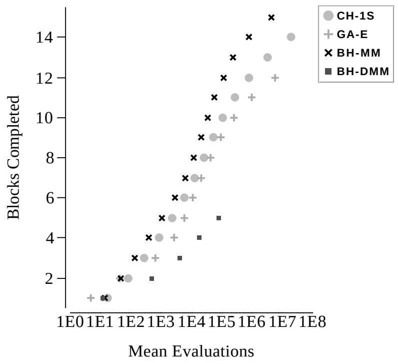

Figure 17: Mean evaluations to complete 15 contiguous 6-bit fully deceptive subproblems.

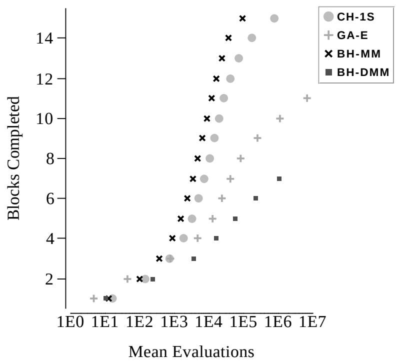

Figure 18: Mean evaluations to complete 15 contiguous 6-bit fully easy subproblems.

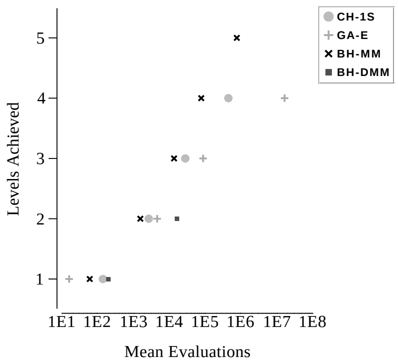

Figure 19: Mean evaluations to reach levels in Holland's royal road with default parameter settings (string length 240 bits).

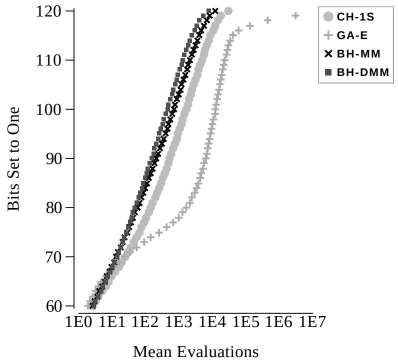

Figure 20: Mean evaluations to find ones in a one max problem of 120 bits.

Eshelman, L. J., & Schaffer, J. D. 1993. Crossover's Niche. *Pages 9–14 of:* Forrest, S. (ed), *Genetic Algorithms: Proceedings of the Fifth International Conference (ICGA 1993)*. San Mateo, CA: Morgan Kaufmann.

Fogel, D. B. 1994. Applying Evolutionary Programming to Selected Control Problems. *Computers Math. Applic.*, **27**(11), 89–104.

Fogel, D. B., & Atmar, J. W. 1990. Comparing Genetic Operators with Gaussian Mutations in Simulated Evolutionary Processes Using Linear Systems. *Biological Cybernetics*, **63**, 111–114.

Fogel, D. B., & Stayton, L. C. 1994. On the Effectiveness of Crossover in Simulated Evolutionary Optimization. *BioSystems*, **32**, 171–182.

Forrest, S., & Mitchell, M. 1993. Towards a Stronger Building-Blocks Hypothesis: Effects of Relative Building-Block Fitness on GA Performance. *Pages 109–126 of: Foundations of Genetic Algorithms*, vol. 2. Vail, Colorado: Morgan Kaufmann.

Holland, J. H. 1993 (Aug 12). *Royal Road Functions*. Internet Genetic Algorithms Digest v7n22. Available via ftp from ftp.santafe.edu in pub/terry/jhrr.tar.gz.

Jones, T. C. 1995a. A Description of Holland's Royal Road Function. *Evolutionary Computation*, **2**(4), 411–417.

Jones, T. C. 1995b (March). *Evolutionary Algorithms, Fitness Landscapes and Search*. Ph.D. thesis, University of New Mexico, Albuquerque, NM.

Jones, T. C., & Rawlins, G. J. E. 1993. Reverse Hillclimbing, Genetic Algorithms and the Busy Beaver Problem. *Pages 70–75 of:* Forrest, S. (ed), *Genetic Algorithms: Proceedings of the Fifth International Conference (ICGA 1993)*. San Mateo, CA: Morgan Kaufmann.

Rado, T. 1962. On Non-computable Functions. *Bell System Technical Journal*, **41**, 877–884.

Spears, W. M. 1993. Crossover or Mutation? *Pages 221–237 of:* Whitley, L. D. (ed), *Foundations of Genetic Algorithms*, vol. 2. San Mateo, CA: Morgan Kaufmann.