EVALUATION OF GENETIC ALGORITHM CONCEPTS USING MODEL PROBLEMS PART 1: SINGLE-OBJECTIVE OPTIMIZATION

Terry L. Holst and Thomas H. Pulliam
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

A genetic-algorithm-based optimization approach is described and evaluated using a simple hill-climbing model problem. The model problem utilized herein allows for the broad specification of a large number of search spaces including spaces with an arbitrary number of genes or decision variables and an arbitrary number hills or modes. In the present study, only single objective problems are considered. Results indicate that the genetic algorithm optimization approach is flexible in application and extremely reliable, providing optimal results for all problems attempted. The most difficult problems—those with large hyper-volumes and multi-mode search spaces containing a large number of genes—require a large number of function evaluations for GA convergence, but they always converge.

Nomenclature

D	multi-mode design space parameter defined by Eq. (12)
E	normalized error defined by Eq. (10)
$\mathbf{G}^n$	n^{th} GA generation
NSEED	user-specified parameter that controls how many solutions with different initializing random number generator seeds are averaged together to construct a single convergence history curve
N_C	number of chromosomes in each GA generation
N_G	number of genes in each chromosome
N_M	number of modes (hills or peaks) in the model problem defined by Eq. (9)
N_O	number of scalar objective functions
P	user-specified vector with four elements that controls which modification operators are used in going from $\mathbf{G}^n$ to $\mathbf{G}^{n+1}$
p_1	user-specified parameter controlling the probability that a specific gene will be modified using the perturbation mutation operator ($0 \le p_1 \le 1$)
p_2	user-specified parameter controlling the probability that a specific gene will be modified using the mutation operator ($0 \le p_2 \le 1$)
R	user-specified parameter that controls the size or range of the design space associated with MP1 [see Eq. (4)]
$R(0,1)$	random number generator that returns a random value between 0 and 1
$x_{i,j}^n$	i^{th} gene from the j^{th} chromosome from the n^{th} GA generation
$\mathbf{X}_j^n$	j^{th} chromosome from the n^{th} GA generation
$x_{\max_i}$	user-specified maximum limit on the i^{th} gene
$x_{\min_i}$	user-specified minimum limit on the i^{th} gene
β	user-specified parameter controlling the size of the perturbation mutations ($0 \le \beta \le 1$)

subscripts

i	gene or decision variable index
j	chromosome index

k	objective function index
m	mode index associated with the model problem defined by Eq. (9)

superscripts

n	GA generation or population index
temp	temporary chromosome and gene values obtained after selection but before operator modification

Background

Numerical methods for optimizing the performance of engineering problems have been studied for many years. Perhaps the most widely used general approach involves the computation of sensitivity gradients. These methods—called gradient methods—have been utilized to produce optimal engineering performance in a wide variety of different forms. The reliability and success of gradient methods generally requires a smooth design space and the existence of only a single extremum or an initial guess close enough to the global extremum that will ensure proper convergence.

In contrast to gradient based methods, design space search methods such as genetic algorithms (GA) offer an alternative approach with several attractive features. The basic idea associated with the GA approach is to search for optimal solutions using an analogy to the theory of evolution. The problem to be optimized is parameterized into a set of decision variables or genes. Each set of genes that fully defines one design is called an individual or a chromosome. A set of chromosomes is called a population or a generation. Each complete design or chromosome is evaluated using a fitness function that determines survivability of that particular chromosome. For example, in aerospace applications, the genes may be a series of geometric parameters associated with an aerospace vehicle that is to be optimized for payload delivered to orbit, aerodynamic performance or structural weight. The fitness function takes as input all the geometric parameters and returns the fitness—the size of the payload, the aerodynamic performance or the structural weight.

During solution advance (or "evolution" using GA terminology) each chromosome is ranked according to its fitness. The higher-ranking chromosomes are selected to continue to the next generation—usually multiple times—while the lower-ranking chromosomes are not selected at all. The newly selected chromosomes in the next generation are manipulated using various operators (combination, crossover or mutation) to create the final set of chromosomes for the new generation. These chromosomes are then evaluated for fitness and the process continues—from generation to generation—steadily improving the design.

Constraints can easily be included in the GA optimization approach either by direct inclusion into the fitness function definition or by preprocessing the candidate design. For example, if a design violates a constraint, its fitness is set to zero (for cases involving maximization), i.e., it does not survive to the next evolution level. Because GA optimization is not a gradient-based optimization technique, it does not need sensitivity derivatives. It theoretically works well in non-smooth design spaces containing several or perhaps many local extrema.

General GA details including descriptions of genetic algorithms can be found in Goldberg,[1] Davis,[2] and Beasley, et al.[3,4] Additional useful studies which survey recent activities in the area of genetic algorithm or evolutionary algorithm research including the presentation of model problems useful for evaluating GA performance are given in Deb,[5] Jiménez, et al.[6] and Van Veldhuizen and Lamont.[7]

A disadvantage of the GA approach is expense. In general, the number of function evaluations required for the GA optimization process to converge exceeds the number required by a finite-difference-based gradient optimization (see the results presented in Obayashi and Tsukahara,[8] Bock[9] and Pulliam, et al.[10]).

This situation is offset, to an extent, by the ease with which GAs can be implemented in parallel or distributed computing environments.
Despite being relatively new, genetic algorithms have already been applied in a broad variety of aerospace design applications. A few single discipline applications include a micropump optimization by Sharatchandra, et al.,[11] wing induced drag minimization by Gage and Kroo,[12] control systems engineering applications by Fleming and Purshouse,[13] transonic wing aerodynamic shape optimization by Holst and Pulliam,[14] computer system efficiency enhancements by Globus, et al.,[15] circuit design by Lohn and Colombano,[16] antenna design by Linden and Altshuler,[17] space vehicle atmospheric reentry optimization by Peigin,[18] airfoil design by Tse and Chan,[19] actuator placement optimization by Sheng and Kapania[20] and Cook and Crossley,[21] air traffic control optimization by Cheng, et al.,[22] and turbine blade shape optimization by Cravero and Satta.[23]

Other applications involving GA search methods have been in the area of multi-objective or multi-discipline optimization. GA optimization techniques are especially attractive in this area because they offer the ability to directly compute so-called "pareto optimal sets" instead of the limited single design point traditionally provided by other methods. Examples of multi-objective optimization applications include airfoil optimization by Marco, et al.,[24] Naujoks, et al.,[25] Quagliarella and Della Cioppa,[26] Vicini and Quagliarella,[27] and Hämäläinen, et al.,[28] missile aerodynamic shape optimization by Anderson, et al.,[29] wing optimization by Anderson and Gebert,[30] Sasaki, et al.,[31] Oyama,[32] and Obayashi, et al.,[33] and rocket engine turbopump design by Oyama and Liou.[34] In some of these examples the multiple objectives were obtained by considering two different aerodynamic design points. In others, the multiple objectives involved different disciplines including aerodynamics, structures, controls and/or electromagnetics.

The Genetic Algorithm used in the present study is described next. Details associated with each of the operators, including selection, passthrough, random average crossover, perturbation mutation and mutation are presented. Genetic Algorithm convergence efficiency is then evaluated using a multi-mode, multi-gene hill-climbing problem from two general points of view—the effect of design space characteristics on GA convergence and the effect of GA control parameter specification on GA convergence.

Problem Statement: Single Objective Optimization

A single-objective optimization problem can be stated as follows: Let f be a scalar objective function of N_G independent variables, x_i, defined on some domain Ω

$$f = f(\mathbf{X}) = f(x_1,\cdots,x_i,\cdots,x_{N_G}) \qquad (1a)$$

In this notation $\mathbf{X}$ is the vector of design space decision variables. The maximum value of f, indicated by f^*, is obtained by finding the values of $\mathbf{X} = \mathbf{X}^*$ such that[†]

$$f^* = \max\{f\} = f(\mathbf{X}^*) = f(x_1^*,\cdots,x_i^*,\cdots,x_{N_G}^*) \qquad (1b)$$

The above maximization operation is subject to N_{CO} conditions or constraints indicated by

$$c_n(\mathbf{X}) \le 0 \quad n = 1,2,\cdots,N_{CO} \qquad (1c)$$

The constraints placed on the decision variable vector $\mathbf{X}$ by Eqs. (1c) serve to limit the design space within Ω for which the optimal solution is sought.

[†] For the purpose of simplifying the discussion of algorithmic details, maximization is generally assumed. The logic for minimization is a straightforward modification and will not be discussed.

Genetic Algorithm

The genetic algorithm optimization procedure utilized to solve the optimization problem, as described by Eqs. (1), is now presented. As mentioned above the general idea behind GA optimization is to discretely describe the design space using a number of decision variables, x_i. In GA parlance these parameters are called genes, and the i subscript is called the gene number. Each set of genes that leads to the complete specification of an individual design, i.e., each decision variable vector, $\mathbf{X}$, is called a chromosome and is indicated by

$$\mathbf{X}_j^n = \mathbf{X}(x_{1,j}^n, x_{2,j}^n, \cdots, x_{i,j}^n, \cdots, x_{N_G,j}^n) \qquad (2)$$

The j subscript, which has been added to $\mathbf{X}$, is the chromosome number. The j subscript has also been added to each gene, so as to identify which chromosome each gene value is identified with. The n superscript has been added to indicate the GA generation number, which iteratively advances as the solution converges. In this notation, $\mathbf{X}_j^n$ is the jth chromosome for the nth generation that consists of N_G genes.

For aerodynamic shape optimization problems, the design space genes are typically a series of geometric parameters, e.g., airfoil thickness and camber and/or wing sweep, twist and taper. For many GA applications genes are computationally represented using bit strings and the operators used to manipulate them are designed to accommodate bit string data. In the present approach, following the arguments of Oyama,[35] Houck, et al.[36] and Michalewicz,[37] real-number encoding is used to represent all genes. The key reason for using real number encoding is that it has been shown to be more efficient in terms of CPU time relative to binary encoded GAs.[37] In addition, real numbers are used for all genes in the present implementation because many engineering applications involve decision variables that are best described using real numbers, e.g., the geometric parameters in aerodynamic shape optimization. Thus, using real number encoding eliminates the need for binary-to-real number conversions.

Initialization

Once the design space has been defined in terms of a set of real-number genes, the next step is to form an initial generation, $\mathbf{G}^0$, represented by

$$\mathbf{G}^0 = (\mathbf{X}_1^0, \mathbf{X}_2^0, \cdots, \mathbf{X}_j^0, \cdots, \mathbf{X}_{N_C}^0)$$

where N_C is the total number of chromosomes. Each gene within each chromosome is assigned an initial numerical value using a process that randomly chooses numbers between fixed user-specified limits. For example, the ith gene in an arbitrary chromosome is initially computed using

$$x_i = R(0,1)(x_{\max_i} - x_{\min_i}) + x_{\min_i} \qquad (3)$$

where $x_{\max_i}$ and $x_{\min_i}$ are the upper and lower limits for the ith gene, respectively, and $R(0,1)$ is a random number generator that delivers an arbitrary numerical value between 0.0 and 1.0.

The random number generator used in the present study requires an integer input—a seed value. If the integer is positive, the next number in the current random number sequence is returned. If the integer is negative, the random number sequence is reset. Utilization of the same negative seed value will always reset the random number generator to the same random sequence. Each new solution begins by resetting the random number generator using a single call to $R(0,1)$ with a negative seed value. All other calls to $R(0,1)$ during that solution use a positive seed value. Thus, a solution can be repeated by simply using the same initial seed value or rerun to determine statistical variation by using a different initial seed value.

For simplicity in the present study, the gene limit values for each gene are forced to be equal, and the minimum values are forced to be equal to the negative of the maximum values. This is indicated by

$$x_{\max_1} = x_{\max_2} = \ldots = x_{\max_{NG}} = x_{\max}$$
$$x_{\min_1} = x_{\min_2} = \ldots = x_{\min_{NG}} = x_{\min}$$
$$x_{\min} = -x_{\max}$$

With these assumptions the complete specification of all gene limits can be made using a single parameter given by

$$R = x_{\max} - x_{\min} \tag{4}$$

This provides a simple way to study the effect of gene limits on GA convergence performance. Use of this simplifying assumption does not limit the generality of the present GA. Separate specification of each gene limit is retained as an option, but is not utilized in this study.

Fitness evaluation

After each generation is established—this includes the initial generation, as well as each succeeding generation in the evolutionary process—fitness values, f_j^n, are computed for each chromosome using a suitable function evaluation. This is analogous to the objective function evaluation in gradient methods and is represented using

$$f_j^n = f(\mathbf{X}_j^n) \tag{5}$$

For aerodynamic shape optimization, f represents a suitable CFD flow analysis that provides a quantitative evaluation for the desired objective, which is typically some function of lift, drag or some integral on the surface pressure distribution.

Ranking

Fitness evaluation is followed by a ranking process where the chromosome with the highest fitness is given a number one ranking ($IR = 1$), the individual with the second highest fitness is ranked number two ($IR = 2$), and so on. This process is represented as follows:

$$\left.\begin{array}{l} ic = 1 \\ \left.if\,(f_j^n < f_{jj}^n)\; ic = ic + 1 \quad jj = 1, N_C \right. \\ IR_j^n = ic \end{array}\right\} j = 1, N_C$$

where j and jj are special counters that range over all chromosomes in the current population or generation level.

This completes the GA ranking process. The next several subsections describe the GA selection process and how the chromosomes, once selected, are then modified.

Selection

The first operation required to produce generation $n+1$ is selection. The chromosomes that will be used by the GA modification operators (to be discussed shortly) must be selected from the n^{th} generation of chromosomes, i.e., from $\mathbf{G}^n = (\mathbf{X}_j^n)$. The selection operation used in the present study, sometimes called "greedy selection," is given by

```
$jj = 1$
if $(IR_j^n \le jj)$ then     }                  }
    $\mathbf{X}_{jj}^{temp} = \mathbf{X}_j^n$   }                  }
    $jj = jj + 1$             } $j = 1, N_C$     } $it = 1, N_C$
endif                         }                  }
if $(jj > N_C)$ stop                             }
```

where each selected chromosome $\mathbf{X}_{jj}^{temp}$ is placed in a temporary holding array indicated by

$$\mathbf{G}^{temp} = (\mathbf{X}_1^{temp}, \cdots, \mathbf{X}_{N_C}^{temp})$$

Note how the individuals with highest fitness in the n^{th} generation are always selected multiple times, thus, the name greedy selection. The individuals with average fitness are selected a small number of times, and the individuals with lowest fitness are not selected at all. This biasing toward individuals with highest fitness, although a key element in any GA, is taken to the extreme in the present greedy selection algorithm. The chromosomes stored in $\mathbf{G}^{temp}$ are used by succeeding operators to produce $\mathbf{G}^{n+1}$.

P Vector

The next phase of the GA process is chromosome modification. In the present implementation four modification operators are used—passthrough, random average crossover, perturbation mutation and mutation. The number of chromosomes modified with each operator is controlled by the P vector, which consists of four parameters—p_B, p_A, p_P, p_M. Each parameter controls one modification operator. The value of each parameter ranges from 0 to 1.0, and, for consistency, the sum of all four parameters must always equal one. A P vector equal to 0.1, 0.3, 0.3, 0.3, for example, will cause the first 10 percent of the chromosomes to be modified using the passthrough operator, the next 30 percent to be modified using random average crossover, the next 30 percent to be modified using perturbation mutation and the last 30 percent to be modified using mutation. That is, $p_B = 0.1$, $p_A = 0.3$, $p_P = 0.3$, and $p_M = 0.3$.

The passthrough operator is always performed first. After passthrough is complete, the implementation order of the remaining operators is immaterial. Once all values of $\mathbf{G}^{n+1}$ have been established, the algorithm proceeds to fitness evaluation, ranking and then onto succeeding generations until the optimization is sufficiently converged.

Passthrough

The simplest operator used in the present GA is "passthrough." As the name implies, a certain number of chromosomes with the highest ranks are simply "passed through" to the next generation from $\mathbf{G}^{temp}$ to $\mathbf{G}^{n+1}$ without modification. The passthrough operator is always performed first on the first chromosome in $\mathbf{G}^{temp}$, which is always the chromosome with the highest fitness. This guarantees that the maximum fitness never drops from generation to generation. The number of chromosomes that are passed through to the next generation is controlled by the first parameter in the P vector, p_B.

Random Average Crossover

The next GA operator is called the random average crossover operator and is implemented by first randomly selecting two chromosomes, $\mathbf{X}_{j1}^{temp}$ and $\mathbf{X}_{j2}^{temp}$, from $\mathbf{G}^{temp}$. Next, the two selected chromosomes are combined on a gene-by-gene basis using the following formula:

$$x_{i,j}^{n+1} = \frac{1}{2}(x_{i,j1}^{temp} + x_{i,j2}^{temp}) \quad i = 1,2,\cdots,N_G \tag{6}$$

where $x_{i,j}^{n+1}$ corresponds to the i^{th} gene in the j^{th} chromosome associated with $\mathbf{G}^{n+1}$ and $x_{i,j1}^{temp}$ and $x_{i,j2}^{temp}$ correspond to the i^{th} genes from the randomly chosen chromosomes $\mathbf{X}_{j1}^{temp}$ and $\mathbf{X}_{j2}^{temp}$. The number of chromosomes modified using the random average crossover operator is contolled by the parameter p_A—the second element in the P vector.

Perturbation Mutation

The next GA operator is called the perturbation mutation operator and is implemented by first selecting a random chromosome $\mathbf{X}_j^{temp}$ from $\mathbf{G}^{temp}$. Next, a probability test is performed for each gene $x_{i,j}^{temp}$ in the selected chromosome involving a call to the random number generator, $R(0,1)$, described above. If the random number is greater than p_1—a user-specified control parameter—the gene is not modified. If the random number is less than p_1 the gene is modified using

$$x_{i,j}^{n+1} = x_{i,j}^{temp} + (x_{\max_i} - x_{\min_i})[R(0,1) - 0.5]\beta \tag{7}$$

where β is a user-specified tolerance that controls the size of the perturbation mutation. The p_1 parameter is specified to statistically control the number of genes that are modified within a specific chromosome. Small values of p_1 result in the modification of few genes. Large values of p_1 result in the modification of many genes. For sensible results, the values of β and p_1 must lie between 0 and 1.0.

Because this operator can cause the value of a particular gene to exceed one of its constraints ($x_{\max_i}$ or $x_{\min_i}$), checks are required to make sure this does not happen. The number of chromosomes modified using the perturbation mutation operator is controlled by the parameter p_P—the third element in the P vector.

Mutation

The last GA operator used in the present study is called the mutation operator and is implemented similarly to the perturbation mutation operator. First, a random chromosome $\mathbf{X}_j^{temp}$ is chosen from $\mathbf{G}^{temp}$. Next, a probability test is performed for each gene $x_{i,j}^{temp}$ in the selected chromosome involving a call to the random number generator, $R(0,1)$. If the returned random number is greater than p_2—a user-specified control parameter—the gene is not modified. If the returned random number is less than p_2, the gene is given a completely different value using

$$x_{i,j}^{n+1} = (x_{\max_i} - x_{\min_i})R(0,1) + x_{\min_i} \tag{8}$$

The p_2 parameter is specified to statistically control the number of genes that are modified within a specific chromosome. Small values of p_2 result in the modification of few genes. Large values of p_2 result in the modification of many genes. For sensible results, p_2 must be between 0.0 and 1.0. The number of chromosomes modified using the mutation operator is determined by the parameter p_M—the fourth element in the P vector.

Computed Results

Model Hill-Climbing Problem

It is useful to use a simplified model problem to study the relative merits of different GA variations. In the present study, a multi-gene, multi-modal, hill-climbing problem is used. This model problem is given by

$$\left.\begin{array}{l} \left.\begin{array}{l} a_{m,k} = \sum_{i=1}^{N_G} (x_i - c_{i,m,k})^2 \\ b_{m,k} = h_{m,k} e^{\frac{-a_{m,k}}{N_G}} \end{array}\right\} \quad m = 1,2,\cdots,N_M \\ \\ z_k = \max\left\{b_{1,k},\cdots,b_{N_M,k}\right\} \end{array}\right\} \quad k = 1,2,\cdots,N_O \tag{9}$$

where the z's are the hill altitudes (design space objectives), the x's are the genes (design space decision variables—note that the j subscript has been omitted), the c's are free parameters, the h's are peak values for each hill or mode and the a, b quantities are intermediate results. The c and h parameter values can either be user input or specified via a random number generator. Once established for a particular problem, they do not change. The i subscript is the gene number and varies from 1 to N_G, the maximum number of genes. The m subscript is the mode number and varies from 1 to N_M, the maximum number of modes or hills in each design space. Finally, the k subscript is the objective number and varies from 1 to N_O, the maximum number of objectives. For the present study, $N_O = 1$, i.e., only single objective problems will be studied. In Eq. (9), the goal is to find values of the x's that will maximize the z values, and, of course, to do so without using any knowledge one might obtain by looking at Eq. (9). With the hill-climbing model presented in Eq. (9), the effect of N_G, N_M and N_O can be studied, either collectively or individually.

During the discussion of results, design space attributes such as "volume" will be mentioned. For design spaces with many dimensions, i.e., many genes, the concept of volume is not a precise one—"hyper-volume" being more appropriate. Even the concept of a "hill" in a design space with many dimensions is difficult to consider. In the present study terms such as "hill," "peak" or "volume" will be retained with the understanding that the "hyper-" counterparts are, in most cases, more appropriate.

Single-mode computations

Stochastic Characteristics of Genetic Algorithms—Genetic algorithms are stochastically-based search algorithms and, as such, produce results with statistical variation from case to case, even if the only quantity being varied is the initializing seed in the random number generator. An example of this is displayed in Fig. 1 where two GA convergence histories—maximum fitness error versus number of function evaluations—are compared. The number of function evaluations is used as a measure of computational work throughout this study (not including the computational work associated with GA algorithm overhead), because it is easy to define and because it does not vary from computer to computer. For most applications, the computational work associated with the GA optimization is easily dominated by function evaluation computation, and thus, the present results are useful in determining which GA parameter and design space attributes produce the most efficient computational results. The n^{th} generation error in the maximum fitness (E^n) plotted on the vertical axis in Fig. 1 is defined by

$$E^n = \frac{\left| f^n - f_{\max} \right|}{f_{\max}} \tag{10}$$

where f^n is the n^{th} generation maximum fitness, and $f_{\max}$ is the global maximum fitness for the model problem defined by Eq. (9). The parameter $f_{\max}$ is equal to

$$f_{\max} = \max\left\{h_{m,k}\right\}, \quad m = 1,\cdots,N_M \quad \text{and} \quad k = 1,\cdots,N_O$$

Both convergence histories in Fig. 1 are from the model problem defined by Eq. (9) and utilize design space parameter values given by $N_G = 32$, $N_M = 1$ and $R = 10.0$ and GA parameter values given by $N_C = 10$, $\beta = 0.01$, $p_1 = 0.2$, $p_2 = 0.05$ and $P = (0.1, 0.3, 0.3, 0.3)$. The single h value from Eq. (9) required for this problem is taken to be 100.0, and the c values are determined via random number generator using

$$c_{i,m,k} = \left[R(0,1) - \frac{1}{2} \right] \frac{R}{2} \tag{11}$$

where $R(0,1)$ is the previously defined random number generator. For each value of c a unique random value is generated using Eq. (11). With this problem setup the location of the global optimum is always within the gene limits. Once established, these h and c values are used for each computation in the single-mode ($N_M = 1$) section of this report.

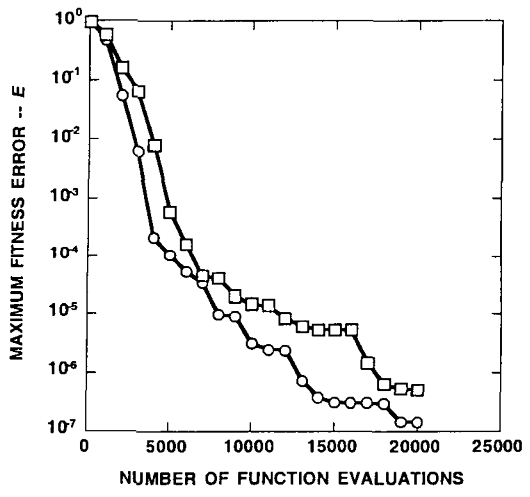


Fig. 1 Two arbitrary GA convergence histories utilizing different initial seed values with all other GA and design space parameters held fixed, $N_G = 32$, $N_M = 1$, $R = 10.0$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $p_1 = 0.2$ and $p_2 = 0.05$.

The above set of parameter values defines the so-called single-mode baseline solution for this study. The GA parameters utilized in this solution were determined via trial and error and represent an efficient set of GA parameters for the given design space definition, i.e., for the given values of R, N_G, N_M, h and c. All design space and GA parameters not being varied in the single-mode section of this report utilize the values that are established above.

Except for the seed value used to initialize the random number generator at the beginning of each GA convergence, all GA and problem parameters are the same for the two convergence histories displayed in Fig.1. But, as can be seen, the two convergence histories are different—in some locations by as much as 30-50%. These differences are caused by statistical differences encountered during solution execution and are typical for a GA search process. For multi-modal cases or cases with noise in the design space, the statistical variations can be much larger.

To study the relative effects of various GA or design space parameters on GA convergence, it is important to remove this statistical variation—at least most of it—so that the average effect of each parameter being studied can be ascertained. This is accomplished by running each case many times with different initializing seed values, and then averaging the results. The first step in this process is to determine how many solutions to use in the average, i.e., the NSEED value. An acceptable value for NSEED can be determined by re-computing a typical case many times with different initializing seed values, and then plotting the number of function evaluations required for GA convergence as a running or ensemble average.

Figure 2 displays such an exercise for the problem from Fig. 1. Running averages are produced for four specific convergence levels, $E=10^{-2}$, 10^{-3}, 10^{-4} and 10^{-5}. In this figure the error is the same as that defined by Eq. (10). Also plotted on the right are the running average asymptotes computed by setting the NSEED value to 1000. The difference between the asymptotic values and the running totals at NSEED = 30 for each of the error levels starting at $E=10^{-2}$ is 1.5%, 1.0%, 1.3% and 2.1%, respectively. As can be seen, sample-size independence is obtained at about 10-20 solutions for the larger levels of convergence error and at about 30 solutions for the smallest convergence error. It should be pointed out that this result is only valid for single-mode results, $N_M=1$. Multi-mode results require even larger NSEED values for sample-size independence. As a result of the information presented in Fig. 2, the sample size (NSEED value) for all single-mode computations presented in this study will be set to 30.

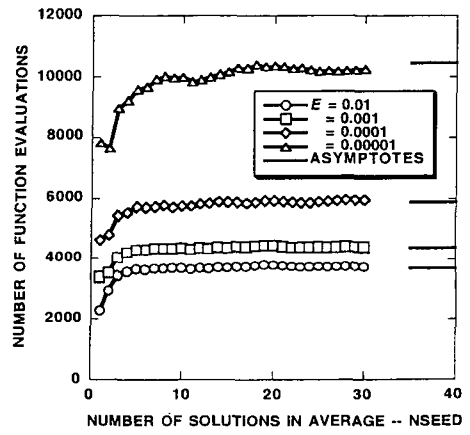


Fig. 2 Effect of running sample size average (NSEED value) on GA convergence to several different levels of convergence error, $N_G=32$, $N_M=1$, $R=10.0$, $N_C=10$, $P=(0.1,0.3,0.3,0.3)$, $\beta=0.01$, $\rho_1=0.2$ and $\rho_2=0.05$.

Effect of P vector on GA convergence—The effect of the P vector on GA convergence efficiency is presented in this section. As described above, the elements of the P vector—p_B, p_A, p_P, p_M—are used to control which operators will be used to modify the newly selected chromosomes at the beginning of each new generation. Figure 3 presents results showing the effect of this group of parameters on GA convergence. In keeping with many of the results in this section, GA convergence information is presented across a range of chromosome values—$N_C=10$, 20, 30, 40, 50, 60, 80, 100, 150, 200—for two different levels of GA convergence, $E=10^{-2}$ and 10^{-4}. The P vector notation used in Fig. 3, e.g.,

$P = 1333$, is shorthand for $P = (0.1, 0.3, 0.3, 0.3)$. The scale used on the vertical axis—always dedicated to the number of function evaluations required for GA convergence—will generally vary from 0 to 100,000 allowing cross comparison between many of the results presented in this section.

As can be seen from Fig. 3, the effect of the P vector on GA convergence efficiency—all other parameters held fixed—is small and ranges from a few tens of percent to a factor of two for the different P vectors studied. For each value of N_C and for both values of E displayed in Fig. 3, the optimal P vector is $P = (0.1, 0.3, 0.3, 0.3)$. The effect of N_C on GA convergence efficiency is also small. Regardless of which P vector from Fig. 3 is utilized, the number of chromosomes that optimizes GA performance is 10.

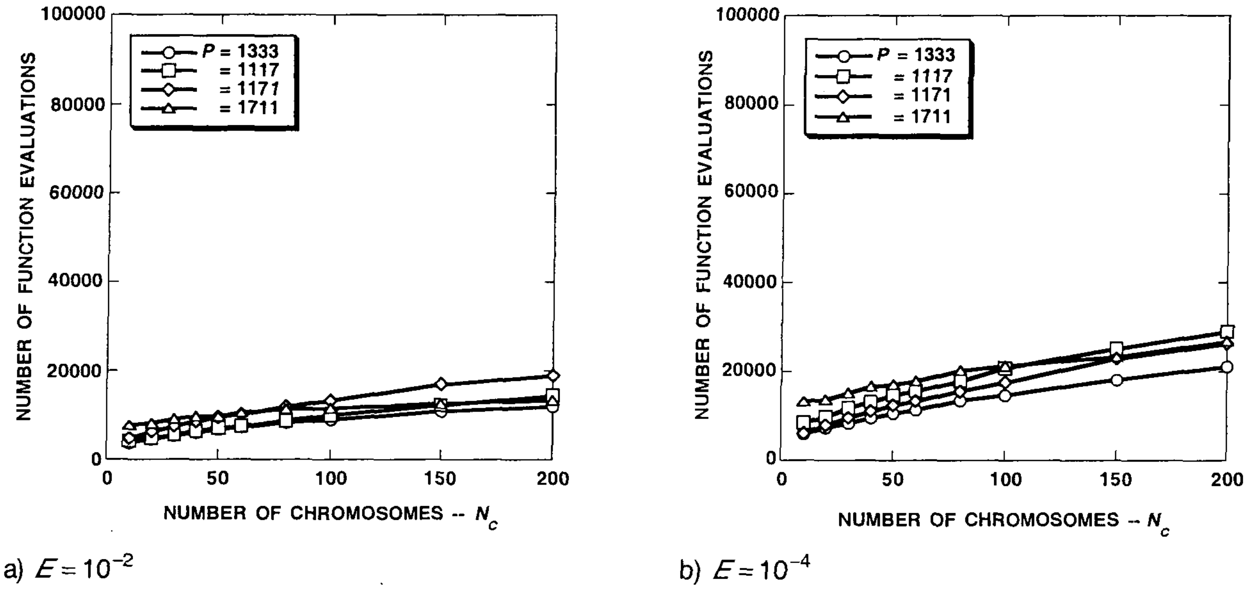


Fig. 3 Average number of function evaluations required to achieve GA convergence as a function of P and N_C, NSEED = 30, $R = 10.0$, $N_G = 32$, $N_M = 1$, $\beta = 0.01$, $p_1 = 0.2$ and $p_2 = 0.05$.

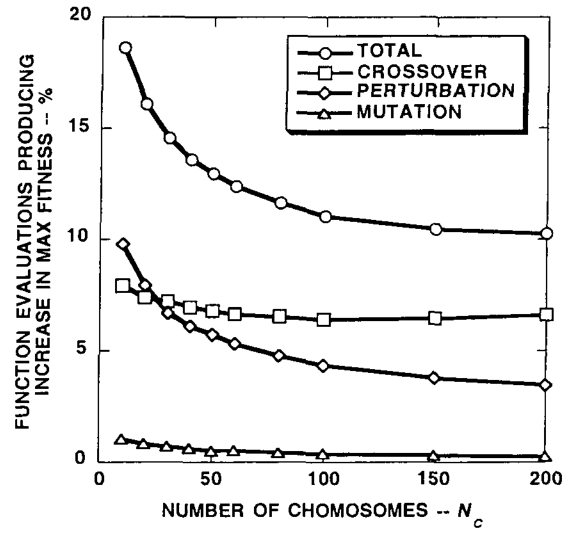
There is another facet to this situation that bears mention. In the present study all chromosomes are reevaluated with a new function evaluation each generation, including the passthrough chromosomes. This fact is reflected in all the function evaluation totals that are presented in this report. But the passthrough chromosomes never change from the old generation to the new generation. Thus, the old function evaluation is identical to the new one and would not have to be repeated. If this algorithm modification were included in the present implementation, the function evaluation totals would be reduced by 10%. This should be considered when comparing the present results to those outside this study.

Each operator controlled by the P vector performs a specialized function in the GA optimization process. To help understand these functions and how efficiently they are performed, a counter for each operator—passthrough, random average crossover, perturbation mutation and mutation—is incremented each time one of the chromosomes produced from that operator generates an increase in fitness over the previous generation's maximum fitness. Results from this statistical analysis are presented in Fig. 4 for the single-mode baseline problem described above. In addition to statistical results from individual operators, a curve labeled "TOTAL," which is the total number of function evaluations from all operators that generate an improvement in fitness value, is also included. As can be seen from the TOTAL curves the percentage of max-fitness-improving function evaluations decreases as the NC value increases. This is primarily manifested in decreasing efficiencies for the perturbation mutation operation, as the random average crossover operation efficiency is relatively constant.

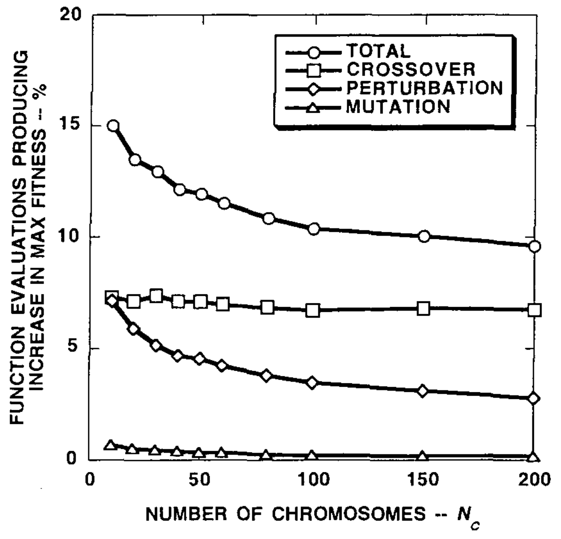
Another obvious result from Fig. 4 is that statistics for the passthrough operator are not included. This is due to the fact that this operator does not allow for chromosome modification and thus contains no mechanism for improving the fitness. Its job is to make sure that the best chromosome does not digress, a task that is amply served for single-objective optimizations when $p_B = 0.1$.

By comparing Figs. 4a and 4b, it can be seen that the max-fitness-improving percentages from all operators decrease somewhat from $E = 10^{-2}$ to $E = 10^{-4}$. Thus, GA efficiency decreases as the search process converges, which qualitatively agrees with the shape of the convergence histories displayed in Fig. 1.

Lastly, the effect of mutation on GA convergence is seen to be small, as the number of mutation function evaluations that improve the fitness is generally less than one percent. For the present problem, this operator is not important, but for many applications, especially those that are noisy or multi-modal in nature, mutation plays an important role, as will be seen later in this report.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

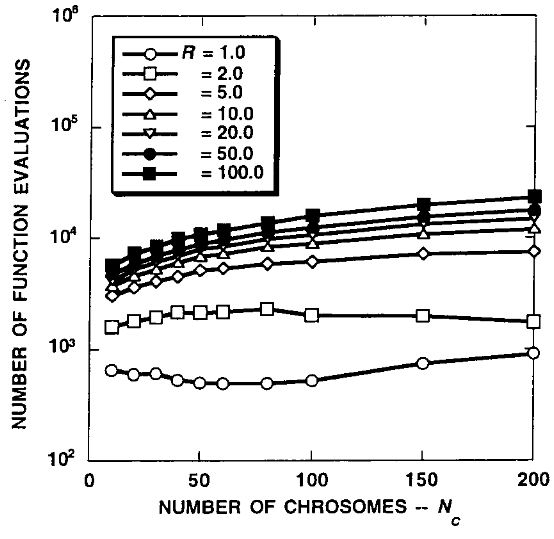
Fig. 4 Average number of function evaluations (in percent) producing an increase in the fitness relative to the previous generation maximum fitness for different values of N_C, NSEED = 30, $R = 10.0$, $N_G = 32$, $N_M = 1$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 0.05$.

Effect of design space size on GA convergence—Implementation of the GA described above requires the specification of upper and lower limits for each gene ($x_{\max_i}$ and $x_{\min_i}$). These user-specified limits (along with the value of N_G), in fact, define the size of the design space that is to be searched for the optimal solution. The specification of these gene limit parameters has a rather dramatic effect on GA operation. If the limits for each gene are set too far apart the design space will be unnecessarily large and GA convergence will be slowed. Conversely, if the limits do not include the problem's global optimum within their specified range, which can result if they are set too close together, convergence will be improved, but the final result will not be the desired global optimum. As a means of reducing the complication of this coefficient specification process, all of the $x_{\max_i}$ and $x_{\min_i}$ gene-limit values have been reduced to a single parameter, R, by utilizing Eqs. (4) and (11).

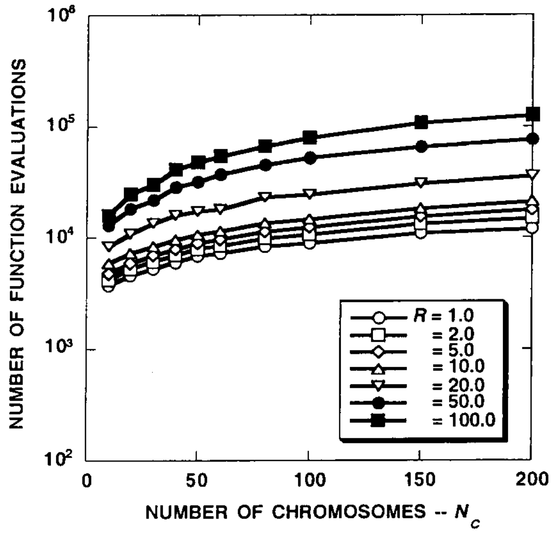
The purpose of this section is to study the effect of R on GA convergence. Results showing the effect of R on GA convergence efficiency are presented in Figs. 5 and 6. Figures 5a and 5b show the effect of R

in combination with N_C for two different levels of the convergence, $E = 10^{-2}$ and 10^{-4}. As readily seen for most values of R, the number of function evaluations required for convergence increases as the number of chromosomes increases. Generally, $N_C = 10$ provides optimal convergence for most situations being as much as a factor of four faster than the results at $N_C = 150$ or 200. More will be given in later sections describing the effects of population size (N_C) on GA convergence.

As expected, the average number of function evaluations required to obtain GA convergence increases with R, but the rate of increase is not as intuitive. From Figs. 5a and 5b it can be seen that the value of R increases by a factor of 100, but the number of function evaluations required for GA convergence generally increases by only a factor of 10—a little more than ten for all values of N_C in Fig. 5a—a little less than 10 for most of the results in Fig. 5b. For $N_C = 10$ and $E = 10^{-4}$, the increase in the number of function evaluations due to a 100-fold increase in R is only about 4.5.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

Fig. 5 Average number of function evaluations required to achieve GA convergence for different values of N_C and R, NSEED = 30, $N_G = 32$, $N_M = 1$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 0.05$.

A clearer picture of this behavior is displayed in Fig. 6 where the average number of function evaluations for GA convergence is plotted versus R for $NC = 10$. Results for eight different levels of GA convergence, ranging from $E = 10^{-1}$ to 10^{-8}, are included. From an analysis of the data in Fig. 6—with the proper set of curve fitting tools—the number of function evaluations (n) generally varies as a 1/3 power law in the R that is,

$$n \propto R^{1/3}$$

This relation becomes increasingly accurate for tighter levels of convergence, $E = 10^{-3}$ to 10^{-8}. For this set of E values the computed exponent actually varies from 0.30 to 0.39. This relation is also valid for larger values of N_C providing the exponent is adjusted to suitably higher values.

Of course, as R increases, the size of the design space increases. For 32 genes a doubling in the value of R from say 10.0 to 20.0—would cause the design space to increase by a factor of 2^{32} or 4 billion times! If a simple trial and error search were implemented for this problem, the search time would be proportional to the size of the design space. The GA search time, on the other hand—assuming optimal GA parameters—would increase by a factor of 1.26.

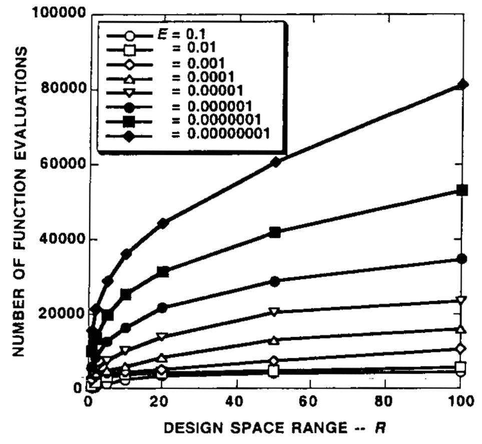
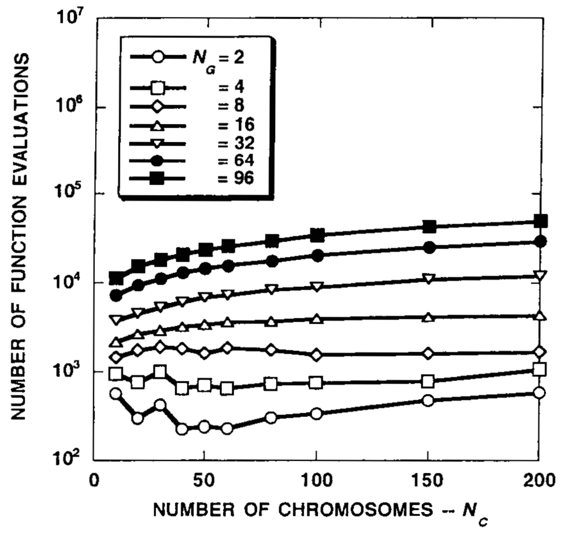
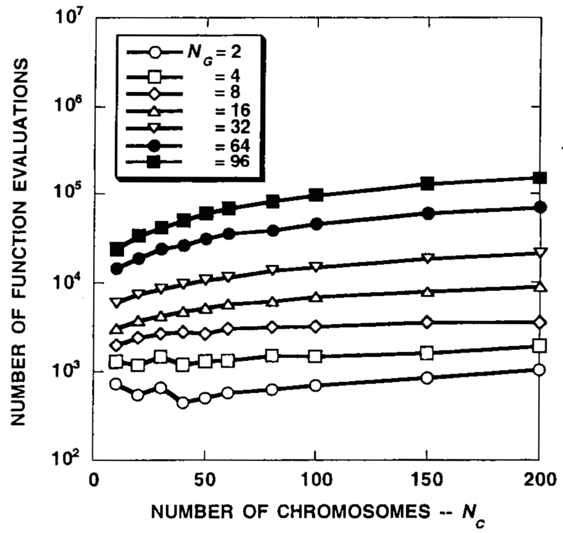


Fig. 6 Average number of function evaluations required to achieve GA convergence as a function of R and E, NSEED = 30, $N_G = 32$, $N_M = 1$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 0.05$.

Effect of number of genes on GA convergence—The effect of the number of genes utilized in defining the design space on GA convergence is studied next. Computed results for wide variations in the population size (N_C) and the number of design space genes (N_G) are presented in Fig. 7 for two different levels of GA convergence, $E = 10^{-2}$ and 10^{-4}. As can be seen, the optimal population size is reasonably independent of the specified level of convergence, but varies with the number of genes. For all but the smallest values of N_G, optimal GA convergence is generally achieved for a population size of 10, being as much as a factor of four faster than for the larger population sizes. For smaller values of NG the optimal population size is mixed, but generally resides around a values near fifty.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

Fig. 7 Average number of function evaluations required to achieve GA convergence as a function of N_C and N_G, NSEED = 30, $R = 10.0$, $N_M = 1$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 0.05$.

Figure 8 presents the average number of function evaluations required for GA convergence versus N_G for eight different convergence levels ranging from $E = 10^{-1}$ to 10^{-8}. The number of function evaluations grows linearly (approximately) with the number of genes, regardless of the specified convergence tolerance. Thus the following approximate relationship is established:

$$n \propto N_G$$

As can be seen from Fig. 8, GA convergence slows as smaller levels of error are achieved. This is in agreement with the convergence histories presented in Fig.1. Analysis of the data used to establish Fig. 8 shows that the rate of increase is a factor of 1.5 (approximately) for every decade change in the value of E. Thus, the following approximate relationship is established:

$$n \propto 1.5^{-\log E}$$

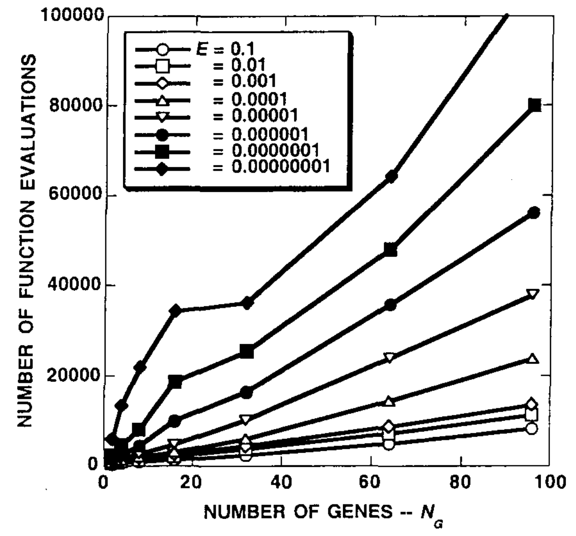
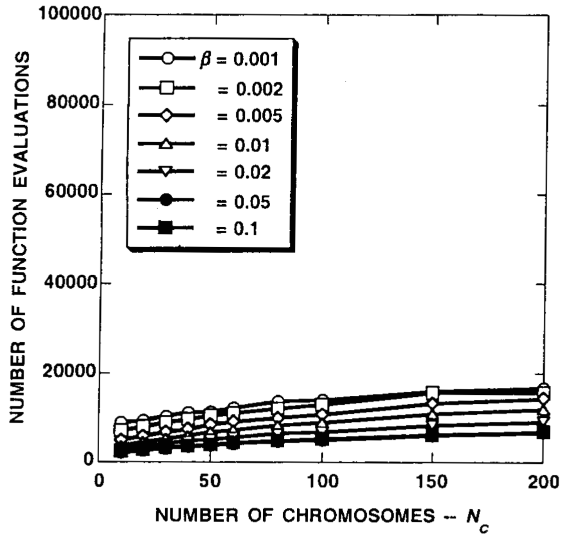


Fig. 8 Average number of function evaluations required to achieve GA convergence as a function of N_G and convergence error, NSEED = 30, $R = 10.0$, $N_M = 1$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 0.05$.

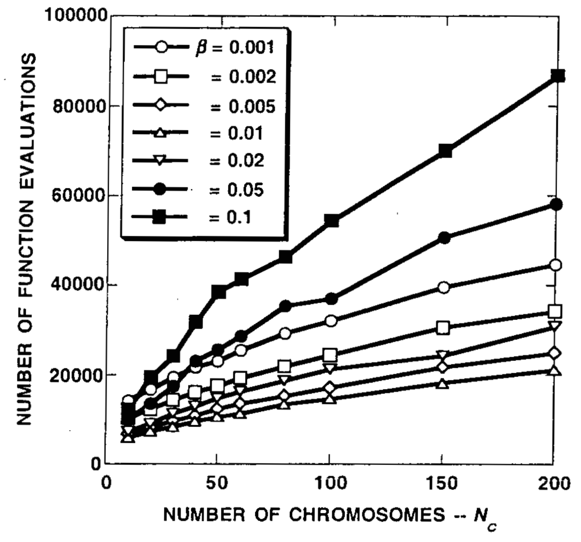
Effect of perturbation mutation on GA convergence—The effect of perturbation mutation parameters—β and ρ_1—on GA convergence efficiency is studied in this section. The β parameter is used to control the size of perturbations in the perturbation mutation operator, and the ρ_1 parameter is used to control the probability that any given gene will be perturbed. Figures 9-13 present results showing the effect of these parameters on GA convergence.

Figure 9 displays the average number of function evaluations required to achieve GA convergence as a function of N_C and β for two different values of the convergence, $E = 10^{-2}$ and 10^{-4}. The effect of β on GA convergence is moderate. The difference in number of function evaluations between the best and the worst values of β—all other parameters held fixed—is a factor of 2 to 4.

Generally speaking, the optimal value of β does not vary with N_C. This can be seen more clearly in Fig. 10 where the roles of N_C and β have been interchanged. As seen from Figs. 10a and 10b, the optimal values of β for $E = 10^{-2}$ and 10^{-4} are 0.05 and 0.01, respectively.

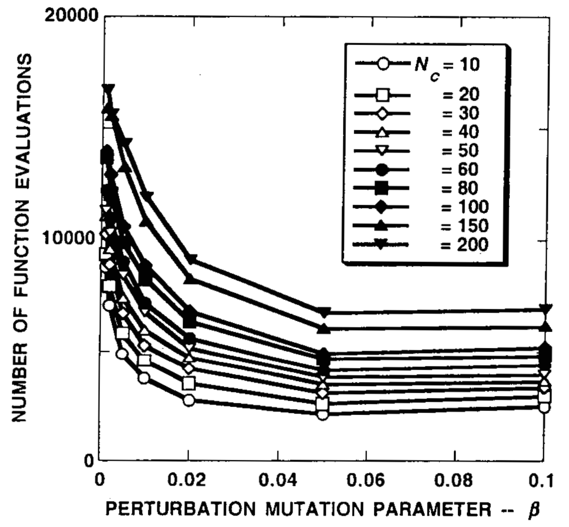


a) $E=10^{-2}$

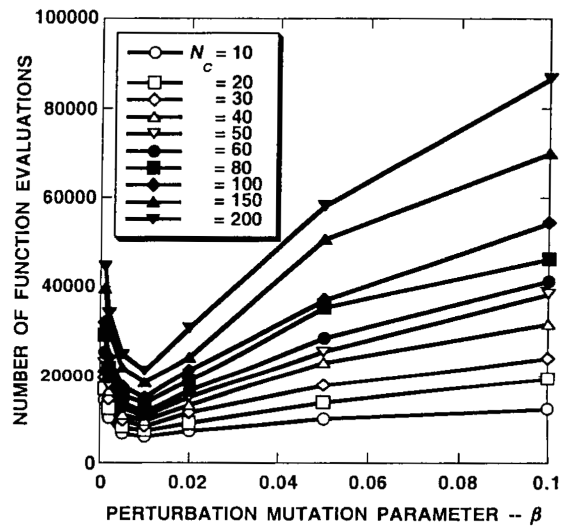


b) $E=10^{-4}$

Fig. 9 Average number of function evaluations required to achieve GA convergence as a function of N_C and β, NSEED = 30, R = 10.0, N_G = 32, N_M = 1, P = (0.1,0.3,0.3,0.3), ρ_1 = 0.2 and ρ_2 = 0.05.



a) $E=10^{-2}$



b) $E=10^{-4}$

Fig. 10 Average number of function evaluations required to achieve GA convergence as a function of β and N_C, NSEED = 30, R = 10.0, N_G = 32, N_M = 1, P = (0.1,0.3,0.3,0.3), ρ_1 = 0.2 and ρ_2 = 0.05.

But it is interesting to note that the optimal value of β for $E=10^{-2}$ is not the optimal value for $E=10^{-4}$. In fact, there is a dramatic reversal in behavior. The optimal value of β for $E=10^{-2}$ (β = 0.05) is one of the worst values for $E=10^{-4}$. This can be seen more clearly in Fig. 11, where the number of function evaluations required for GA convergence is plotted as a function of β for eight values of E ranging from 10^{-1} to 10^{-8}. The optimal value of β starts at 0.1 for $E=10^{-1}$ and falls to a value near 0.001 for $E=10^{-8}$. Thus, optimal GA convergence requires a correlation with the convergence level, E, that is sought.

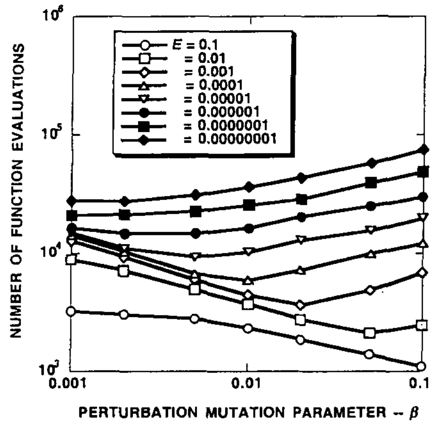
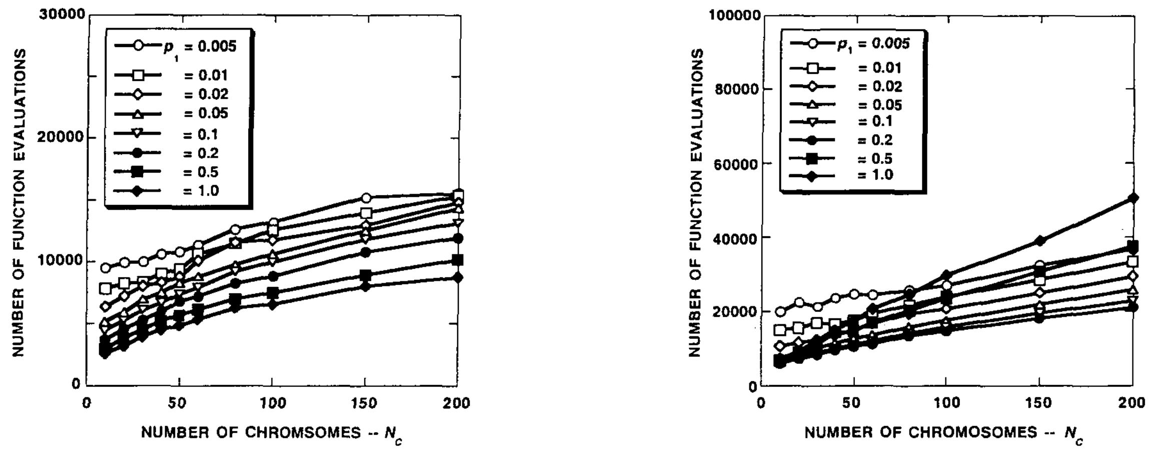


Fig. 11 Average number of function evaluations required to achieve GA convergence as a function of β and E, NSEED = 30, R = 10.0, N_G = 32, N_M = 1, N_C = 10, P = (0.1,0.3,0.3,0.3), p_1 = 0.2 and p_2 = 0.05.

The explanation for this behavior is straightforward. As the GA process converges the perturbations required to improve the solution become smaller, and thus, smaller values of β are more beneficial for tighter levels of GA convergence. The theoretical optimal would be to vary the value of β during GA convergence, starting with a larger value and automatically moving to smaller values as the GA process converges. A limited amount of work has been performed in this area with some success, but it is difficult for a general implementation as the evaluation of convergence error is generally unknown for practical applications. An "on the fly" statistical analysis of the perturbations that achieved improvement in the optimal fitness value might be a more successful alternative for choosing an optimal variation of β, but to date nothing along that line has been studied.

The perturbation mutation probability, p_1, is used to control whether a particular gene is modified by the perturbation mutation operator or not and was defined in conjunction with Eq. (7). The effect this parameter has on GA convergence is presented in Figs. 12 and 13. Figure 12 shows convergence information as a function of N_C and p_1 for two different levels of convergence, $E = 10^{-2}$ and 10^{-4}. As with other results seen in this section, N_C has a moderate effect on convergence for most values of p_1 with the smaller values of N_C always producing the best convergence.



a) $E = 10^{-2}$

b) $E = 10^{-4}$

Fig. 12 Average number of function evaluations required to achieve GA convergence as a function of p_1 and N_C, NSEED = 30, R = 10.0, N_G = 32, N_M = 1, P = (0.1,0.3,0.3,0.3), β = 0.01 and p_2 = 0.05.

The effect of p_1 on GA convergence is more complex. The optimal values of p_1 for $E=10^{-2}$ and 10^{-4} are 1.0 and 0.2, respectively. Thus, like the β results presented above, the optimal value of p_1 changes depending on the level of convergence. Figure 13 presents a global picture of this behavior. The optimal value of p_1 is 1.0 at $E=10^{-1}$. It decreases from that point to 0.1 at $E=10^{-5} \sim 10^{-6}$ and then increases again reaching a value of 1.0 at $E=10^{-8}$. The optimal regions for the $E=10^{-6}-10^{-8}$ curves are relatively flat, and thus, it is difficult to be precise about the optimal values or p_1 for smaller values of E.

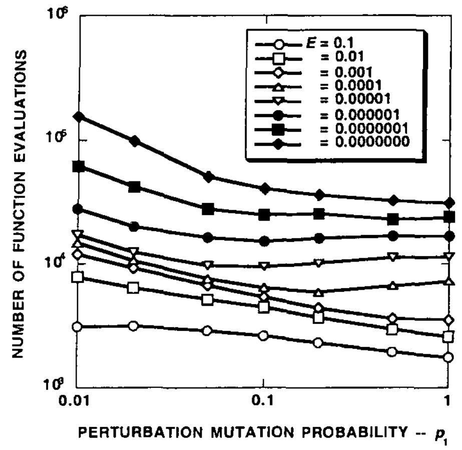
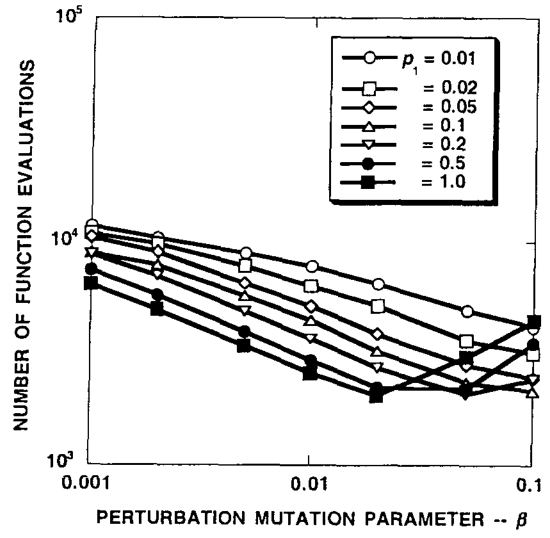


Fig. 13 Average number of function evaluations required to achieve GA convergence as a function of p_1 and E, NSEED = 30, R = 10.0, $N_G = 32$, $N_M = 1$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$ and $p_2 = 0.05$.

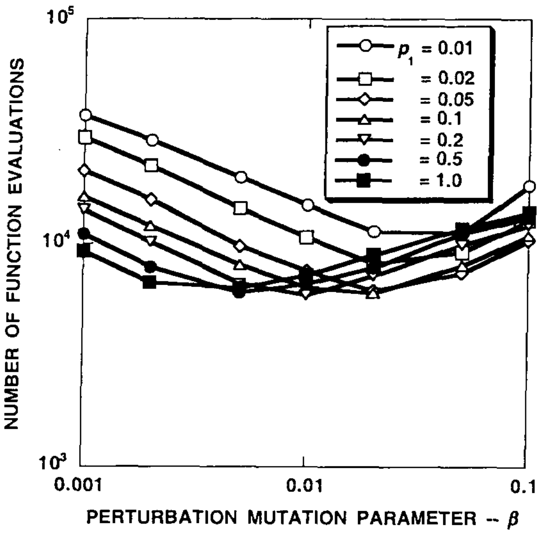
So far most of the results have involved the analysis of a single GA parameter over a suitable error or chromosome range while all the other parameters were held fixed at their baseline values. Figure 14 presents results where variations of both perturbation mutation parameters—β and p_1—are performed simultaneously for two values of convergence error—$E=10^{-2}$ and 10^{-4}. This allows a more complete analysis of GA convergence efficiency for non-optimal parameter values. For all computational results presented in Fig. 14, $N_C = 10$. From Fig. 14, it can be seen that the optimal values of β and p_1 occur at 0.02 and 1.0, respectively, for $E=10^{-2}$ and at 0.01 and 0.2 for $E=10^{-4}$. Generally speaking, midrange values of β and larger values of p_1 (for the ranges studied) typically produce acceptable convergence, while the results associated with the smallest values of β and p_1 are poor.

Effect of mutation on GA convergence—The effect of the mutation operator parameter—p_2—on GA convergence efficiency is studied in this section. The p_2 parameter is used to control the probability that any given gene will be mutated. Figures 15-16 present results showing the effect of this parameter on GA convergence. Figure 15 shows the average number of function evaluations required to achieve GA convergence as a function of p_2 in combination with N_C for two different values of the convergence tolerance, $E=10^{-2}$ and 10^{-4}. The effect of p_2 on GA convergence is moderate. The difference in number of function evaluations between the best and the worst values of p_2—all other parameters held fixed—is a factor of 1.5 to 3.

Note that results for $p_2 = 0.0$ are included. This value of the mutation probability does not allow any genes to be selected in the mutation operator. Thus, the chromosomes selected by the mutation operator become pure "passthrough" chromosomes, that is, they are unmodified from generation to generation. GA convergence is not as efficient for this value of p_2, but the process still converges with reasonable efficiency. This result strongly supports the observation made in conjunction with Fig. 4. The mutation operator is not important for single-mode optimization, i.e., it is not important for optimization problems involving pure "hill-climbing."



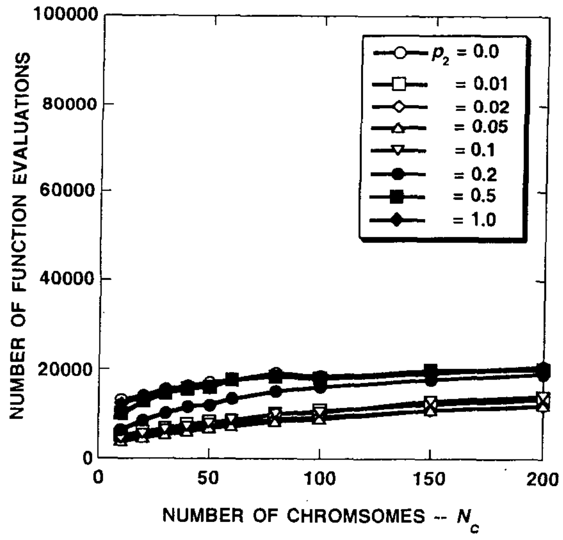
a) $E = 10^{-2}$



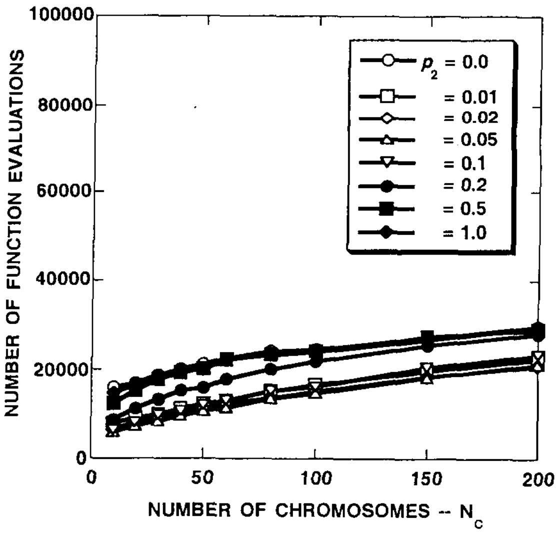
b) $E = 10^{-4}$

Fig. 14 Average number of function evaluations required to achieve GA convergence as a function of p_1 and β, NSEED = 30, R = 10.0, $N_G = 32$, $N_M = 1$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$ and $p_2 = 0.05$.

Generally speaking, the optimal value of p_2 does not vary with N_C. This behavior is similar to that of the β parameter (see Fig. 9). However, unlike the β parameter, the optimal value of p_2 does not vary from $E = 10^{-2}$ to $E = 10^{-4}$. It is 0.05 for both curves. The effect of p_2 over a broad range of convergence error can be seen in Fig. 16, where the number of function evaluations required for GA convergence is plotted versus p_2 for eight values of E ranging from 10^{-1} to 10^{-8}. The optimal value of p_2 starts at 0.05 for $E = 10^{-1}$ and remains relatively constant, increasing only slightly as the smaller values of E are approached.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

Fig. 15 Average number of function evaluations required to achieve GA convergence as a function of p_2 and N_C, NSEED = 30, R = 10.0, $N_G = 32$, $N_M = 1$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$ and $p_1 = 0.2$.

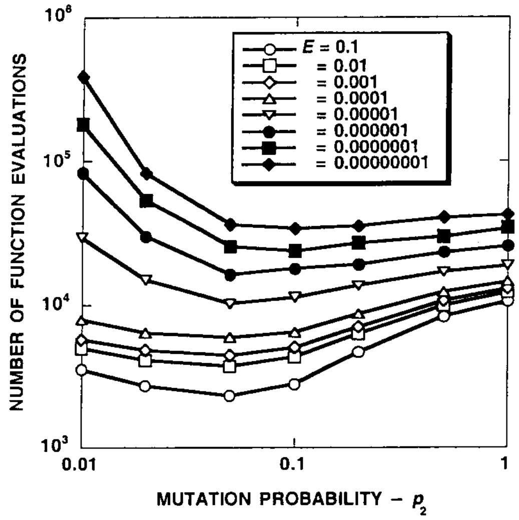


Fig. 16 Average number of function evaluations required to achieve GA convergence as a function of p_2 and E, NSEED = 30, R = 10.0, $N_G = 32$, $N_M = 1$, $N_C = 10$, $P = (0.1, 0.3, 0.3, 0.3)$, $\beta = 0.01$ and $p_1 = 0.2$.

Multi-modal computations

The results in this section focus on multi-modal design spaces—that is, design spaces with more than one, perhaps many, local extrema. Of course, one solution is always best—the so-called global optimum. The goal for any optimization algorithm is to find the global optimum without getting hung up by one or more of the local extrema. This is difficult for most gradient-based methods, especially as the number of modes in the design space increases.

For this section all computations utilize the model problem described by Eq. (9). The value of N_M, which was always equal to one in the previous section, will now be greater than one. The multi-modal characteristic creates a difficulty because there are so many new problem variables that must be specified and so many different ways in which to specify them. For example, for a design space with four modes and eight genes, i.e., $N_M = 4$ and $N_G = 8$, there are 32 independent c parameters—eight c's for each of the four modes—and 4 independent h parameters. Each requires a value to completely define the design space given by Eq. (9).

These values, while not being part of the GA, define the shape of the design space and thus, affect GA convergence. For example, a design space containing two modes with nearly identical peak values, one being the desired global optimum, represents a difficulty for any optimization procedure including the present GA approach. Once on the lower peak, it is difficult to move to the peak that contains the global optimum. As the two peak values approach each other, the number of function evaluations required for GA convergence, averaged over a suitably large sample of solutions, will approach infinity. Of course, if the two peaks are close, one might not care which peak is found, as either would be a suitable answer to the optimization problem. Although GA convergence performance as two or more peaks approach the global optimum will not specifically be examined in this study, GA performance for a wide variety of multi-modal scenarios will be presented.

Stochastic variations of multi-mode solutions—First, as was done at the beginning of the previous section, it is of interest to determine how much statistical variation exists in a typical GA convergence history—one that involves a multi-modal design space. This is accomplished by re-computing a specific case many times with different initializing seed values, and then plotting the number of function evaluations required for convergence as a running or ensemble average. Figure 17 displays such a plot for two different multi-mode cases that utilize the same design space parameters—RANGE = 5.0, $N_G = 8$ and $N_M = 8$. Each solution in Fig. 17 was continued until the convergence error—defined by Eq. (10)—was reduced below 10^{-4}. The eight h-values required for Eq. (9) are taken to be

$$h_1 = 100.0$$
$$h_2 = h_3 = \cdots = h_8 = 80.0$$

Note that the k subscript—the objective subscript from Eq. (9)—has been dropped from h for simplicity, as N_O is always one. The c values from Eq. (9) are determined via random number generator using Eq. (11). Once established, these h and c values are used for each re-computation for both cases.

The GA parameters utilized for the two convergence histories displayed in Fig. 17 are different. Case A consists of $N_C = 10$, $P = (0.1, 0.2, 0.2, 0.5)$, $\beta = 0.01$ $\rho_1 = 0.2$ and $\rho_2 = 1.0$, and Case B consists of $N_C = 200$, $P = (0.1, 0.7, 0.1, 0.1)$, $\beta = 0.01$ $\rho_1 = 0.2$ and $\rho_2 = 1.0$. Note that only the P vector and number of chromosomes in the fixed population size are different.

Case A produces nearly optimal convergence, at least for this definition of the design space, and will be used as the multi-mode baseline for this section. All design space and GA parameters not being varied in the multi-mode section of this report utilize the values that are established in this baseline solution. Case B is far from optimal and is close to a worst case scenario, for this design space configuration.

As can be seen from Fig. 17, the running average is plagued with oscillations. Each time the GA becomes "stuck" on one of the design space's lesser peaks, the running average jumps. Each time convergence "finds" the global optimum quickly the running average drops. As the running average continues to higher NSEED values, each curve becomes smoother, especially case A—the baseline solution. The two solid lines to the right of each curve indicate the maximum and minimum extent of each running average from NSEED = 101 to 1000, and the dashed curve is the final value at NSEED = 1000. The total variation beyond an NSEED value of 100 for Case A is –8.4% to +14.8%, and for Case B it is –10.7% to +35.0%.

It is easy to see that multi-modal cases involve more statistical variation than single-mode cases. As such, more averaging is required to obtain sensible results. Thus, for each solution in this section the minimum NSEED value will be 100. Even then, a statistical variation of at least 10 percent is typical and several tens of percent possible.

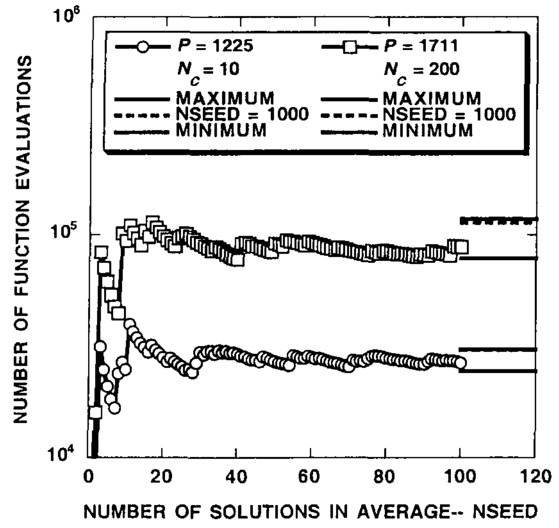
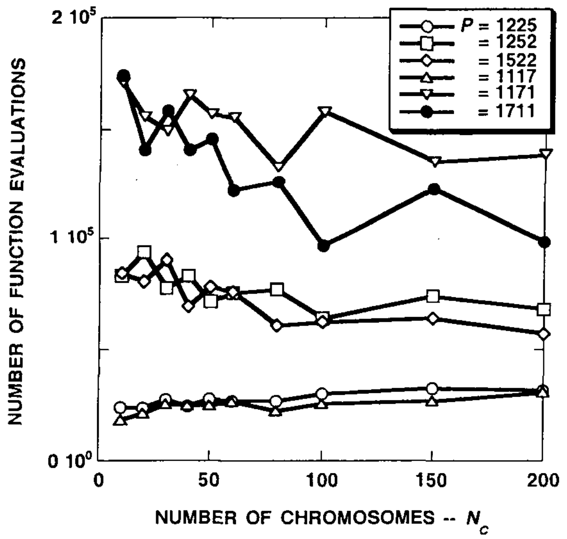


Fig. 17 Effect of running sample size average (NSEED value) on GA convergence for two different multi-modal solutions, $E = 10^{-4}$, $R = 5.0$, $N_G = 8$, $N_M = 8$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 1.0$.

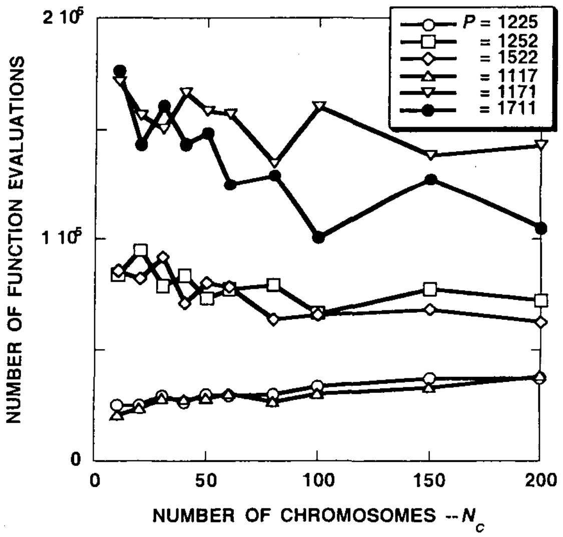
Effect of P vector on multi-modal GA convergence—Figure 18 presents results showing the effect of the P vector on GA convergence for the multi-mode conditions. Because some the results associated with Fig. 18 have more statistical variation than other multi-mode sections, an NSEED value of 200 is used. In keeping with other results in this study, GA convergence information is presented across a range of chromosome values—$N_C = 10$, 20, 30, 40, 50, 60, 80, 100, 150, 200—for two different levels of convergence, $E = 10^{-2}$ and 10^{-4}.

A quick comparison of Figs. 18a and 18b suggests that they are the same, but a more careful examination indicates a slight difference. Each data point in Fig. 18b is slightly higher than the corresponding point in Fig. 18a, indicating (as expected) more work required to attain the tighter level of convergence. The reason they are close is simply that most function evaluations for the present multi-mode computations are used to find the proper peak in the design space. Once that is accomplished climbing to the top of that peak requires only a small fraction of the total computational effort. Thus, there is not much difference in the work required to achieve $E = 10^{-2}$ relative to that of $E = 10^{-4}$. For that matter there is not much difference in the work required to achieve any two values of E (for sufficiently complex multi-modal conditions) providing they both require being on the "primary hill" in the design space.

As can be seen from Fig. 18, the effect of the P vector on GA convergence efficiency for the present multi-modal case—all other parameters being held fixed—is large. The most efficient cases converge 5 to 10 time faster than the least efficient cases. Having at least 50% mutation, i.e., $p_M \geq 0.5$, is critical for good GA convergence efficiency. In particular, the best convergence efficiency from Fig. 18 corresponds to $P = 1225$ and 1117. This behavior is in dramatic contrast with the single-mode results displayed in Fig. 3 where the P vector had little effect on convergence. Although there is little variation in GA efficiency across the N_C range, relatively speaking, the optimal value of N_C for the most efficient high-mutation P vectors is 10 and between 100 and 200 for the less efficient low-mutation P vectors. The latter observation for the low-mutation P vectors is tentative at best, as these results are noisy and thus, less amenable to trend analysis.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

Fig. 18 Average number of function evaluations required to achieve GA convergence as a function of the P vector and N_C, NSEED = 200, $R = 5.0$, $N_G = 8$, $N_M = 8$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 1.0$.

Figure 19 presents a statistical analysis for how often each P vector operator produced a chromosome with a fitness that exceeded the previous generation's maximum fitness value. These results are for the multi-modal baseline solution. Results for two different levels of convergence are included—$E = 10^{-2}$ and 10^{-4}. In addition to statistical results from individual operators, a curve labeled "TOTAL"—the total number of function evaluations from all operators that produced an improvement in maximum fitness value—is also included.

As can be seen from Fig. 19, N_C has only a small effect on each of these curves. Crossover and perturbation are equally effective (approximately) in producing improvement to the maximum fitness, but mutation (seemingly) is a disappointment, as it barely registers on the plot.

Further insight into the present results can be obtained by comparing the multi-modal results of Fig. 19 with the single-mode results of Fig. 4. Because these two problems are quite different, a quantitative comparison is not possible. Nevertheless, interesting contrasts between the two sets of results can be drawn. First of all, the success rate of all operators is dramatically lower for the multi-modal computations, especially the mutation operator. Crossover and perturbation mutation are reduced in efficiency by a factor ranging from 3 to 8, but mutation is reduced in efficiency by a factor of 50 to 120. This is a direct result of the GA search becoming "stalled" on one or more of the lower peaks—potentially for large numbers of generations—before finding its way to the global optimum. The mechanism by which the GA moves from a local optimum to the global optimum is mutation. For this "peak jumping" operation the mutation operator is best served by setting ρ_2 to one. (The numerical demonstration of this will be presented in a later section). This is because, for peak jumping, all gene values, in general, need to be changed simultaneously, and $\rho_2 = 1.0$ forces this to be the case. Because of this, the ability of mutation to produce fitness improvements in the hill-climbing part of the GA process is dramatically reduced. Thus, it serves its role in jumping from one peak to another but does little else for multi-modal search spaces.

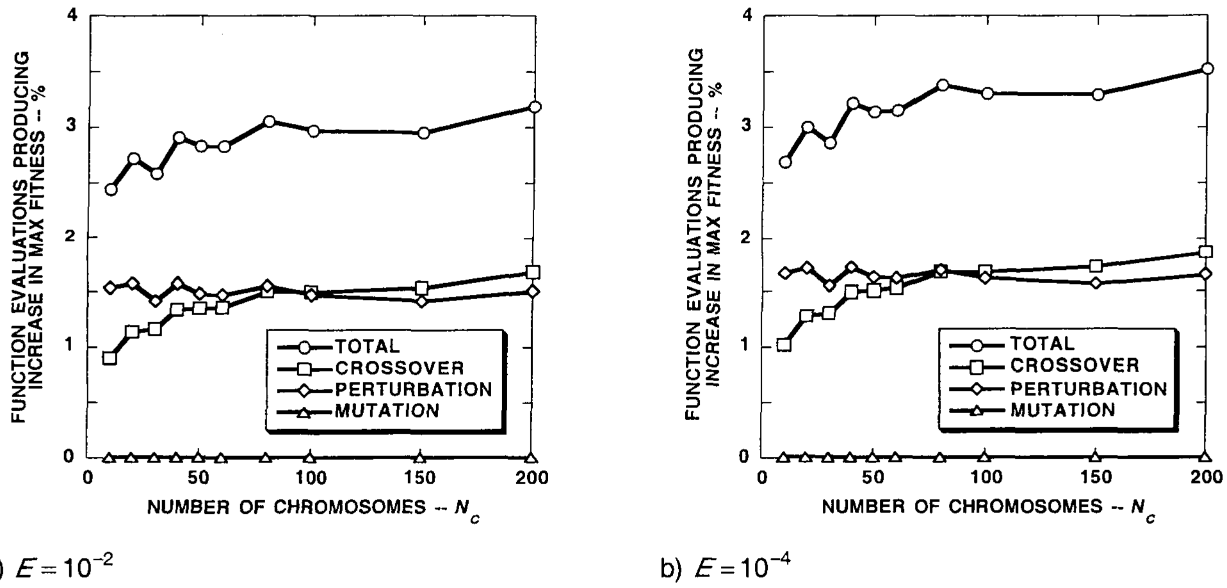
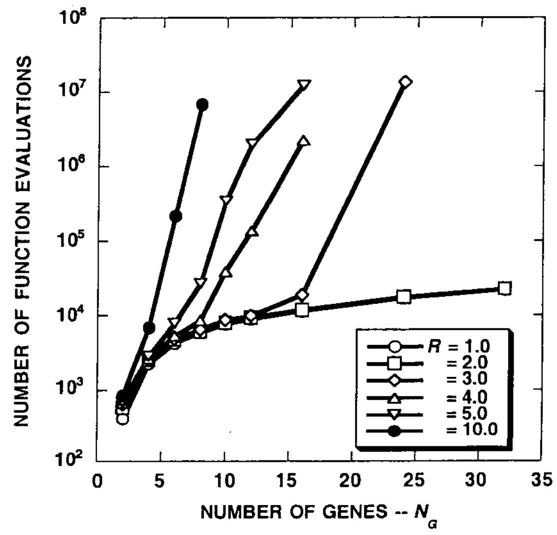


Fig. 19 Average number of function evaluations producing and increase in the fitness relative to the previous generation maximum fitness for different values of N_C, NSEED = 200, $R = 5.0$, $N_G = 8$, $N_M = 8$, $P = (0.1, 0.2, 0.2, 0.5)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 1.0$.

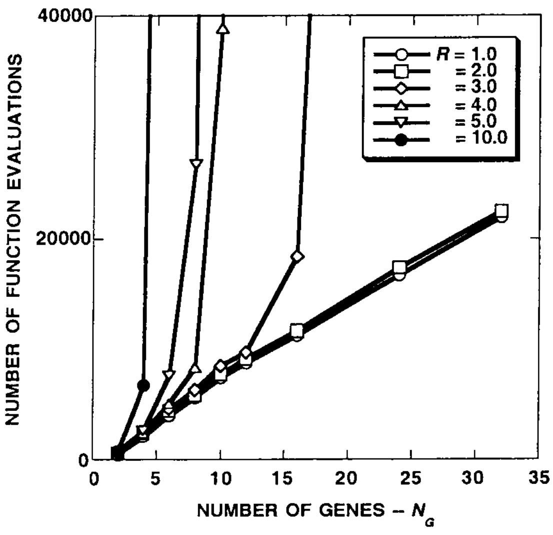
Effect of design space on GA convergence for multi-modal applications—The effect of design space size—R and N_G—on GA convergence efficiency is studied next. Figure 20 presents the effect of these two parameters on GA convergence performance. Note that Fig. 20a utilizes a full-extent semi-log scale on the vertical axis, and Fig. 20b utilizes a truncated linear scale on the vertical axis. Otherwise these two figures are the same. All computations were averaged over 100 solutions (NSEED = 100) each being continued until $E = 10^{-4}$ was achieved.

As can be seen from Fig. 20a, the combination of large values of R and large numbers of genes creates a difficulty for the present GA optimization process. The GA always converges, but the number of function evaluations becomes quite large. This, of course, is not unexpected. As the size and dimensionality of any multi-modal optimization problem increases, any optimization approach will become bogged down. This is the co-called "curse of dimensionality."

Nevertheless, in examining the results of Fig. 20 further, it is interesting to note that the curves corresponding to the first two values of R—1.0 and 2.0—are essentially linear (see Fig. 20b), increasing by approximately 700 function evaluations for each additional gene added to the design space. The other curves seem to follow this linear trend until a critical value of N_G is reached after which they increase exponentially. The critical value of N_G becomes smaller as the R is increased. This indicates the importance of setting the proper values for $xmin_i$ and $xmax_i$ in any GA optimization problem. As seen in the previous section, this was important for single mode computations. It is much more important for multi-mode computations.



a) Log scale along vertical axis (full extent)



b) Linear scale along vertical axis (truncated)

Fig. 20 Average number of function evaluations required to achieve GA convergence for different values of N_G and R, $E = 10^{-4}$, NSEED = 100, $N_M = 8$, $P = (0.1, 0.2, 0.2, 0.5)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 1.0$.

The effect of design space modality on GA convergence efficiency—i.e., the number and relative sizes of each of the modes—is studied next. Of course, there are literally an infinite number of design space variations that can be studied in this area. Using the model problem of Eq. (9) alone and fixing N_G and R, the variations in the values of h and c create many possibilities that can affect the functioning of an optimization scheme's convergence.

To facilitate the necessary reduction of possibilities, it is assumed that the single global peak—$h_{\max}$—utilized in Eq. (9), will be fixed at100.0 and that the remaining values of $h_{m,k}$, no matter what value of N_M is utilized, will be equal to each other. With this assumption a new parameter—D—is defined as

$$D = \frac{h_{\max} - h}{h_{\max}} \tag{12}$$

where h is the constant height of all secondary peaks in the design space.

Figure 21 presents the effect of D and N_M on GA convergence performance. All computations were averaged over 100 solutions (NSEED = 100) each being continued until $E = 10^{-4}$ was achieved. As can be seen from Fig. 21, there is little variation in GA convergence efficiency with N_M. At most a factor of 3 or 4 increase in the average number of function evaluations for GA convergence is seen as N_M increases from 2 to 64—and this for only the smaller values of D. But the variation of GA convergence efficiency with D—all other parameters held fixed—is more dramatic, being two orders of magnitude for the values of D studied. The reason for this behavior—discussed at the beginning of this section—is due to the inefficiencies associated with "peak-jumping" as opposed to "hill-climbing." As D becomes small the peak-jumping difficulties increase. The GA optimization never fails to converge, but the average number of function evaluations required for convergence increases dramatically.

In is interesting to note that the curves in Fig. 21 corresponding to $D = 0.5$ and 0.8 are nearly flat as N_M increases. In fact the convergence history for any case with these values of D and $N_M > 2$ is essentially the same as a single mode case with $N_M = 1$—assuming all other parameters are also the same. For these values of D the design space is more representative of a single mode design space with superimposed noise—the higher the value of N_M, the higher the noise frequency. With this interpretation for the $D = 0.5$ and 0.8 curves presented in Fig. 21, it can be concluded that the present GA convergence efficiency does not degrade in the presence of noise at least for low to moderate frequencies.

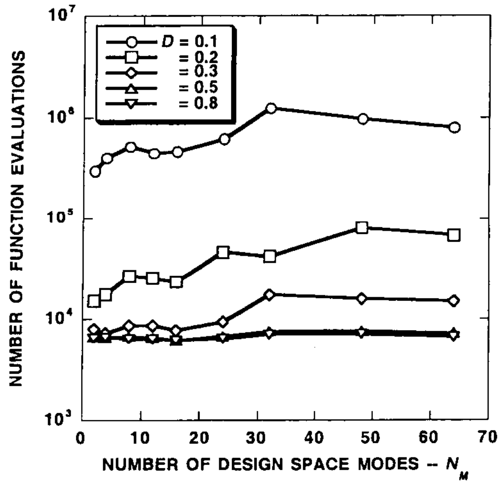
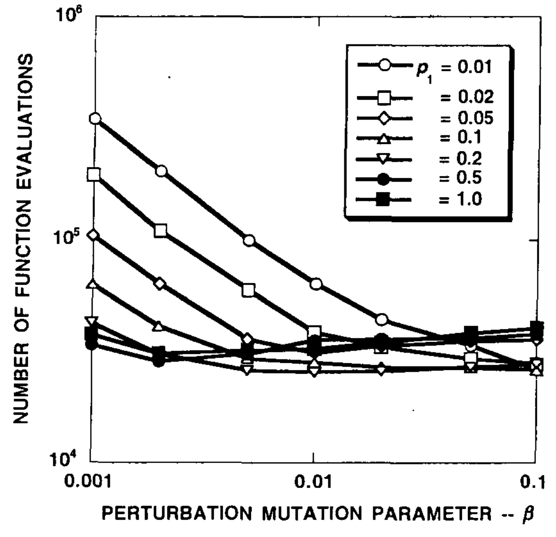


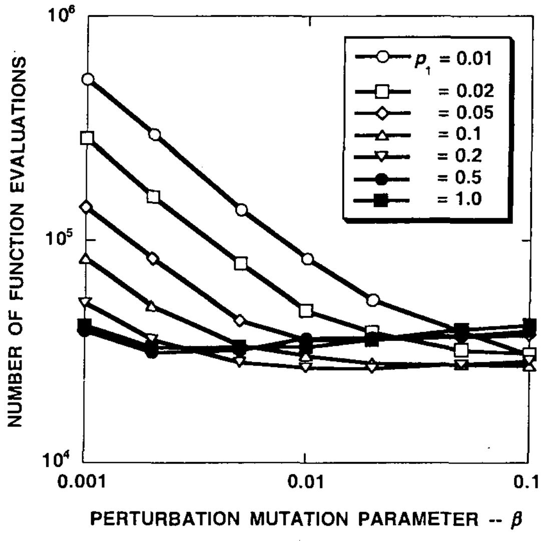
Fig. 21 Average number of function evaluations required to achieve GA convergence for different values of N_M and D, $E = 10^{-4}$, NSEED = 100, $R = 5.0$, $N_G = 8$, $P = (0.1, 0.2, 0.2, 0.5)$, $\beta = 0.01$, $\rho_1 = 0.2$ and $\rho_2 = 1.0$.

<u>Effect of perturbation mutation on GA convergence for multi-modal applications</u>—The effect of the perturbation mutation parameters—β and ρ_1—on GA convergence efficiency for multi-modal applications is studied in this section. Figure 22 presents results showing the effect of these parameters on GA

convergence for two levels of error, $E = 10^{-2}$ and 10^{-4}. In each case all averages are performed with NSEED = 100.



a) $E = 10^{-2}$



b) $E = 10^{-4}$

Fig. 22 Average number of function evaluations required to achieve GA convergence for different values of β and p_1, NSEED = 100, R = 5.0, N_C = 10, N_G = 8, N_M = 8, P = (0.1,0.2,0.2,0.5) and p_2 = 1.0.

As can be seen from Fig. 22, the optimal value of β is 0.01, and the optimal value of p_1 is 0.2. These optimal values hold for both levels of convergence and are the same as the $E = 10^{-4}$ results reported for the single mode case (see Fig. 14). The general trends presented in Fig. 22 also agree reasonably well with the results of Fig. 14, especially Fig. 14b. However, the results presented in Fig. 22 generally require an order of magnitude more function evaluations for convergence than those in Fig. 14. Figure 14 represents a pure hill-climbing process, while Fig. 22 represents a combined hill-climbing and peak-jumping process. As with Fig. 14, Fig. 22 suggests that small values of β in conjunction with small values of p_1—for the ranges studied—produce particularly poor convergence efficiency and should be avoided.

Effect of mutation on GA convergence for multi-modal applications—The effect of the mutation operator—the p_2 parameter—on GA convergence efficiency for multi-modal applications is studied in this section. Figure 23 presents results showing the effect of this parameter on GA convergence across a range of N_C values for a convergence error of $E = 10^{-4}$. (The curves for all other E levels—above a value of 0.2—virtually over plot the corresponding curves displayed in Fig. 23). Due to increased levels of statistical variation, especially for smaller values of p_2, NSEED = 200.

As can be seen from Fig. 23 the mutation operator has a profound effect on GA convergence efficiency for the present multi-modal problem. As the value of p_2 is decreased, GA convergence efficiency drops dramatically. As was seen from Fig. 19, the mutation operator does not often produce new chromosomes that improve the fitness relative to the previous generation's maximum fitness. But the small number of times that it does, for multi-modal problems, is crucial in the "peak-jumping" process. When the value of p_2 is reduced, the mutation operator is not nearly as efficient in peak-jumping. The number of GA function evaluations goes up as the GA process becomes "stuck" on local optima for larger numbers of function evaluations. This is because, in general, all gene values must be changed to jump from a local optimum to the global optimum and that is best achieved when p_2 = 1.0.

The variation of GA convergence efficiency with N_C is small, especially relative to the variations caused by p_2. The noise in these results—especially for the smaller values of p_2—make additional specific observations in this area difficult.

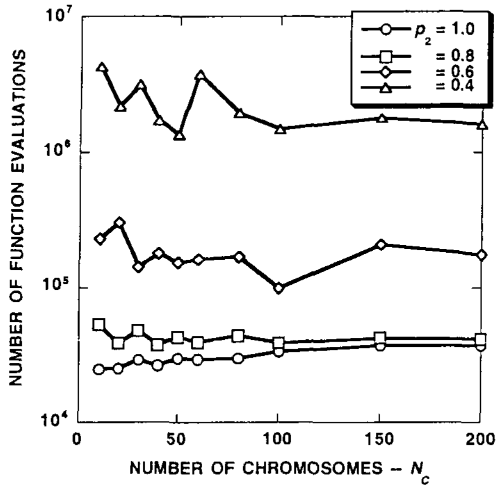


Fig. 23 Average number of function evaluations required to achieve GA convergence for different values of N_C and p_2, $E=10^{-4}$, NSEED=200, $R=5.0$, $N_G=8$, $N_M=8$, $P=(0.1,0.2,0.2,0.5)$, $\beta=0.01$ and $p_1=0.2$.

Conclusions

A genetic algorithm (GA) procedure suitable for performing engineering optimizations has been presented. It uses real-number encoding to represent all design space decision variables as genes and populations of fixed size to go from generation to generation. Four gene modification operators are utilized to advance from one generation to the next. They include passthrough, random average crossover, perturbation mutation and mutation.

The GA optimization procedure converged to the global optimum for every case attempted, demonstrating robustness and wide applicability. In some cases convergence was achieved quickly, while in other cases convergence was much slower. A systematic study exploring the effects of design space attributes and GA parameter selection on GA convergence performance was conducted with the following specific conclusions:

Increasing the design space size, including the number of genes, N_G, and the distance between gene maximum and minimum values, R, diminished GA convergence performance and made the search for an optimum more difficult and more costly. In addition, for multi-modal applications a large number of modes in conjunction with small values of D—the normalized difference between the global and local optimal peak values—decreased GA performance. Cases involving moderately large values of D (~ 0.8 to 0.5) showed almost no degradation in performance, even as the number of modes increased to a large value ($N_M \sim 64$). These conditions are akin to noise (of low to moderate frequency) and suggest that the present GA optimization procedure is reasonably unaffected by design space noise.

The above conclusion, which states that GA convergence slows for increasing design space size, is reasonably intuitive. A more difficult less intuitive aspect to assess is the effect of algorithm parameters on GA convergence performance. The conclusions in this area are presented by looking at each GA parameter separately.

The number of chromosomes used in each generation, N_C, had a surprisingly small effect on GA convergence efficiency for most situations encountered, especially relative to other parameters studied. The optimal value was almost always the smallest value studied—$N_C = 10$. A small number of cases generally utilizing non-optimal GA parameters or lower levels of GA convergence, yielded a mid-range optimal value for the number of chromosomes—$N_C \sim 50$.

The four-element P vector controls which operators are used to modify the selected chromosomes for each new generation. P vector variations produced little change in GA convergence efficiency for single mode cases, but large changes in efficiency for multi-mode cases. The first operator, passthrough, keeps the maximum fitness from degrading by passing the chromosome with the highest fitness through to the next generation without modification. The second and third P vector operators—random average crossover and perturbation mutation—create generally small variations about the higher ranked chromosomes and are the primary mechanism for "hill climbing" in the GA convergence process. Mutation, the final P vector operator, is not important for single-mode applications, but is very important for multi-mode applications, being the chief mechanism for changing from one design space hill to another—i.e., "hill jumping."

The perturbation mutation parameters—β and p_1—had optimal values of 0.01 and 0.2 (respectively) for most cases tested, both single and multi-mode problems, providing nominal values of the GA convergence error were used—$E \sim 10^{-4}$. However, optimal values for both of these parameters, especially β, exhibited a considerable dependence on convergence error. For example, the optimal value of β ranged from 0.1 for $E = 10^{-1}$ to 0.001 for $E = 10^{-8}$ (single mode cases). As the hill-climbing part of the optimization process converges, smaller perturbations—smaller values of β—are required for optimal convergence.

Of all GA parameters, mutation probability, p_2, had the most dramatic effect on GA convergence efficiency. For single-mode cases the optimal value was 0.05—a value that did not vary significantly with other GA parameters or the level of convergence error requested. For multi-mode cases the optimal value of p_2 was 1.0. Any value of p_2 slightly smaller than 1.0, for multi-mode cases, produced significant degradations in GA convergence performance.

The purpose of this report has been to study the convergence efficiency of Genetic Algorithms when used for single-objective optimization. Part II of this report will look at the same topic in conjunction with multi-objective problems.

References

1. Goldberg, D. E., "*Genetic Algorithms in Search, Optimization and Machine Learning,*" Addison-Wesley, Reading, MA, 59-88, 1989.

2. Davis, L., "*Handbook of Genetic Algorithms,*" Van Nostrand Reinhold, New York, 1991.

3. Beasley, D., Bull, D. R. and Martin, R. R., "An Overview of Genetic Algorithms: Part 1, Fundamentals," *University Computing*, Vol. 15, No. 2., 1993, pp. 58-69.

4. Beasley, D., Bull, D. R. and Martin, R. R., "An Overview of Genetic Algorithms: Part 2, Research Topics," *University Computing*, Vol. 15, No. 4., 1993, pp. 170-181.

5. Deb, Kalyanmoy, "Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems," *Evolutionary Computation*, Vol. 7, No. 3, 1999, pp. 205-230.

6. Van Veldhuizen, David and Lamont, Gary, "Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art," *Evolutionary Computation*, Vol. 8, No. 2, 1999, pp. 125-147.

7. Jiménez, José, Cuesta, Pedro and Abderramán, Jesús, "Mixed Strategy in Genetic Algorithms: Domain's Reduction and Multirecombination," *European Congress on Computational Methods in Applied Sciences and Engineering*, ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

8. Obayashi, S. and Tsukahara, T., "Comparison of Optimization Algorithms for Aerodynamic Shape Design," *AIAA J.*, Vol. 35, 1997, pp. 1413-1415.

9. Bock, K.-W., "Aerodynamic Design by Optimization," Paper 20, AGARD CP-463, 1990.

10. Pulliam, Thomas H., Nemec, Marian, Holst, Terry L. and Zingg, David W., "Comparison of Genetic and Adjoint Methods for Multi-Objective Viscous Airfoil Optimizations," Accepted for presentation at the AIAA 41st Aerospace Sciences Meeting, Reno, NV, Jan. 2003, AIAA Paper No. 2003-0298.

11. Sharatchandra, M., Sen, M. and Gad-el-Hak, M., "New Approach to Constrained Shape Optimization Using Genetic Algorithms," *AIAA J.,* Vol. 36, No. 1, Jan. 1998, pp. 51-61.

12. Gage, P. and Kroo, I., "A Role for Genetic Algorithms in a Preliminary Design Environment," AIAA Paper No. 93-3933, Aug. 1993.

13. Fleming, P. and Purshouse, R., "Genetic Algorithms in Control Systems Engineering," Dept. of Automatic Control and Systems Engineering, University of Sheffield, Research Report No. 789, May 2001.

14. Holst, T. and Pulliam, T., "Aerodynamic Shape Optimization Using a Real-Number-Encoded Genetic Algorithm," AIAA Paper No. 2001-2473, June 2001.

15. Globus, A., Langhirt, E., Livny, M., Ramamurthy, R., Solomon, M. and Traugott, S., "JavaGenes and Condor: Cycle—Scavenging Genetic Algorithms," Java Grande 2000, ACM SIGPLAN, San Francisco, Calif., June 2000.

16. Lohn, J. and Colombano, S., "Automated Analog Circuit Synthesis Using a Linear Representation," Second Inter. Conference of Evolvable Systems: From Biology to Hardware, Springer-Verlag, Sept. 1998, pp. 23-25.

17. Linden, D. and Altshuler, E., Automating Wire Antenna Design Using Genetic Algorithms," *Microwave Journal,* Vol. 39, No. 3, March 1996.

18. Peigin, Sergey, "Genetic Algorithms for Optimization Problems with Non-Linear Constraints," *European Congress on Computational Methods in Applied Sciences and Engineering,* ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

19. Tse, D. and Chan, L., "Transonic Airfoil Design Optimization Using Soft Computing Methods," Canadian Aeronautics and Space Journal, Vol. 46, No. 2, June 2000, pp. 65-73.

20. Sheng, L. and Kapania, R., "Genetic Algorithms for the Optimization of Piezoelectric Actuator Locations," AIAA Paper No. 2000-1581, 2000.

21. Cook, A. and Crossley, W., "Investigation of Genetic Algorithm Approaches for Smart Actuator Placement for Aircraft Maneuvering," AIAA Paper No. 2001-0924, Jan. 2001.

22. Chen, V., Crawfore, L. and Menon, P., "Air Traffic Control Using Genetic Search Techniques," *Optimal Synthesis, Inc.,* Palo Alto, Calif., 1999.

23. Cravero, C. and Satta, A., "A Hierarchical Optimization Approach for Automatic Turbomachinery Blade Design," AIAA Paper No. 2001-3044, June 2001.

24. Marco, N, Désidéri, J.-A. and Lanteri, S., "Multi-Objective Optimization in CFD by Genetic Algorithms," Institut National de Recherche en Informatique et en Automatique, Research Report No. 3686, April 1999.

25. Naujoks, B., Willmes, L., Haase, W., Bäck, T. and Schütz, M., "Multi-Point Airfoil Optimization Using Evolution Strategies," *European Congress on Computational Methods in Applied Sciences and Engineering,* ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

26. Quagliarella, D. and Della Cioppa, A., "Genetic Algorithms Applied to the Aerodynamic Design of Transonic Airfoils," AIAA Paper 94-1896-CP, 1994.

27. Vicini, A. and Quagliarella, D., "Inverse and Direct Airfoil Design Using a Multiobjective Genetic Algorithm," *AIAA J.*, Vol. 35, 1997, pp. 1499-1505.

28. Hämäläinen, J., Mäkinen, A., Tarvainen, P. and Toivanen, J., "Evolutionary Shape Optimization in CFD with Industrial Applications," *European Congress on Computational Methods in Applied Sciences and Engineering,* ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

29. Anderson, M., Burkhalter, J. and Jenkins, R., "Missile Aerodynamic Shape Optimization Using Genetic Algorithms," J. of Spacecraft and Rockets, Vol. 37, No. 5, Sept.-Oct. 2000, pp. 663-669.

30. Anderson, M. and Gebert, G., "Using Pareto Genetic Algorithms for Preliminary Subsonic Wing Design," AIAA Paper No. 96-4023-CP, 1996.

31. Sasaki, D, Obayashi, S. and Nakahashi, K., "Navier-Stokes Optimization of Supersonic Wings with Four Design Objectives Using Evolutionary Algorithm," AIAA Paper No. 2001-2531, 2001.

32. Oyama, A., "Multidisciplinary Optimization of Transonic Wing Design Based on Evolutionary Algorithms Coupled with CFD Solver," *European Congress on Computational Methods in Applied Sciences and Engineering,* ECCOMAS 2000, Barcelona, Spain, Sept. 2000.

33. Obayashi, S., Yamaguchi, Y. and Nakamura, T., "Multiobjective Genetic Algorithm for Multidisciplinary Design of Transonic Wing Planform," *J. of Aircraft,* Vol. 34, 1997, pp. 690-693.

34. Oyama, A. and Liou, M.-S., "Multiobjective Optimization of Rocket Engine Pumps Using Evolutionary Algorithm," AIAA Paper No. 2001-2581, June 2001.

35. Oyama, A., "Wing Design Using Evolutionary Algorithms," PhD Thesis, Dept. of Aeronautics and Space Engineering, Tohoku University, Senadi, Japan, March 2000.

36. Houck, G. R., Joines, J. A. and Kay, M. G., "A Genetic Algorithm for Function Optimization: A Matlab Implementation," North Carolina State University-IE, TR 95-09, 1995.

37. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs,* AI Series, Springer-Verlag, New York, 1994.