
Pareto Optimality, GA-easiness and Deception

Sushil J. Louis
Department of Computer Science
Indiana University
Bloomington, IN 47405
louis@cs.indiana.edu

Gregory J. E. Rawlins
Department of Computer Science
Indiana University
Bloomington, IN 47405
rawlins@cs.indiana.edu

Abstract

This paper defines a class of spaces which are easy for genetic algorithms and hard for stochastic hill-climbers. These spaces require genetic recombination for successful search and are partially deceptive. Problems where tradeoffs need to be made subsume spaces with these properties. Preliminary results comparing a genetic algorithm without crossover against one with two-point crossover support these claims. Further we show how a genetic algorithm using pareto optimality for selection, outperforms both. These results provide insight into the kind of spaces where recombination is necessary suggesting further study of properties of such spaces, and what it means to be GA-easy and hill-climbing hard.

1 INTRODUCTION

Recombination plays a central role in genetic algorithms (GAs) and constitutes one of the major differences between GAs and other blind search algorithms.¹ The answer to the question of whether a search space is particularly suited to a genetic algorithm therefore hinges on the power of recombination in generating an appropriate search bias. We define one class of spaces that requires genetic recombination for successful search. This class includes problems where a particular kind of tradeoff exists and is exemplified by many design problems in engineering and architecture.

Recent work on GA-easy and “royal-road” functions suggests that it is not easy to find spaces where GAs emerge a clear winner when compared with a stochastic hill-climber or a population of stochastic hill-climbers (SHC) [Wilson, 1991, Mitchell and Forrest, 1993]. The work suggests that

¹A good definition and introduction to blind search algorithms can be found in [Rawlins, 1991].

it is not enough that optimal lower order schemas combine to form optimal higher order schemas. Hill-climbers often solve such problems in a fewer number of function evaluations than a genetic algorithm. Instead we look to work on deception to provide insight into the kind of spaces that are *deceptive* for SHCs or a population of SHCs and that are relatively easy for genetic algorithms [Goldberg, 1989a]. Identifying properties of such spaces should provide insight into the inner workings of a GA and aid in the practical selection of optimization methods.

The next section defines the model of a genetic algorithm and a stochastic hill-climber used in this paper. We then design a problem that is partially deceptive and needs recombination to overcome deception. Spaces with structure similar to the problem defined in section three may be found in multicriteria optimization, thus Pareto optimality is defined in section four and used in selecting candidates for recombination in a genetic algorithm. Empirical evidence in section five compares performance on the problem defined in section three. Finally, the last section summarizes and discusses the results pointing out directions for further work.

2 GAS, HILL-CLIMBERS AND RECOMBINATION

A genetic algorithm works with a population and encodes a problem’s parameters in a binary string. The initial population is formed by a randomly generated set of strings. Our introductory model of a classical genetic algorithm assumes proportional selection, 2-point crossover and a mutation operator that complements a bit at a randomly selected position. For experimental purposes our population of stochastic hill-climbers is a genetic algorithm without crossover. In this paper hill-climber and stochastic hill-climber are used interchangeably and problems are cast as maximization problems in Hamming space, eliding the issue of encoding.

A genetic algorithm without crossover and only mutation can be likened to an *Evolution Strategy* [Bäck et al., 1991], albeit a simple one with a nonvarying mutation rate and random bias. It is a stochastic hill-climber and depending on the rate of mutation, can make jumps of varying sizes through the search space. Being population based and stochastic it makes fewer assumptions about the search space and as such is more robust and harder to deceive than the bit-setting optimizable and steepest-ascent optimal algorithms defined in Wilson’s paper [Wilson, 1991].

Although there is debate in natural genetics as to the evolutionary viability of recombination, for our purposes it suffices to note that it exists. In genetic algorithms, Holland’s schema theorem emphasizes the importance of recombination and provides insight into the mechanics of a GA [Holland, 1975]. With the genetic algorithm and hill-climbers as defined above, recombination is the key difference between the two. It allows *large, structured* jumps in the search space. In other words, it facilitates combination of low order building blocks to form higher order building blocks, if they exist. The type of combinations facilitated, depends on the crossover operator and population distribution. In a random population of strings (encoded points) large jumps in the search space are akin to jumps that are possible with a high temperature in simulated annealing. However the size of the jumps due to recombination in a GA depends on the hamming average or the degree of convergence of the population. As the population converges, the usual and corresponding decrease in hamming average of the population leads to a decrease in maximum jump size and exploration of a correspondingly smaller neighborhood. But this kind of neighborhood exploration can be carried out by a GA with mutation only; a stochastic hill-climber.

During the early stages of GA processing, large jumps, combinations of building blocks that are hamming dissimilar, are more probable since the hamming distance between possible mates is higher. The GA should search effectively in spaces where such combinations of building blocks lead toward higher order optimal schemas. But at this early stage, not only is the variance in estimated fitnesses of schemas high, but there may be enough diversity in a population of hill-climbers to stumble on the correct higher order schema. Both the order of early building blocks and diversity of the population need to be considered in analyzing recombination.

The discussion above implies that for recombination to be effective and hill-climbing ineffective, a GA needs to be able to make large, *useful*, structured jumps through recombination even after partial convergence; low probability jumps for a hill-climber at this stage.

3 A HILL-CLIMBING DECEPTIVE PROBLEM

Consider the function shown in figure 1. For this

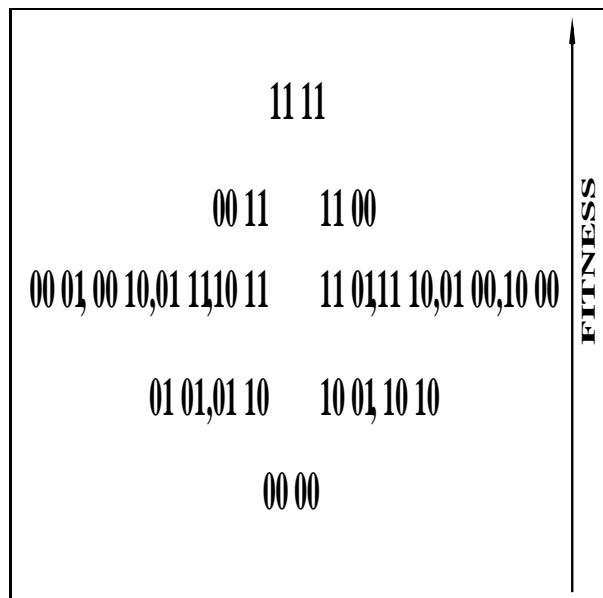


Figure 1: A hill-climbing deceptive function that should be easy for GAs

four-bit function, the global optimum is at 1111, but there are two hill-climbing deceptive optima at 0011 and 1100, the global optimum is 2 bits away from the suboptima. For an l length string, the function can be stated as follows: Let x_1 be the number of ones on the right half of the string ($l/2$ bits) and let x_2 be the number of ones on the left half of the string then,

$$f(\text{string} \neq 1111\dots) = |2 \times x_1 - x_2| + |2 \times x_2 - x_1|$$

$$f(1111\dots) = x_1 \times x_2 \tag{1}$$

Intuitively, counting from one side (left or right), fitness increases as the number of 1’s increases until the halfway point of the string, after this point the fitness decreases as the number of 1’s increases. Figure 2 gives an alternate view of the same function.

For sufficiently large l this type of function is deceptive for a SHC or PSHC since for a string of length l , once a stochastic hill-climber reaches a deceptive optima, it will have to simultaneously set all the remaining $l/2$ bits to 1 to attain the global optima. Setting fewer than $l/2$ bits decreases fitness. The probability of simultaneously setting $l/2$ bits to 1 decreases exponentially with l , thus making it hard on a SHC or PSHC for sufficiently large l . Using our model of a stochastic hill-climber with mutation probability p_m , after reaching the suboptima, the probability of setting $l/2$ bits is: the probability that $l/2$ incorrect bits are flipped,

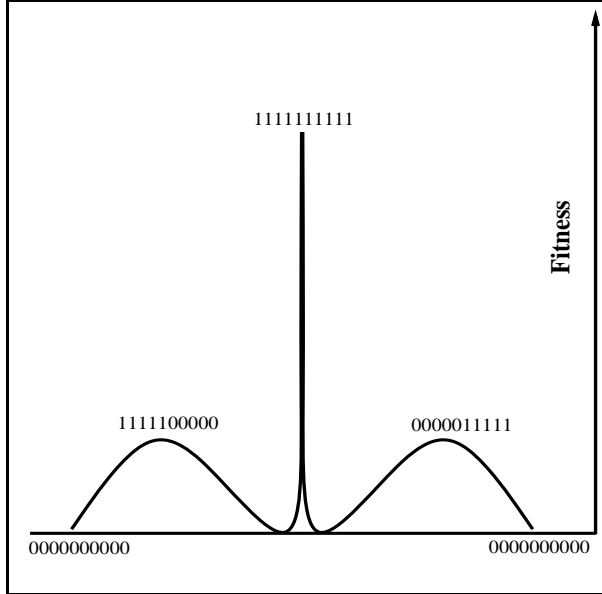


Figure 2: An alternate view of a hill-climbing deceptive function that should be easy for GAs (10 bits)

multiplied by the probability that the correctly set bits are not flipped

$$(p_m)^{l/2} \times (1 - p_m)^{l/2}$$

Now consider the power of recombination. Once a genetic algorithm reaches the hill-climbing deceptive optima, crossover can produce the global optima in one operation, *assuming* that the population contains representatives of both suboptima. Niching operators can be used to provide representatives at the suboptima as long as these operators don't entail mating restrictions which may then not allow the algorithm to find the global optimum.

Next, consider whether this type of space may be found in normal application domains. In many areas of engineering a problem is usually stated in terms of multiobjective (or multicriteria) optimization, and the function above can be easily and naturally stated as a two-objective optimization problem. We suggest that multicriteria optimization problems may provide fertile ground for finding search spaces in which genetic algorithms do better than other blind search algorithms. However, as with niching, multicriteria optimization raises a point. The issue for our algorithms is that, in general, it is difficult to combine multiple measures into a single fitness in a reasonable way without making assumptions about the relationships among the criteria. Unfortunately, a single fitness measure is the usual feedback for genetic algorithms and for the niching methods in [Deb and Goldberg, 1989]. For example consider a beam section problem, where the objectives are to maximize the moment of inertia and minimize the perimeter. Minimizing the perimeter

usually decreases the moment of inertia and maximizing the moment of inertia increases the perimeter; a tradeoff. Combining the two criteria into a single measure makes no sense² unless the relationship between the two is already known. Fortunately, the concept of *pareto optimality* allows us to do this reasonably and, as will be seen, provides a ready-made niching mechanism, killing two birds with one stone.

4 PARETO OPTIMALITY AND SELECTION

Pareto optimality operates on the principle of non-dominance of solutions. An n criteria solution s_1 with values (c_1, c_2, \dots, c_n) dominates s_2 with values (d_1, d_2, \dots, d_n) if

$$\forall_i (c_i \geq d_i) \text{ AND } \exists_i (c_i > d_i)$$

Considering the maximal value along each criteria to be one of our hill-climbing deceptive optima, we have a correspondence between the function defined in section 2 and multicriteria optimization problems. If a central peak (global optimum) exists as in figure 2 then recombination should find it, if both suboptima are represented in the population. Incorporating pareto optimality into the selection mechanism of a genetic algorithm facilitates the maintenance of stable subpopulations representing the suboptima.

To incorporate pareto optimality in the genetic algorithm, we use a variant of binary tournament selection. The algorithm selects two individuals at random from the current population, mates them, and produces the pareto optimal set of the parents and offspring. Two random individuals from this pareto optimal set form part of the next population. The procedure repeats until the new population fills up, thus becoming the subsequent current population. This method with a tournament size of 2 is computationally inexpensive since only 4 solutions compete at a time.

The pareto optimal genetic algorithm is used to attack the problem in section 2. First, the problem is split into two, corresponding to the two additive terms in equation 1. Cast as a 2 criteria optimization problem, the first criterion is

$$c_1(\text{string} \neq 1111\dots) = |2 \times x_1 - x_2|$$

$$c_1(1111\dots) = x_1 \times x_2$$

where x_1 stands for the number of 1's in one half of the string and x_2 the number of 1's in the other half of the string. The second criterion c_2 then becomes

$$c_2(\text{string} \neq 1111\dots) = |2 \times x_2 - x_1|$$

$$c_2(1111\dots) = x_2 \times x_1$$

²Comparing apples to oranges

5 RESULTS

I compare the number of times the optimum is found by a GA without crossover, a classical GA with two-point crossover (CGA) and a pareto optimal GA with two-point crossover. The population size in all experiments was 30 and the results were averaged over 11 runs of the algorithm with different random seeds. The GA with mutation ran with mutation probability from 0.01 to 0.10 in increments of 0.01 and the best results used. The probability of crossover was 0.8 and the mutation rate 0.02 for the other two algorithms. To make the function a little easier for all algorithms, it was modified slightly and the peak in figure 2 broadened at the base. If the length of the string is l , x_1 is the number of ones in one half of the string and x_2 the number of ones in the second half of the string, the modified function for the pareto GA can be stated as

$$\begin{aligned} &\text{if} && (x_1 + x_2) \geq (l - p) \\ &\text{then} && c_1(\text{string}) = x_1 \times x_2 \\ &\text{else} && c_1(\text{string}) = |2 \times x_1 - x_2| \end{aligned} \quad (2)$$

for c_1 , the first criterion. Here p is a parameter denoting the width of the base of the peak in figure 2. Similarly for c_2

$$\begin{aligned} &\text{if} && (x_1 + x_2) \geq (l - p) \\ &\text{then} && c_2(\text{string}) = x_1 \times x_2 \\ &\text{else} && c_2(\text{string}) = |2 \times x_2 - x_1| \end{aligned} \quad (3)$$

For the GA with two-point crossover and the GA without crossover, the function becomes

$$\begin{aligned} &\text{if} && (x_1 + x_2) \geq (l - p) \\ &\text{then} && \text{fitness}(\text{string}) = (x_1 \times x_2)^2 \\ &\text{else} && \text{fitness}(\text{string}) = |2 \times x_2 - x_1| + |2 \times x_1 - x_2| \end{aligned}$$

The squaring of the values near the peak tries to ensure that once found by a classical genetic algorithm, the peak will not be lost easily. The pareto GA is not affected by differences in magnitude but only the relative ordering.

Figures 3 and 4 compare the maximum fitness over time for the three algorithms. The algorithms ran for 200 generations or 6000 function evaluations. The string length was 30, the parameter p in the definition of the function was set to 2 to produce figure 3 and to 4 to produce figure 4. The significant difference in performance of the pareto optimal GA provides strong support for the importance of recombination and the niching effect of pareto optimal selection. In addition,

as expected, the width of the peak significantly effects performance with wider peaks allowing the GA with two-point crossover to perform significantly better than the stochastic hill-climber. The plot for the pareto optimal GA shows the average fitness for only one of the two criteria, hence the low (about 1/2 the value of the other two plots) initial values.

Performance with peak width = 2

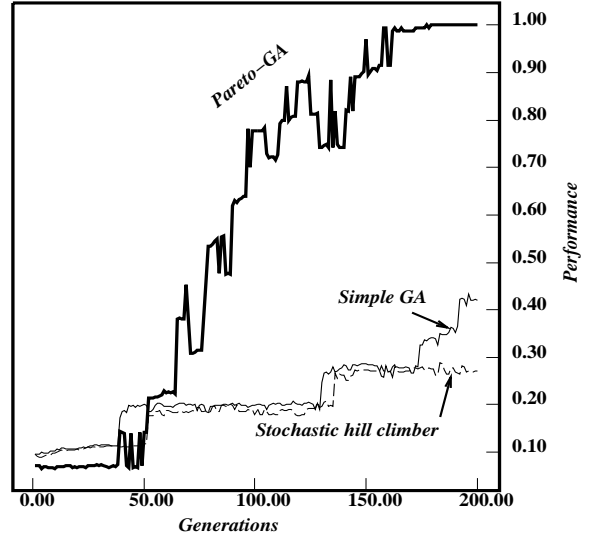


Figure 3: Performance comparison for string length 30. The peak width is 2.

Performance with peak width = 4

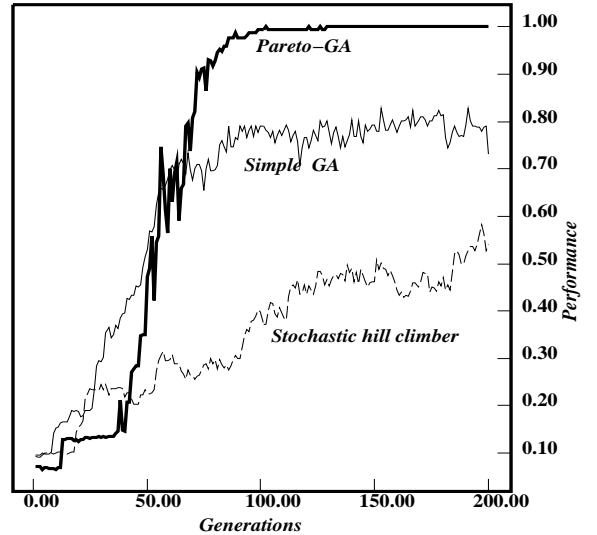


Figure 4: Performance comparison for string length 30 but with peak width 4.

Table 1 compares the percentage of times the algorithms found the optima for strings of length 20, 30,

Table 1: Comparing the number of times the optimum was found in 200 generations for narrow peaks.

Length	Width	PGA	CGA	PSHC
20	1	100 %	55 %	45 %
30	2	100 %	27 %	18 %
40	3	73 %	0 %	0 %
50	4	45 %	0%	0%

Table 2: Comparing the number of times the optimum was found in 200 generations for broader peaks.

Length	Width	PGA	CGA	PSHC
20	2	100 %	91 %	100 %
30	4	100 %	82 %	55 %
40	6	100 %	64 %	0 %
50	8	82 %	0%	0%

40 and 50 for peaks of size 1, 2, 3 and 4. In the table, peak size is given in column 2, PGA is the pareto GA while CGA is the GA with two-point crossover and PSHC is the GA with no crossover – a population of stochastic hill-climbers. Table 2 depicts the same information but for wider peaks.

These results strongly indicate that a pareto GA is more likely to find the optimum on problems of this type, followed by a GA with crossover. The stochastic hill-climber is least likely to find the optimum and in this case had less then a 50% chance, even on the smaller 20 length problem when the peak was narrow. Wider peaks help the hill-climber far small problems (length 20, peak width 2) but as the problem size increases the hill-climber’s performance deteriorates to become worse than even a simple GA. In all cases the Pareto GA does as well or better than the other two algorithms.

6 DISCUSSION, CONCLUSIONS AND FUTURE WORK

Design problems are often formulated as multiobjective or multicriteria optimization problems. In many cases such problems involve tradeoffs among possibly conflicting criteria. Spaces where recombination of two or more single criterion optima leads toward a global optimum, with a deceptive basin in between to trap hill-climbers, seem well suited for genetic algorithms especially when using use pareto optimality in their selection process. Pareto optimal selection appears to provide the necessary niches until recombination produces the global optimum. We defined a problem in hamming space with these properties and empirically compared the performance of a classical GA with two-point crossover to that of a stochastic hill-climber (A GA without crossover), and a GA with pareto optimal selection. The GA with pareto optimal selection always found the optimum more often than the others, while the classical GA usually did better than the stochastic hill-climber. Although preliminary, the evidence suggests a closer look at the spaces defined in this paper for function encoding combinations that are relatively easy for genetic algorithms and hard for hill-climbings methods.

Pareto optimal selection also eliminates the need to combine disparate criteria into a single fitness as is usual in genetic algorithms. The method described in this paper is distributable and computationally less expensive compared to other suggested methods for incorporating pareto optimality[Goldberg, 1989b]. Exploring niche forming with pareto optimality selection remains an interesting area for further research.

Ongoing work seeks a more rigorous analysis of the properties of the spaces described in this paper. Using a simple function that counts the number of ones makes preliminary work easier but may hide more subtle complexities that underly and complicate search in such domains. Finally, this paper only considers two criteria optimization; as the number of criteria rises, the number of possible combinations rises combinatorially. How GAs work on such problems remains to be analyzed and their relative effectiveness in these spaces determined.

Acknowledgements

This work owes much to John S. Gero and Sourav Kundu of the Design Computing Unit, Department of Architectural and Design Science at the University of Sydney, Australia.

References

[Bäck et al., 1991] Bäck, T., Hoffmeister, F., and Schwefel, H. (1991). A survey of evolution strategies.

- In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 1–10. Morgan Kauffman, San Mateo, CA.
- [Deb and Goldberg, 1989] Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. Morgan Kauffman, San Mateo, CA.
- [Goldberg, 1989a] Goldberg, D. E. (1989a). Genetic algorithms and walsh functions: Part 2, deception and its analysis. *Complex Systems*, 3:153–171.
- [Goldberg, 1989b] Goldberg, D. E. (1989b). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA.
- [Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI.
- [Mitchell and Forrest, 1993] Mitchell, M. and Forrest, S. (1993). Relative building-block fitness and the building-block hypothesis. In Whitley, L. D., editor, *Foundations of Genetic Algorithms - 2*, pages 109–126. Morgan Kauffman, San Mateo, CA.
- [Rawlins, 1991] Rawlins, G. J. E. (1991). Introduction. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms*, pages 1–10. Morgan Kauffman, San Mateo, CA.
- [Wilson, 1991] Wilson, S. W. (1991). Ga-easy does not imply steepest-ascent optimizable. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 85–89. Morgan Kauffman, San Mateo, CA.