

GENETIC ALGORITHMS FOR MIXED DISCRETE/CONTINUOUS OPTIMIZATION IN MULTIDISCIPLINARY DESIGN

Marc A. Stelmack *

Nari Nakashima †

Stephen M. Batill‡

Department of Aerospace and Mechanical Engineering
University of Notre Dame
Notre Dame, Indiana

Abstract

Genetic algorithms have been applied to a simple demonstration problem which had previously been used in the validation of a multidisciplinary design optimization framework referred to as Concurrent Subspace Design. Discrete optimization in the most recent implementation of that framework was performed by simulated annealing. The results obtained by applying both genetic algorithms and simulated annealing to the demonstration problem demonstrated that the performance of both methods was very similar, both in terms of their ability to locate optimal designs and the computational requirements involved in doing so. It was therefore concluded that the capabilities and computational cost of Concurrent Subspace Design would be similar regardless of which discrete optimization method was incorporated into the framework.

I. Introduction

Recent work in the field of multidisciplinary design optimization (MDO) has been inspired by the need to efficiently design complex, nonhierarchical systems. These typically involve many mutually dependent disciplines that must be coordinated, each of which has some impact on the system design goals. Many engineering systems to which MDO techniques are potentially applicable are “mixed,” in other words, contain both continuous and discrete design variables. The designer of an aircraft wing, for example, has the freedom to choose the number of spars in the wing, their

locations, and the material(s) from which the wing is to be constructed. The number of spars must be an integer and thus is a discrete design variable, while the location of a spar is a continuous design variable. Material selection is another example of a discrete design variable, as a structural designer would likely choose one of several available materials. Aircraft wing structural design is a problem that contains both discrete and continuous design variables, as all of this information would be required to perform the analyses involved in the design process. Problems such as this present a particular challenge because there are a limited number of methods which are both suitable for mixed problems and robust enough to be reliable in the context of designing a complex engineering system. Furthermore, discrete optimization methods are typically very expensive in terms of computational resources required.

The focus of this paper is the application of *genetic algorithms*, a method for performing discrete or mixed optimization, to an existing MDO framework called *Concurrent Subspace Design* (CSD). CSD is closely related to the Concurrent Subspace Optimization (CSSO) method proposed by Sobieski [1] and modified by Renaud and Gabriele [2] and by Sellar [3]. Both CSSO and CSD incorporate means by which discipline-level designers are provided with information regarding the influence of their decisions on the system-level objective function and constraints. During design at both the subspace and system levels in the current implementation of CSD, that information is provided by response surface approximations in the form of artificial neural networks. Previous efforts have demonstrated that this framework can be applied to the design of engineering systems that contain both discrete and continuous design variables [4]. This paper describes the application

*Graduate Research Assistant, Member AIAA

†Visiting Scholar (Patent Examiner, Japanese Patent Office)

‡Professor, Associate Fellow AIAA

Copyright ©1998 by Stephen M. Batill. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

analyses” (CA’s), are closed-form analytic expressions in the continuous design variable x_1 , state variables, and additional parameters (p_1, \dots, p_4). The latter are constant over the course of a single system analysis, but are dependent on the discrete design variables x_2 and x_3 . The functional form of each CA is given below.

$$\text{CA1: } y_1 = p_1^2 + \frac{p_2}{10} x_1^2 + p_3 y_2 \quad (1)$$

$$\text{CA2: } y_2 = \sqrt{y_1} \sin p_4 + x_1 + p_2$$

The values of parameters p_1, \dots, p_4 are dictated by the discrete design variables according to Table 1. The table is interpreted such that if x_2 is “red”, then $p_1 = 10$ and $p_2 = 1$. It is essential to note that the choice of x_2 determines both p_1 and p_2 . For instance, while $p_1 = 10$ when x_2 is “red” and $p_2 = 7$ when x_2 is “orange”, it is *not* possible that $p_1 = 10$ and $p_2 = 7$ simultaneously.

Table 1: Discrete Design Variables

x_2	p_1	p_2	x_3	p_3	p_4
red (R)	10	1	pink (Pnk)	0.1	$\pi/6$
orange (O)	2	7	white (Wht)	-0.1	$\pi/3$
yellow (Y)	7	8	gray (Gry)	0	$\pi/4$
green (G)	6	9	black (Blk)	-0.2	$\pi/2$
blue (B)	8	3	brown (Brn)	0.2	π
indigo (I)	3	9			
violet (V)	4	5			

The system analysis for this problem is the process by which, for a fixed design vector \bar{x} , the values of states y_1 and y_2 are determined. This process consists of first “looking up” the values of parameters p_1, \dots, p_4 corresponding to the discrete DV’s, and then solving iteratively the set of coupled, nonlinear algebraic equations (1).

The “merit” of each design is given by an analytic expression in design variables, related parameters, and states. The goal of the system optimization problem is to minimize the merit function while meeting constraints on the values of states y_1 and y_2 . The system optimization problem is stated as

$$\begin{aligned} \text{Min.: } f(\bar{x}, \bar{p}, \bar{y}) &= 5p_3 y_1 + \frac{y_2}{p_2} \\ &\quad + p_1 \sin\left(\frac{p_4 x_1}{5}\right) + 130 \\ \text{S.T.: } y_1 &\leq 60 \\ y_2 &\geq 10 \\ -10 &\leq x_1 \leq 10 \end{aligned} \quad (2)$$

This problem does not correspond to a particular physical application, but its size makes it a convenient benchmark for studying different mixed optimization methods.

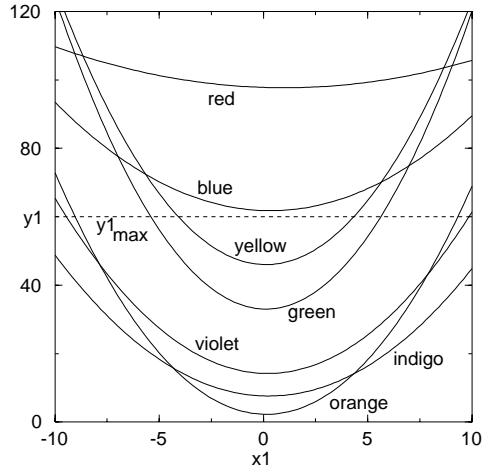


Figure 2: y_1 vs. x_1 for x_3 =black

A cross-section of the design space for this problem is illustrated in Figure 2. The figure depicts state y_1 as a function of x_1 for each of the seven possible “colors” (values) of x_2 while x_3 is fixed at black. The maximum value of y_1 allowed by the constraint on the optimization problem (Equations 2) is indicated by the dashed horizontal line on the graph. As can be seen from the figure, the character of y_1 vs. x_1 , and consequently the values of x_1 (if any) for which designs are feasible, depends upon which “colors” have been selected for the two discrete DV’s.

The global optimum for this problem was located by performing 35 one-dimensional optimizations, during which x_1 was the only active design variable and x_2 and x_3 were fixed. For each possible combination of x_2 and x_3 , the value of x_1 that minimized f was determined. Considering only the feasible solutions found (there are *no* feasible designs for some combinations of x_2 and x_3), that associated with the minimum value of f was $\bar{x} = [-4.08, \text{yellow}, \text{black}]$, at which point $f = 64.7$ and the first constraint is active ($\bar{y} = [60.0, 11.7]$). This point is indicated by the diamond on Figure 3, which shows the dependence of f on x_1 for the seven possible values of x_2 while x_3 is fixed at black. The circles on that figure show the locations of other design points which will be referred to as “local” optima. These points were obtained as solutions to the one-dimensional optimization problem for other “color combinations” (combinations of x_2 and x_3) and have merit values that were considered to be “competitive” (arbitrarily defined here as $f < 80$) with that of the global optimum. The locations of the global and local optima are listed in Table 2. Finally, while space does not

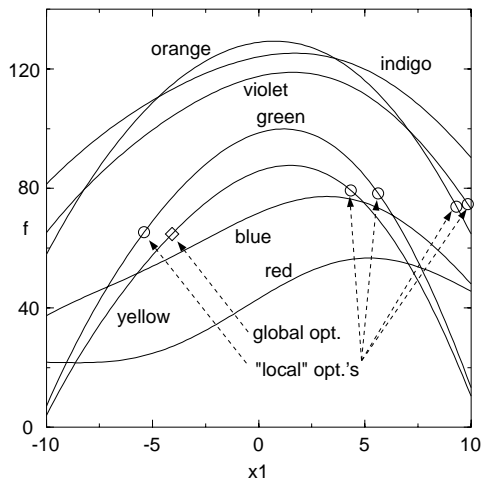


Figure 3: f vs. x_1 for x_3 =black

Table 2: Optimal Points

Glob/Loc	x_1	x_2	x_3	f
Global	-4.08	yellow	black	64.7
Local	-5.40	green	black	65.3
	9.32	orange	black	73.9
	9.85	violet	black	74.7
	5.63	green	black	78.4
	4.33	yellow	black	79.4

permit graphical presentation of the entire design space, it is worth noting that altering x_3 affects the behavior of the states and merit function in other ways. The range of f throughout the entire design space is approximately $0 < f < 270$.

This problem has also been solved using the Concurrent Subspace Design framework [6]. The optimization that took place within that implementation of CSD was not purely discrete, but rather mixed continuous/discrete. The method used to perform this optimization was a hybrid method in which the continuous design variables were initially discretized with some specified resolution and the problem solved as a fully discrete one. Subsequently, the discrete design variables were fixed at those values specified by the discrete optimizer and the final design obtained by performing a gradient-based optimization, during which only the continuous design variables were active. This hybrid method, along with the reasons for performing mixed optimization this way, are described in more detail in [9].

IV. Genetic Algorithms

Genetic Algorithms (GA's) are stochastic search techniques which mimic the mechanism of natural evolution. Unlike many other search techniques, GA's consider multiple designs at each iteration. Such a set of designs is referred to as a "population." The designs in a population are called "chromosomes" or "individuals" and are typically encoded into binary strings. During each iteration, the individuals which comprise the population undergo three operations: selection, crossover, and mutation. These processes are briefly described below and illustrated in Figure 4.

Structure of Genetic Algorithms

Selection is the operation which determines which chromosomes will be represented, and to what extent, in the next population. It is based on a "fitness function" which is defined such that good designs have high fitness values relative to bad designs. In the context of constrained optimization, the fitness function is usually some metric which incorporates the objective function plus one or more penalty terms corresponding to the constraints. The latter are included to discourage infeasible designs. A common selection strategy is "roulette wheel" selection, in which the probability of selecting a specific chromosome is proportional to its fitness value. The sum of these probabilities is unity. "Spinning" the roulette wheel then amounts to generating a random number to determine which chromosome will be selected; this is repeated as many times as there are designs in each population. This generally results in the most fit chromosomes being selected more than once and the least fit being discarded. The group that gets selected during this phase of the algorithm is called the "mating pool."

Crossover takes place after selection and occurs with a predetermined probability, the *crossover rate* (P_c). During "single cut point crossover," the simplest form of this operation, a pair of chromosomes is selected from the mating pool and a "cut point" is randomly determined for that pair. All the binary digits which occur after the cut point in one of the chromosomes are then replaced by the digits after the cut point in the other chromosome, and vice-versa. This procedure generates two new designs, each of which possesses some characteristic(s) of previous designs that had relatively high fitness. This is analogous to offspring inheriting traits of their parents. The number of pairs of designs which undergo crossover is dependent on the crossover rate.

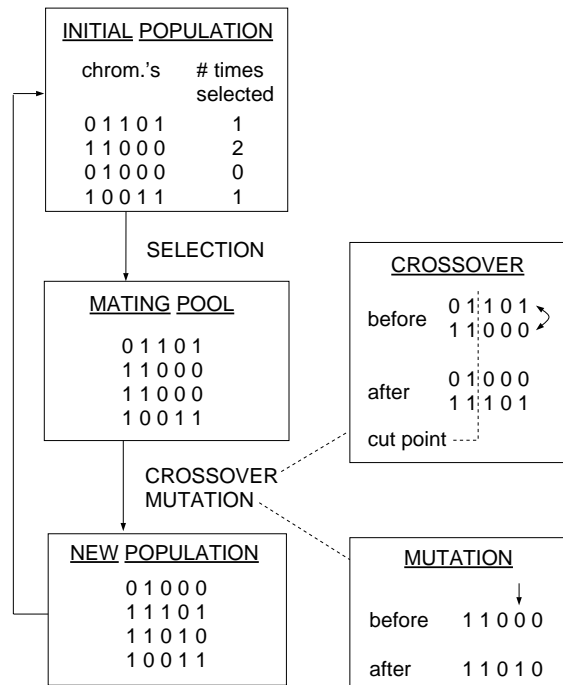


Figure 4: Basic Structure of a Genetic Algorithm

The last operation to take place is mutation. This involves simply altering the value of a few randomly selected bits throughout the population. This occurs with a predetermined probability called the *mutation rate* (P_m), the value of which influences how many bits will be changed. Mutation is intended to enable GA's to explore new regions of the design space and escape from local optima. When it is complete, the next population of designs is the result. From this point, the processes of selection, crossover, and mutation are repeated until some termination criteria are met.

Implementation of Genetic Algorithms

The demonstration problem described previously was solved using a genetic algorithm in which roulette wheel selection, single cut point crossover, and random mutation were implemented. The design vector $\{x_1, x_2, x_3\}$ was encoded into an 11-bit binary string, with the bits allocated as shown below.

$$\underbrace{00101}_{x_1} \underbrace{011}_{x_2} \underbrace{100}_{x_3}$$

The continuous variable x_1 was discretized into 21 values, specifically, the integers -10, -9, ..., 10. Also, the discrete variables x_2 and x_3 have 7 and 5 possible values, respectively. Thus, some binary strings do not represent any designs; such strings are called "redundant." It is desirable to

maintain a 1-to-1 mapping, in which only one binary string maps to each design [10], so every new chromosome created was checked to insure it corresponded to a valid design. Those chromosomes that did not meet this criterion were replaced with new, randomly generated ones.

A very simple termination criterion was employed for the genetic algorithms discussed herein. The process stopped when there was no improvement in the best chromosome for two successive iterations. This method, as well as that of replacing redundant chromosomes, was selected primarily for ease of implementation.

Initially, the population size, crossover rate, and mutation rate were the only control parameters adjusted. During later experimentation, three other techniques were employed in attempts to improve the performance of the genetic algorithms. These techniques are listed and briefly described below.

Elitism

It is possible for chromosomes with high fitness to be eliminated during selection, crossover, or mutation. Elitism is a means of retaining those chromosomes in the next generation. As implemented in this study, the single best chromosome yet encountered replaced the worst chromosome present in the event that the former was found to be eliminated during selection.

Linear Scaling

Linear scaling involves adjusting all fitness values according to the formula

$$F' = aF + b \quad (3)$$

where F and F' are the raw and scaled fitness values, respectively, and coefficients a and b are control parameters. The latter are used to control the frequency with which the best chromosome gets selected relative to that with which chromosomes of average fitness get selected. For example, it is possible to set a and b such that the probability of selecting the best chromosome is double that of selecting an average chromosome. The term "average" in this sense refers to only designs in the current population. In the event that some scaled fitness values become negative, the coefficients may be adjusted set such that the worst raw fitness value is scaled to zero.

Enlarged Sampling Space

This technique increases the number of chromosomes considered during selection to include not only those that resulted from the latest crossover ("offspring"), but also those that existed

immediately prior to that crossover (“parents”). In other words, the selection pool consists not only of the current population, but of the most recent mating pool as well.

V. Results

The results presented in this section focus on the ability of genetic algorithms to locate optimal designs for the demonstration problem described previously and the computational requirements required to do so. Again, these issues are of interest because they are directly related to whether it would be beneficial to replace simulated annealing with genetic algorithms as a method of discrete optimization at the subspace design step of the CSD algorithm. It was not necessary to implement the CSD framework using genetic algorithms, rather, qualitative assessments of the impact they would have on CSD were made by comparing their performance on this demonstration problem to that of simulated annealing. For the sake of making that comparison, some results of solving the same problem via simulated annealing are also included.

With regards to both methods, constrained optimization is performed by augmenting the objective function with penalty terms corresponding to the constraints. Only two penalty functions were considered during this study; a more complete discussion of penalty functions is available in [11]. The forms of the penalized merit functions, f_1 and f_2 , are

$$f_1 = f + \max(y_1 - 60, 0) + \max(10 - y_2, 0) \quad (4)$$

$$f_2 = f + 2 \max(y_1 - 60, 0) + 2 \max(10 - y_2, 0) \quad (5)$$

where f , y_1 , and y_2 correspond to the objective function and states of the demonstration problem described earlier.

It is important to keep in mind that, as mentioned earlier, the optimization that takes place within CSD is performed using a hybrid method in which discrete and continuous optimizations occur sequentially. A consequence of this is that the designs yielded by simulated annealing and genetic algorithms in this study would not be the final results of subspace optimization in CSD, but rather the “intermediate solutions” from which the continuous optimization begins. This is considered in the discussion of the results that follow.

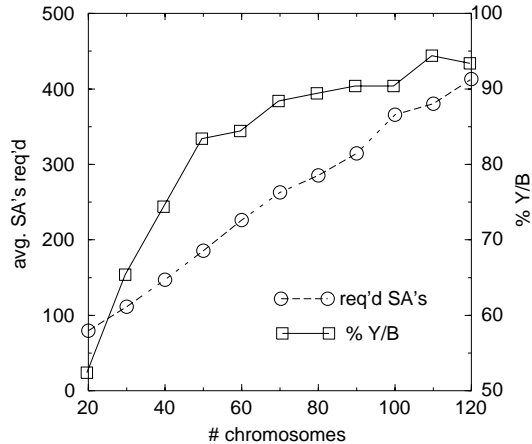


Figure 5: GA Population Size

Selection of Population Size

A considerable number of combinations of population size (number of chromosomes), crossover rate, and mutation rate were implemented during initial experimentation with genetic algorithms. While space does not permit a description of all these results, Figure 5 shows the impact of population size on both the ability of genetic algorithms to locate optimal designs and the number of system analyses (SA's) required to do so. Each data point shown on the figure reflects the average of 250 trials in which penalized merit function f_1 (Equation 4) was used, the crossover rate was 0.9, and the mutation rate varied from 0.05 to 0.001 as the number of chromosomes increased. The latter was not held constant because, according to Goldberg [12], the performance of genetic algorithms is enhanced if the mutation rate drops as population size increases.

The dashed line in Figure 5 shows that, over the range of population sizes considered, the relationship between the number of chromosomes and the number of SA's required is approximately linear. The ability to locate optimal designs is reflected as the percentage of trials in which the genetic algorithm determined that $x_2 = \text{yellow}$ and $x_3 = \text{black}$ (their values at the global optimum) for the demonstration problem, indicated as “%Y/B” on the graph. The value of the continuous variable x_1 determined by the GA was not considered at this point because, in the context of the mixed optimization method used in CSD, its final value is not determined by the discrete optimizer (subsequent results will demonstrate that the ability to determine the optimal values of the discrete DV's, rather than locate the design nearest the optimum, is the critical characteristic of a discrete optimizer

used as part of this mixed optimization scheme). The solid line on the graph indicates that when up to 50 chromosomes were used, the percentage of trials which successfully determined those values increased much more quickly than it did after the population size exceeded 60. Thus, 50 chromosomes were used in subsequent implementations because the cost of using a larger population (more required SA's) provided only marginally better performance.

Purely Discrete Optimization

As stated previously, the first phase of the mixed optimization method being considered is to discretize the continuous design variables and solve the problem as a fully discrete one. The results of solving the demonstration problem described earlier, having discretized x_1 into the integers -10, -9, . . . , 10 using 3 different methods of discrete optimization are summarized in Table 3. The three methods considered were simulated annealing and two different genetic algorithms. Of the latter, the first was a "simple" GA in which $P_c=0.9$ and $P_m=0.01$. The performance of this genetic algorithm was altered through the use of various combinations of the elitism, linear scaling, and enlarged sampling space techniques. After some experimentation it was determined that improved performance could be obtained by adding elitism and linear scaling such that the probability of selecting the best design in each population was 4 times that of selecting an average design in the same population. For each method, the table lists the number of trials, out of 250, in which either the global (#G) or one of the local (#L) optima listed in Table 2 was located. Obviously, all of those optima lie between points in the discretized space. The trials considered to have located the global optimum all resulted in (-4, Y, Blk), which in the discretized space is the closest point to the global optimum. It is also the only point in the discretized space within 0.5 units in the x_1 direction of the global optimum; an analogous criterion was used to determine which trials located local optima. The middle two columns correspond to penalized merit function f_1 (Equation 4) while the rightmost two columns show the same data for function f_2 (Equation 5). The improved GA was never applied to the latter, for reasons to be discussed later.

Table 3 suggests that the choice of penalty function actually dictates performance to a much larger extent than does the optimization method. With both simulated annealing and genetic algorithms, optimal points were located much more

Table 3: Discrete Optimization Results

method	function f_1		function f_2	
	# G	# L	# G	# L
Sim. Ann.	21	0	84	65
Simple GA	27	0	75	48
Improved GA	31	0	-	-

often using function f_2 than they were with f_1 . The reason for this becomes evident if one considers the shape of the design space. Referring back to Figure 3, moving x_1 towards its lower bound from the global optimum reduces the merit function, however, Figure 2 shows that this also results in violating the constraint $y_1 \leq 60$. The considerable difference in performance between the two penalty functions is due to the fact that while the larger penalty in f_2 effectively bounds the feasible region, infeasible designs such as (-5, Y, Blk), (-6, Y, Blk), etc. were prevalent among results obtained using f_1 .

The results in Table 3 are of limited utility in comparing the performance of genetic algorithms to that of simulated annealing. The former were able to locate the global optimum slightly more often using function f_1 , and both the global and local optima slightly less often using the larger penalty functions. The differences in performance, however, reflect a relatively small fraction of the 250 total trials performed in each case.

Mixed Optimization

The generalized reduced gradient (GRG) method was used to perform a continuous, one-dimensional optimization, in which x_1 was the only active design variable, starting from every point that was obtained through the purely discrete optimization methods described above. This was done to obtain the results of the three variations of the mixed optimization scheme created by preceding the continuous optimization with each of the discrete methods being considered. The results yielded by mixed optimization are summarized in Table 4, which is in the same format as Table 3. In this case, the designs obtained were considered to be located at an optimum if they were less than 0.01 unit in the x_1 direction away from one of the optima listed in Table 2.

By comparing the results in Tables 3 and 4, one can determine the number of trials in which the discrete optimizer identified the desired values of the discrete design variables (x_2 = yellow and x_3 = black) but did not locate the value of x_1 in the discretized space which was nearest the optimum.

Table 4: Mixed Optimization Results

method	function f_1		function f_2	
	# G	# L	# G	# L
Sim. Ann.	233	13	121	89
Simple GA	208	20	129	67
Improved GA	222	16	-	-

For example, the simple genetic algorithm alone located (-4, Y, Blk) in only 27 of 250 trials using function f_1 , but when a continuous optimization was performed from the resultant points, the global optimum (-4.08, Y, Blk) was obtained 208 times. Thus, in 208-27=181 trials, the simple genetic algorithm itself did not locate the global optimum, but did determine the optimal values of the discrete design variables, x_2 and x_3 . When the GA was incorporated into the mixed scheme, this was sufficient as the subsequent GRG optimization adjusted x_1 to its optimal value.

This comparison of Tables 3 and 4 indicates that the discrete optimizers yielded designs which were non-optimal solely due to their x_1 values considerably more often when function f_1 was used as opposed to f_2 . As mentioned previously, this trend is attributable to the relative magnitudes of the raw merit function f and the penalty terms, as the penalties on f_1 weren't large enough to discourage infeasible designs with lower objective values than the optimum. A consequence of this is that during mixed optimization, performance was markedly better when merit function f_1 was used. This is in contrast to purely discrete optimization, when function f_2 resulted in optimal designs being located much more frequently. Obviously, this phenomenon is dependent on the fact that in this design space, the infeasible designs that result from the smaller penalty functions are characterized by the same values of the discrete design variables as the global optimum. Although the influence of a specific form of penalty function on an optimizer's performance is certainly a problem-dependent issue, these results do emphasize a point that is perhaps non-intuitive. Specifically, when a discrete method is incorporated into the mixed optimization scheme considered herein, its most critical characteristic is not the ability to pinpoint the optimal design, but rather to determine the optimal values of the discrete design variables.

It is obvious from Table 4 that the mixed optimization scheme is more successful than purely discrete optimization in locating optimal points. The hybrid schemes consisting of simu-

lated annealing and the simple genetic algorithm followed by GRG optimization located the global optimum in 83% and 93%, respectively, of the trials conducted using function f_1 . The same schema applied to f_2 only located the global optimum in roughly half of the corresponding trials. The improved GA was never applied to penalized merit function f_2 because the goal of this study was to determine which discrete method would result in better performance *when incorporated into the mixed scheme*. Furthermore, the observed difference in behavior between f_1 and f_2 was considered to be problem-dependent and not one that could be alleviated by the improved genetic algorithm.

As with purely discrete optimization, the relative performance of genetic algorithms versus simulated annealing depended on the penalty functions used. Restricting discussion, however, to mixed optimization using function f_1 for reasons detailed above, Table 4 indicates that simulated annealing performed slightly better for this problem. First, optimal designs were located in a greater number of trials, 246 as opposed to 228 and 238 with the simple and improved genetic algorithms, respectively. Also, the percentage of total trials, as well as the percentage of trials which located an optimum, which resulted in the *global optimum* were both higher for simulated annealing than for either of the genetic algorithms.

Computational Requirements

The number of analyses required by any optimization method is an important consideration in MDO. It becomes paramount in the context of a framework such as CSD, in which the optimization methods employed are invoked repeatedly as multiple iterations are performed. Table 5 lists the number of system analyses required by the methods implemented in this study. As before, the columns in the table are grouped according to the two penalized merit functions used. Within each group, the column designated "D" corresponds to purely discrete optimization and the column designated "M" corresponds to mixed optimization. Again, the values reflect the average of the 250 trials performed in each case.

Table 5: Computational Requirements

method	function f_1		function f_2	
	D	M	D	M
Sim. Ann.	171.2	179.9	121.1	130.4
Simple GA	185.9	195.3	185.3	195.1
Improved GA	168.7	177.9	-	-

A number of statements can be made regarding the results listed in Table 5. It can be seen that for this problem, the number of system analyses required for mixed optimization is only slightly greater than the number required to perform the corresponding discrete optimization. This is not surprising, as discrete optimization methods are notoriously computationally expensive, while one-dimensional continuous optimization is fairly straightforward. Of greater interest are the relative requirements of the different discrete methods. There is a pronounced difference between simulated annealing and the simple GA seen in the cases corresponding to function f_2 ; whether these computational savings would be worth the reduced performance observed by using f_2 as opposed to f_1 in mixed optimization, however, is dependent on many factors which are beyond the scope of this discussion.

While in all cases the simple genetic algorithm required slightly more system analyses than simulated annealing and the improved genetic algorithm slightly less, the computational requirements were on the same order of magnitude for all three methods. One conclusion that can be drawn from this is that utilizing genetic algorithms within CSD would not result in substantial computational savings for problems of this character and scope. Thus, if subspace designers wished to incorporate full-fidelity analysis tools for large, complex problems during subspace design, genetic algorithms would not provide a viable means of performing traditional optimization at that level.

Repeatability of Results

A relatively large number of trials, 250, was run for all the cases considered in this study. This was done because both simulated annealing and genetic algorithms are stochastic search methods, and it was not known a priori how many trials would be adequate for the results to provide a valid comparison of the different methods. Figures 6 and 7 show the number of trials which located the global optimum and the average number of system analyses required, respectively, versus the total number of trials performed, with data points shown on intervals of 50 total trials for clarity. Although the figures correspond to the case in which mixed optimization using the improved GA was applied to function f_1 , they are representative of results from other cases. Figure 6 shows that a similar number out of each subset of 50 trials resulted in the global optimum. Also, Figure 7 indicates that the average number of system analyses required remained between 176 and 178

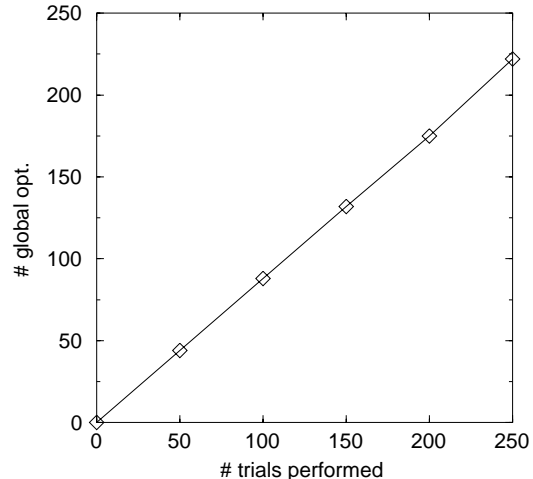


Figure 6: Trials locating global optimum

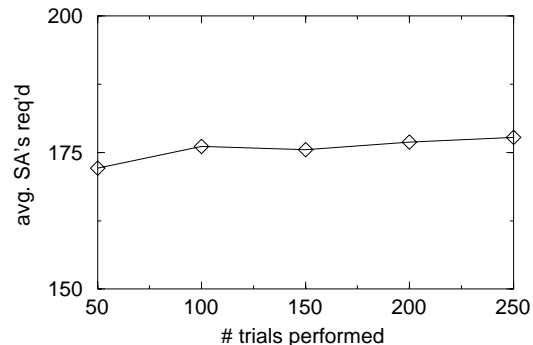


Figure 7: Required System Analyses

during the last 150 trials. This suggests that the results obtained were representative of the methods being considered, rather than a few anomalous trials.

VI. Summary

The method of genetic algorithms has been applied to a simple demonstration problem containing one continuous and two discrete design variables. This problem has previously been used to validate a multidisciplinary design optimization framework referred to as Concurrent Subspace Design, in which discrete optimization was performed by simulated annealing. The problem was solved both as a fully discrete one and using the mixed optimization scheme included in the most recent implementation of CSD, in which discrete and continuous optimizations are performed sequentially. The purpose of this study was to compare the performance of simulated annealing with that of genetic algorithms for this problem, and to assess whether genetic algorithms would significantly al-

ter the computational requirements of CSD.

Results demonstrated that for this problem, there were no substantial performance differences between the discrete optimization methods considered, neither when those methods were implemented by themselves, nor when they were incorporated into the mixed optimization scheme. In fact, the manner in which the constraints were formulated as penalty functions, a problem-dependent issue regardless of the method used, affected the results of optimization to a much greater extent. Using the specific penalty functions which resulted in the global optimum being located in the greatest percentage of trials, simulated annealing located the global optimum in 233 of 250 trials. By comparison, the better-performing of the genetic algorithms implemented in this study located the optimum in 222 trials. The computational requirements of the two methods were also nearly equal; an average of 179.9 system analyses were required to perform mixed optimization using simulated annealing, compared to 177.8 with the genetic algorithm.

From the results obtained, it was concluded that replacing simulated annealing with genetic algorithms as the method of discrete optimization in CSD would not have a profound effect on the performance or capabilities of the framework. A consequence of this is that repeated use of full-fidelity analysis tools, as is required by traditional optimization methods, at the subspace design level is not a realistic approach for problems of significant scope. Thus, to prevent CSD from being cost-prohibitive for large, complex problems, subspace design must exploit response surface approximations and/or consist of less rigorous design methods.

Acknowledgements

This work was supported in part by the National Aeronautics and Space Administration, Langley Research Center, Grant NAG-1-1561. Dr. J. Sobieski, Project Monitor. Sincere thanks also to the Japanese Patent Office for allowing Mr. Nakashima to take part in this effort.

References

- [1] J. Sobiesszczanski-Sobieski. Optimization by Decomposition: A Step From Hierarchic to Non-Hierarchic Systems. NASA Conference Publication 3031, Part 1, Second NASA/Air Force Symposium on Recent Advances in Multidisciplinary Analysis and Optimization, Hampton, Virginia, September 1988.
- [2] J. E. Renaud and G. A. Gabriele. Improved coordination in nonhierarchic system optimization. *AIAA Journal*, 31(12), December 1993.
- [3] R. S. Sellar, S.M. Batill, and J.E. Renaud. Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design. 34th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 96-0714, Reno, Nevada, January 1996.
- [4] M. A. Stelmack, S. M. Batill, B. C. Beck, and D. J. Flask. Application of the concurrent subspace design framework to aircraft brake component design optimization. 39th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA 98-2033, Long Beach, California, April 1998.
- [5] Jacek M. Zurada. *Introduction to Artificial Neural Systems*. PWS Publishing Company, 1992.
- [6] M. A. Stelmack and S. M. Batill. Neural network approximation of mixed continuous/discrete systems in multidisciplinary design. 36th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 98-0916, Reno, Nevada, January 1998.
- [7] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein. Generalized simulated annealing for function optimization. *Technometrics*, 28(3), 1986.
- [8] G. Gabriele and T. Beltracchi. *OPT3.2: A FORTRAN Implementation of the Generalized Reduced Gradient Method*. Department of Aerospace and Mechanical Engineering and Mechanics, Rensselaer Polytechnic Institute, 1988.
- [9] M. A. Stelmack and S. M. Batill. Concurrent Subspace Optimization of Mixed Continuous/Discrete Systems. 38th AIAA/ASME/-

ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA 97-1229, Kissimmee, Florida, April 1997.

- [10] M. Gen and R. Cheng. *Genetic Algorithms and Engineering Design*. John Wiley and Sons, Inc., 1997.
- [11] G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. *Engineering Optimization: Methods and Applications*. John Wiley and Sons, 1983.
- [12] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley Longman, Inc., 1989.