Guided Crossover:
A New Operator for Genetic Algorithm Based Optimization

Khaled Rasheed
Computer Science Department
Rutgers University
New Brunswick, NJ 08903, USA
krasheed@cs.rutgers.edu

Abstract-

Genetic algorithms (GAs) have been extensively used in different domains as a means of doing global optimization in a simple yet reliable manner. They have a much better chance of getting to global optima than gradient-based methods which usually converge to local sub-optima. However, GAs have a tendency of getting only moderately close to the optima in a small number of iterations. To get very close to the optima, the GA needs a very large number of iterations, whereas gradient-based optimizers usually get very close to local optima in a relatively small number of iterations.

In this paper we describe a new crossover operator which is designed to endow the GA with gradient-like abilities without actually computing any gradients and without sacrificing global optimality. The operator works by using guidance from all members of the GA population to select a direction for exploration. Empirical results in several engineering design domains demonstrate that the operator can significantly improve the steady state error of the GA optimizer.

1 Introduction

Genetic Algorithms (GAs) [Goldberg 1989] are search algorithms that mimic the behavior of natural selection. GAs attempt to find the best solution to some problem (e.g., the maximum of a function) by generating a collection ("population") of potential solutions ("individuals") to the problem. Through mutation and recombination (crossover) operations, better solutions are hopefully generated out of the current set of potential solutions. This process continues until an acceptably good solution is found. GAs have many advantages over other search techniques, including the ability to deal with qualitatively different types of domains, such as continuous variable domains, discrete or quantized variable domains, or mixed-type variable domains. In continuous domains though, local search methods such as gradient-based algorithms tend to have an advantage over the GA, and that is their ability to get extremely close to local optima in a relatively small number of iterations. Most GAs need a large number of iterations to get very close to optima. On the other hand, GAs have a much higher probability of reaching the global optimum than local-based methods. One classical approach to solve this dilemma is to use a GA optimizer to get to the "good region" and then run a local search method to complete the optimization. This approach is indeed very good, but it has many problems. One problem is how to decide that the GA has reached the "good region". The good region may be thought of as the basin of attraction of the global optimum. In the case of highly multi-modal spaces, this region can be very small. In this case, we have to run the local search method many times starting, say, from selected points of the final GA population. This can get very expensive. Another problem is that local search methods suffer from a "curse of dimensionality".

This paper presents a new crossover operator, called "Guided Crossover" (GC), capable of improving the steady state error of a GA optimizer (the distance between the best point found and the global optimum of the search space) in a continuous space, without having to use gradient information or gradient-based methods.

We demonstrate the merit of our operator in GA optimization of realistic continuous-variable engineering design domains. In such domains (such as the two explored in Section 4) a design is represented by a number of continuous design parameters, so that potential solutions are vectors (points) in a multidimensional vector space. Determining the quality ("fitness") of each point involves the use of a simulator that computes relevant physical properties of the artifact and summarizes them into a single measure of merit. The simulator will often take a non-negligible amount of time, ranging from a fraction of a second to, in some cases, many days. Consequently, gradients are extremely expensive to compute. Moreover, the use of table lookup in simulations, and the high dimensionality and multi-modality of the search space make it very difficult to use gradient-based methods with a high degree of reliability. We also demonstrate the utility of guided crossover in a set of benchmark domains that have been used in several engineering design optimization studies [Powell and Skolnick 1993].

The remainder of this paper first presents a more detailed description of the new operator. We then present a number of experiments concerning the use of our operator on several realistic engineering design tasks. We conclude the paper with a discussion of related efforts and future work.

2 GA Architecture

GADO, the GA used in this research is described in detail in [Rasheed *et al.* 1997, Rasheed 1998, Rasheed and Hirsh 1999]. Each individual in the GA population represents a parametric description of an artifact, such as an aircraft, or a process, with each parameter taking on a value in some continuous interval. The fitness of each individual is based on the sum of a proper measure of merit computed by a simulator (such as the takeoff mass of an aircraft), and a penalty function if relevant (such as to impose legal limits on the permissible noise of an aircraft). Operators are applied to elements of the population via some selection scheme. Here selection was performed by rank (rather than via the actual value of the fitness function on each individual) because of the wide range of fitness values caused by the use of a penalty function — rank selection prevents the first discovered evaluable/feasible points from dominating the population. A steady state GA model is used, in which existing points in the population are replaced by newly generated points via some replacement strategy. The replacement strategy used here takes into consideration both the fitness and the proximity of the points in the GA population, with the goal of selecting for replacement a point that both has low fitness and is relatively close to the point being introduced. The GA stops when either the maximum number of evaluations has been exhausted or the population completely loses diversity and practically converges to a single point in the search space. The GA architecture also includes a screening module (SM) [Rasheed and Hirsh 1997, Rasheed and Hirsh 1999] and a diversity maintenance module (DMM) which can both be turned on or off. The SM saves time but preventing the GA from evaluating points that are close to previously encountered bad points. It uses a case-based learning technique to do this. The DMM maintains diversity by preventing the GA from evaluating candidate point which are extremely close to previously evaluated points. The DMM also attempts to restore diversity through a re-seeding operation if severe loss of diversity is detected in the early stages of the optimization.

The following section describes the new crossover operator in detail.

3 Guided Crossover

Guided Crossover (GC) works as follows:

1. One candidate point is selected from the GA population using the normal selection rule (by rank) and called $candidate_1$.

2. The second candidate point is also selected from the GA population but in a different way: for each point X in the GA population other than $candidate_1$ a quantity Mutual_fitness(X,$candidate_1$) is computed, where

$$Mutual\_fitness(A, B) = \frac{(fitness(A) - fitness(B))^2}{Euclidean\_distance(A, B)^2}$$

   A choice for X that maximizes Mutual_fitness(X,$candidate_1$) is taken to be $candidate_2$.[1]

3. $candidate_1$ and $candidate_2$ are swapped if necessary, to make $candidate_1$ the point that has the higher fitness among the two.

4. The result of the crossover is a point along the line joining $candidate_1$ to $candidate_2$ which is selected at random from the small region around $candidate_1$ (the better point) as follows:

$$Result = L * candidate_1 + (1 - L) * candidate_2$$

   where L is a uniformly distributed random number in the interval [1-0.2*x,1+x] and x is a function of the number of elapsed iterations I_e and the total allowed number of iterations I_t such that:

$$x = 0.75 * \frac{(I_t - I_e)}{I_t} + 0.25$$

In words, GC examines all the directions that can be formed by joining the randomly selected first candidate point to all other points in the current GA population. The directions are ranked based on the contribution they give to the objective function when moving between the two end points relative to the distance between the end points. The best direction according to this ranking is chosen, and a small step is taken in this direction in the vicinity of its best end point. The magnitude of the step decreases as the GA optimization progresses.

The guided crossover operator should not be used as the only crossover operator in a GA architecture, because it is greedy in nature. We propose using it as a substitute crossover operator only a fraction of the time. In the current implementation, a random choice is made in every iteration between GC and more conventional crossover operators. The probability of choosing GC is increased linearly from 0 to its maximum value as the number of iterations increases to its maximum allowed value. The maximum value is one of the external parameters. Its default value is 0.25.

The intuition behind guided crossover is that it endows the GA with a way to get very close to the optimum once it is already near it — an advantage usually claimed for gradient-based methods over GAs — without the costly computation of gradients using potentially expensive evaluations in high-dimension spaces. Instead, the quantity Mutual_fitness(A,B) used to rank directions serves as a crude form of gradient calculation and does not entail a single additional evaluation of fitness.

4 Experimental Results

To evaluate our GA operator we applied it to design problems in two modern engineering design domains: the conceptual design of supersonic transport aircraft, and the design of

[1] In the last 5% of the iterations, the point with the best fitness in the current GA population is taken to be $candidate_2$ (unless this is $candidate_1$, in which case the point with the second best fitness is taken to be $candidate_2$). This is an effort to force the best point at the end of the optimization to be locally optimum.

Table 1: Aircraft Parameters to Optimize

No.	*Parameter*
1	exhaust nozzle convergent length
2	exhaust nozzle divergent length
3	exhaust nozzle external length
4	exhaust nozzle radius(r7)
5	engine size
6	wing area
7	wing aspect ratio
8	fuselage taper length
9	effective structural t/c
10	wing sweep over design mach angle
11	wing taper ratio
12	Fuel Annulus Width

supersonic missile inlets, as well as several benchmark engineering design domains. It should be noted that the merit and competitiveness of the GADO optimization approach with respect to traditional optimization methods (including gradient-based methods) in these domains has been demonstrated in [Rasheed 1998]. In this research we focused on the effect of the proposed crossover operator on the GA performance, rather than the overall GA behavior.

To demonstrate the merit of guided crossover we ran experiments with three different setups:

- GADO with guided crossover (the default).
- GADO without guided crossover.
- GADO with guided crossover replaced by the Heuristic Crossover (HC) operator.

The heuristic crossover operator [Michalewicz 1996] is similar in principle to guided crossover. The heuristic crossover operator works as follows: let the parents be $\bar{X}$ and $\bar{Y}$ such that $\bar{Y}$ is not worse in fitness than $\bar{X}$. The newborn is $\bar{Y} + r \cdot (\bar{Y} - \bar{X})$ where r is a random value selected uniformly in the interval $[0, 1]$. In fact, heuristic crossover may be viewed as a simple (less knowledge-based, less adaptive) version of guided crossover.

This section discusses our results in the different domains.

4.1 Supersonic Transport Aircraft Design

Our first domain concerns the conceptual design of supersonic transport aircraft. We summarize it briefly here; it is described in more detail in [Gelsey *et al.* 1996]. Figure 1 shows a diagram of a typical airplane automatically designed by our software system. The GA attempts to find a good design for a particular mission by varying the aircraft conceptual design parameters in Table 1 over a continuous range of values.

The GA evaluates candidate designs using a multidisciplinary simulator. In our current implementation, the GA's goal is to minimize the takeoff mass of the aircraft, a measure of merit commonly used in the aircraft industry at the conceptual design stage. Takeoff mass is the sum of fuel mass, which

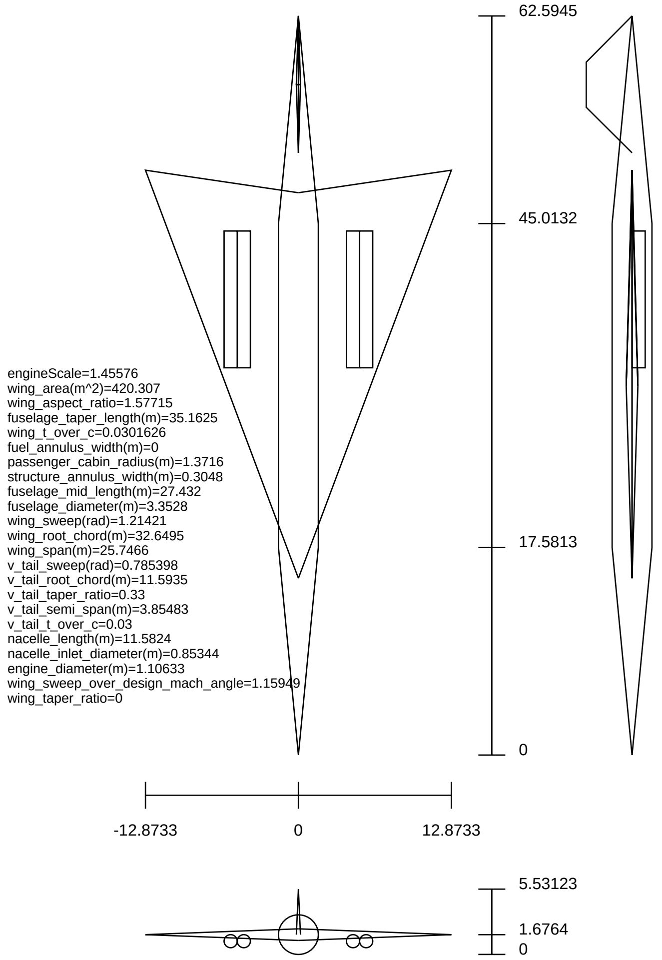

Figure 1: Supersonic transport aircraft designed by our system (dimensions in feet)

provides a rough approximation of the operating cost of the aircraft, and "dry" mass, which provides a rough approximation of the cost of building the aircraft. A complete mission simulation requires about 1/5 second of CPU time on a DEC Alpha 250 4/266 desktop workstation.

The aircraft simulation model used is based on both implicit and explicit assumptions and engineering approximations and since it is being used by a numerical optimizer rather than a human domain expert, some design parameter sets may correspond to aircrafts that violate these assumptions and therefore may not be physically realizable even though the simulator does not detect this fact. For this reason a set of constraints has been introduced to safeguard the optimization process against such violations. In particular, a penalty function approach was used to incorporate the effect of constraint violations into the optimization: the penalty function was added to the takeoff mass value returned by the simulator and the resulting sum was the quantity that the optimizer actually minimizes (which therefore also serves as the fitness value assigned to each point of the GA population). The specific penalty function was simply a large constant multiplied

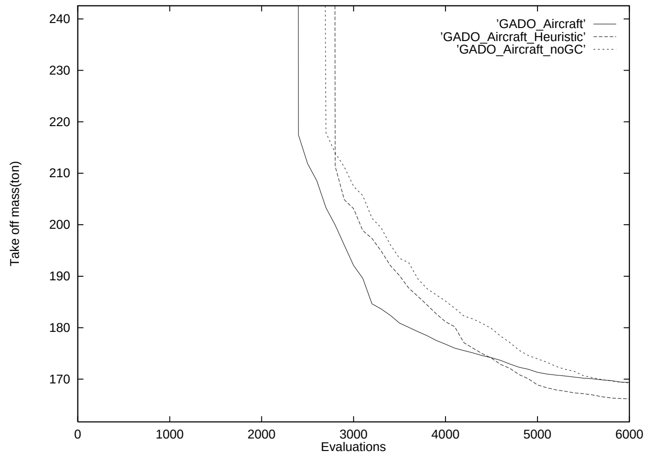

Figure 2: Effect of guided crossover in the Aircraft Design Domain

by the sum of the amounts of constraint violation for all the violated constraints.

The presence of constraints induced a partition of the search space into three mutually exclusive regions:

1. Unevaluable points: These are points that represent designs that violate the model assumptions so much that the simulator cannot complete the simulation process to produce any significant information. For such points a very large fictitious takeoff mass is generated as the value of the objective function.

2. Infeasible evaluable points: These are points that represent unrealizable aircrafts but the type and extent of model violation is moderate enough for the simulator to complete its work and report the constraint violation information. As described above, a penalty function is added to the takeoff mass returned by the simulator to account for the model violation.

3. Feasible points: The simulator succeeds in evaluating the take off mass for such points and no violations occur in the process. The penalty function for these points is zero.

Ten random populations of 60 points each were generated, and for each population the GA was allowed to proceed for 6000 iterations (an iteration denotes one call to the simulator, which takes, on average, 0.2 seconds) once with GC and once without it and a third time with heuristic crossover replacing GC. The results of our experiment are shown in Figure 2. The graph plots the measure of merit (takeoff mass) of the best point found so far against the number of iterations. Each curve represents the average of the runs from the ten starting populations. The solid curve represents the average of the ten runs of GADO with the default parameters (which includes guided crossover) one dotted curve represents the average of the runs with heuristic crossover and the other dotted curve represents the average of the runs with the parameters adjusted so that guided crossover is never done.

The figure indicates that although GC did not improve the steady state error in this domain (in fact HC had a better average final performance), it improved the intermediate performance of the GA. If the search were to be terminated, at say 3000 iterations, the performance of the runs which used GC would still be reasonably acceptable, but not without GC or with HC.

4.2 Supersonic Missile Inlet Design

Our second domain concerns the design of inlets for supersonic and hyper-sonic missiles. We summarize it briefly here; it is described in more detail in [Zha *et al.* 1996].

The missile inlet designed is an axisymmetric mixed compression inlet that cruises at Mach 4 at 60000 feet altitude. Minimizing manufacture cost for this inlet is critical, and therefore, techniques such as boundary layer bleed and variable geometry are not used — the performance of the inlet thus relies solely on the aerodynamic design of the rigid geometry, such as the extent of external and internal compression, contraction ratio, inlet start throat area, throat location, shock train length, and divergence of sub-sonic diffuser. The simulator used in this domain is a program called "NIDA" which was developed at United Technology Research Center (UTRC) as an inlet analysis/design tool [Haas *et al.* 1992]. The eight design parameters (all continuous valued) are given in Table 2, with coordinates given in terms of axial (x) and radial (r) positions. The objective of the optimization is to minimize the total pressure recovery, a quantity that is commonly used to measure the performance of inlets.

Table 2: Inlet Parameters to Optimize

Param	*Definition*
θ_1	initial cone angle
θ_2	final cone angle
x_d	axial location of throat
r_d	radial location of throat
x_e	axial location of end of "constant" cross section
θ_3	internal cowl lip angle
H_{ej}	height at end of constant cross section
H_{fk}	height at beginning of constant internal cross section

As in the aircraft domain, our experiments consist of GA runs with and without guided crossover and with heuristic crossover replacing guided crossover, starting from five random initial populations. In this domain each population consists of 20 random individuals. For each population the GA was allowed to proceed for 2000 iterations (an iteration denotes one call to the simulator, which, on average, takes six seconds).

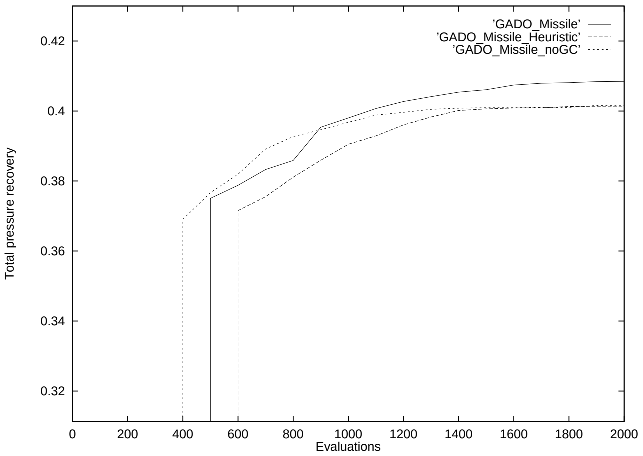

Figure 3: Effect of guided crossover in the Missile Inlet Design Domain

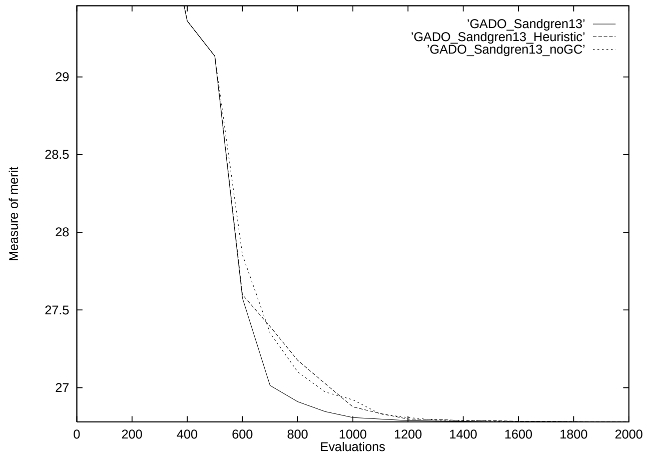

Figure 4: Effect of guided crossover in benchmark 1 (Sandgren's problem 13)

Table 3: Benchmark domains

Domain	Sandgren	*Dim.*	**Constraints**		best
No.	*No.*		*inequ.*	*equ.*	f
1	13	5	4	0	26.78
2	2	3	2	0	-3.3
3	3	5	6	0	-3.06
4	8	3	2	0	-5.68
5	6	6	0	4	8.92
6	15	16	0	8	244.8
7	21	13	13	0	97.5
8	22	16	19	0	174.7

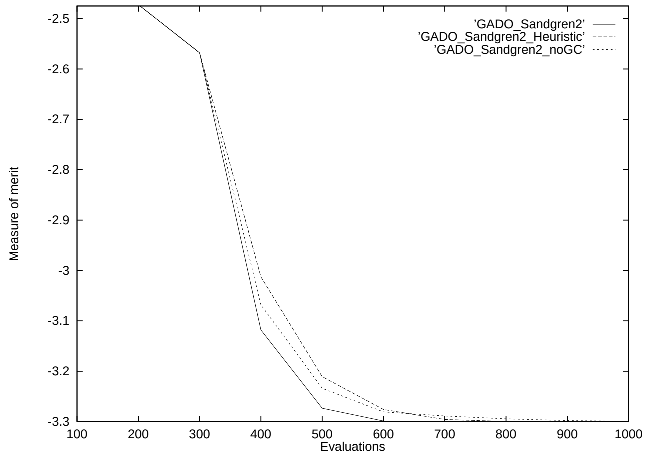

Figure 5: Effect of guided crossover in benchmark 2 (Sandgren's problem 2)

The results are shown in Figure 3, which plots the measure of merit (total pressure recovery) as a function of iteration number. The figure demonstrates that a significant amount of improvement in the later stages of the optimization was realized by using guided crossover.

4.3 Benchmark domains

These problems were first introduced by Eric Sandgren in his Ph.D. thesis [Sandgren 1977] and have since been used in engineering design optimization research as benchmarks. One of the recent experiments involving these domains was reported in [Powell and Skolnick 1993], in which a GA package called OOGA and a numerical optimization package called NumOpt were compared to each other in 10 of Sandgren's domains. The 10 domains were a representative sample of the original 30. We ran experiments in eight of these 10 domains.[2] All eight were minimization problems.

Some properties of the benchmarks used are summarized in Table 3. The second column of the table shows the problem numbers as they appeared in Sandgren's thesis. The third column shows the problem dimensions (i.e., the number of design variables in each problem). The fourth and fifth columns show the number of inequality and equality constraints respectively. The sixth column shows the best known optima of the problems. A detailed description of these domains is given in [Sandgren 1977]. A detailed description of the experiments comparing GADO to other optimizers in these domains can be found in [Rasheed 1998]. In this paper we only focus on studying the effect of guided crossover on performance.

Experiments were performed using five random starting populations for each domain. GADO was run once with GC and once with HC replacing GC and a third time without GC. The results are shown in Figure 5 to Figure 9. It is clear how GC enhanced the performance in these domains[3]. The final performance with GC was always the best. The intermedi-

[2]We were unable to do any comparison in 2 of the 10 domains because they had unbounded variables.

[3]We had to exclude benchmark 6 (Sandgren's problem 15) from our final analysis as several runs under all different setups did not find any feasible points. we were unable to do any qualitative comparisons.

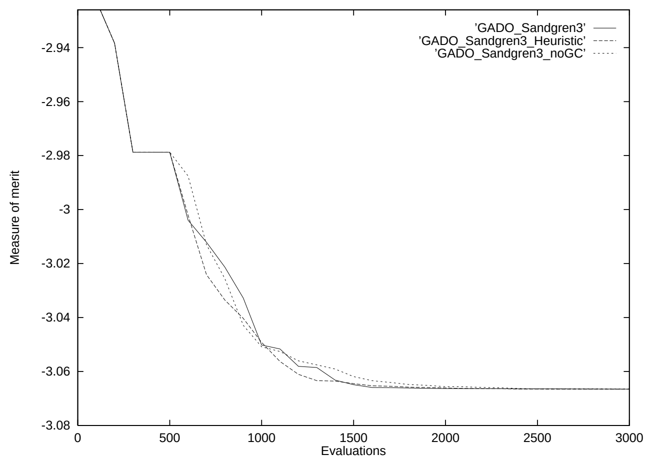

Figure 6: Effect of guided crossover in benchmark 3 (Sandgren's problem 3)

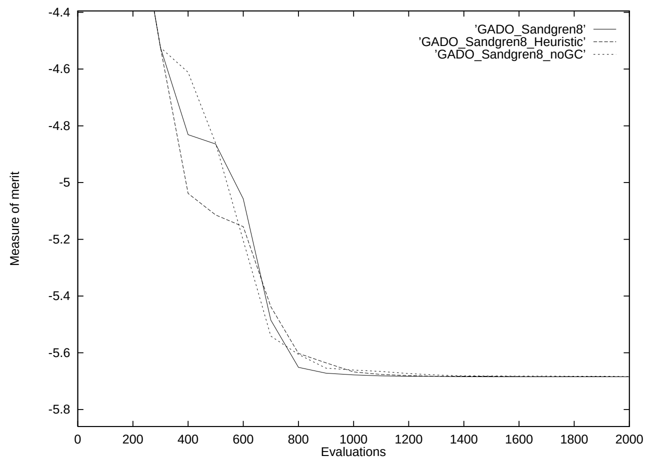

Figure 7: Effect of guided crossover in benchmark 4 (Sandgren's problem 8)

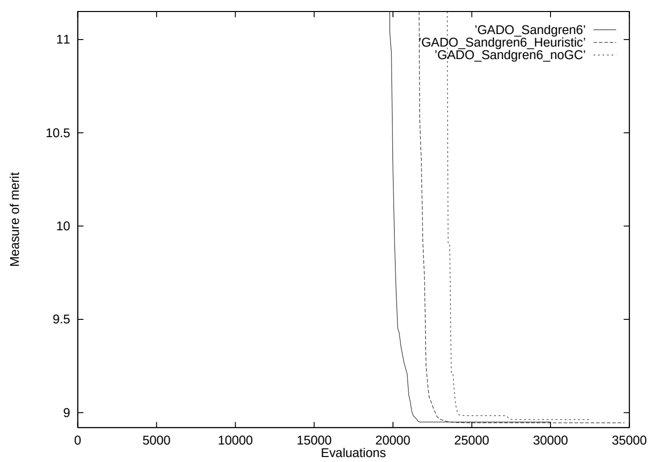

Figure 8: Effect of guided crossover in benchmark 5 (Sandgren's problem 6)

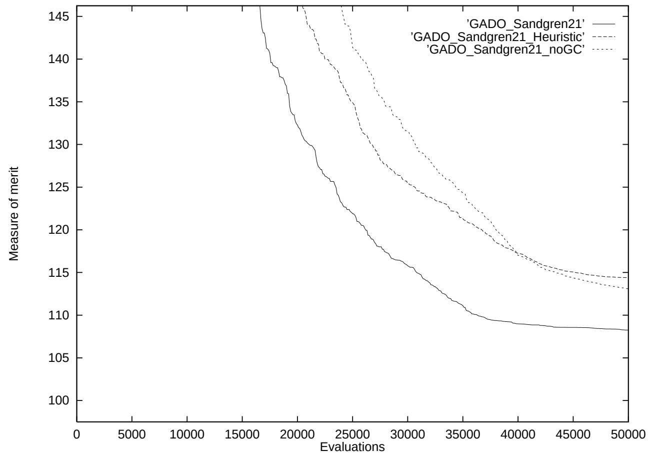

Figure 9: Effect of guided crossover in benchmark 7 (Sandgren's problem 21)

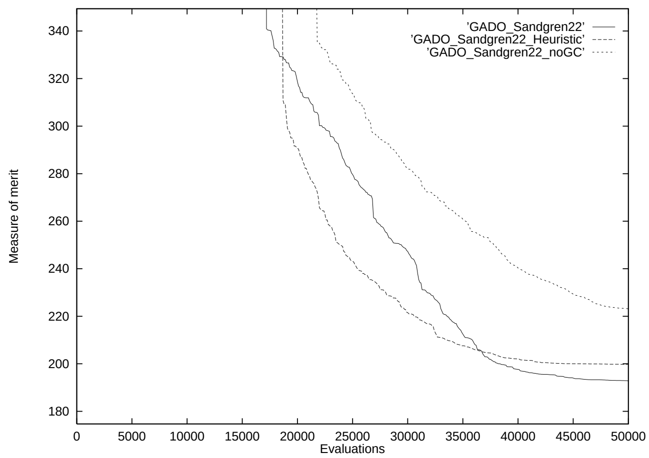

Figure 10: Effect of guided crossover in benchmark 8 (Sandgren's problem 22)

ate performance with GC was the best in all stages of the optimization in four out of the seven domains used in the final comparison.

5 Final Remarks

A new crossover operator for continuous space search has been presented. Experimental results demonstrated the merit of using the new operator in the domains of aircraft design, missile inlet design and several benchmark engineering design domains.

Several research efforts have applied genetic algorithms to engineering design optimization problems in a variety of domains, including control system design [Kundu and Kawata 1996], architectural and civil engineering design [Gero *et al.* 1997, Rosenman 1997], VLSI design [Lienig and Thulasiraman 1993], mechanical design [Chapman and Jakiela 1996, Deb 1997] and aircraft design [Obayashi *et al.* 1997]. Some of these efforts included innovative GA operators [Deb 1997].

The new crossover operator contributed a lot of improvement to the GA optimization. The idea of taking a little step in a promising direction, proved very powerful. In GC, the promising direction is selected by joining lines between points of the GA population and choosing the most promising line. However, a very interesting idea is to create new directions by combining several such lines. We plan on exploring this method in the near future.

Acknowledgments

I thank our aircraft design expert, Gene Bouchard of Lockheed, for his invaluable assistance in this research. I also thank all members of the HPCD project, especially Haym Hirsh, Donald Smith, and Keith Miyake. This research was partially supported by NASA under grant NAG2-817 and is also part of the Rutgers-based HPCD (Hypercomputing and Design) project supported by the Advanced Research Projects Agency of the Department of Defense through contract ARPA-DABT 63-93-C-0064.

Bibliography

[Chapman and Jakiela 1996] C. D. Chapman and M. J. Jakiela. Genetic algorithm-based structural topology design with compliance and topology simplification considerations. *Journal of Mechanical Design*, 118(1):89–98, 1996.

[Deb 1997] Kalyanmoy Deb. Geneas: A robust optimal design technique for mechanical component design. In *Evolutionary Algorithms in Engineering Applications*, pages 497–514. Springer-Verlag, 1997.

[Gelsey *et al.* 1996] Andrew Gelsey, M. Schwabacher, and Don Smith. Using modeling knowledge to guide design space search. In *Fourth International Conference on Artificial Intelligence in Design '96*, 1996.

[Gero *et al.* 1997] John S. Gero, Vladimir A. Kazakov, and Thorsten Schinier. Genetic engineering and design problems. In *Evolutionary Algorithms in Engineering Applications*, pages 47–68. Springer-Verlag, 1997.

[Goldberg 1989] David Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Mass., 1989.

[Haas *et al.* 1992] M. Haas, R. Elmquist, and D. Sobel. NAWC Inlet Design and Analysis (NIDA) Code, Final Report. UTRC Report R92-970037-1, 1992.

[Kundu and Kawata 1996] Sourav Kundu and Seiichi Kawata. AI in control system design using a new paradigm for design representation. In *Fourth International Conference on Artificial Intelligence in Design '96*, 1996.

[Lienig and Thulasiraman 1993] Jens Lienig and K. Thulasiraman. A genetic algorithm for channel routing in VLSI circuits. *Evolutionary Computation*, 1(4):293–311, 1993.

[Michalewicz 1996] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1996.

[Obayashi *et al.* 1997] S. Obayashi, Y. Yamaguchi, and T. Nakamura. Multiobjective genetic algorithm for multidisciplinary design of transonic wing platform. *Journal of Aircraft*, 34(5):690–693, 1997.

[Powell and Skolnick 1993] D. Powell and M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 424–431. Morgan Kaufmann, July 1993.

[Rasheed and Hirsh 1997] Khaled Rasheed and Haym Hirsh. Using case-based learning to improve genetic-algorithm-based design optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997.

[Rasheed and Hirsh 1999] Khaled Rasheed and Haym Hirsh. Learning to be selective in genetic-algorithm-based design optimization. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, 1999. To appear.

[Rasheed *et al.* 1997] Khaled Rasheed, Haym Hirsh, and Andrew Gelsey. A genetic algorithm for continuous design space search. *Artificial Intelligence in Engineering*, 11(3):295–305, 1997. Elsevier Science Ltd.

[Rasheed 1998] Khaled Rasheed. GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers University, New Brunswick, NJ, January 1998. Ph.D. Thesis, http://www.cs.rutgers.edu/$\sim$krasheed/thesis.ps.

[Rosenman 1997] M. A. Rosenman. The generation of form using an evolutionary approach. In *Evolutionary Algorithms in Engineering Applications*, pages 69–86. Springer-Verlag, 1997.

[Sandgren 1977] Eric Sandgren. The utility of nonlinear programming algorithms. Technical report, Purdue University, 1977. Ph.D. Thesis.

[Zha *et al.* 1996] G.-C. Zha, Don Smith, Mark Schwabacher, Khaled Rasheed, Andrew Gelsey, and Doyle Knight. High performance supersonic missile inlet design using automated optimization. In *AIAA Symposium on Multidisciplinary Analysis and Optimization '96*, 1996.