AIAA JOURNAL
Vol. 34, No. 12, November 1996

Constraint Handling in Genetic Search Using Expression Strategies

P. Hajela* and J. Yoo†
Rensselaer Polytechnic Institute, Troy, New York 12180-3590

A traditional penalty-function formulation for treatment of nonlinear constrained optimization problems in genetic search has been shown to be extremely sensitive to user-specified schedules of selecting penalty parameters. The sensitivity of such an approach is manifested in a biasing of the search toward suboptimal designs and a general increase in the number of function evaluations required to obtain a converged design. Alternative methods are described for handling constraints that are motivated by the fact that the structure of both feasible and infeasible designs is generally present in the population of designs at any generation of search. A preconditioning of the infeasible designs prior to the genetic transformations, by an expression operation that is conceptually analogous to the theory of dominant and recessive genes in genetics, is shown to be highly effective in evolving feasible solutions, and with savings of computational resource. Two alternative implementations of this approach are described and a comparison is made of numerical efficiency vis-à-vis the penalty-function-based approach.

I. Introduction

THERE has been considerable recent interest in the application of genetic-algorithm-based search methods in design optimization.[1–5] This interest has been piqued by an increased application of optimization methods in practical design problems where the design space may be nonconvex,[1] or, in some instances, even disjoint.[2] Similarly, design problems where the design variables are of discrete or integer type[3] are also more difficult to accommodate using the more conventional mathematical programming techniques, and the use of the latter has been shown to yield suboptimal designs.

Genetic algorithms (GAs) emulate the biological evolution process. A number of design alternatives constitute a population that is subjected to a number of genetic evolution-type operators to evolve a new generation of designs. The new generations are better adapted to an environment that could be defined in terms of the objective function value and/or the level of constraint satisfaction. The population of designs thus evolves into a direction of improving objective and constraint function values. Among the basic genetic transformation operators is a selection operator that simply chooses more fit members of the present generation and gives them a higher probability of mating and propagating their genes into subsequent generations. There are two other operators—crossover and mutation—introduced to explore the design space for other promising designs. The former, in particular, is analogous to the biological counterpart, where gene contributions from mating parents come together in a progeny. In addition to these basic operations, transformation operators, such as inversion,[4] have been introduced in recent work as another approach to enhance the exploratory qualities of this search process. Detailed discussions on the many facets of GAs and their applications are available.[6]

The strength of GAs is largely attributed to the fact that the method is not a point-to-point search as in gradient-based search methods; instead, the approach uses information from several regions of the design space to advance an entire set of designs. The method uses representations of design variables rather than variable values themselves; this provides for routine handling of discrete and integer variables in the design problem. Even though the approach belongs to a general category of stochastic search techniques, it is less vulnerable to criticisms of loss in efficiency with increased problem dimensionality, primarily because of its intrinsically parallel data processing capability.[7] Nevertheless, enhancements to the basic approach have been proposed for application of GAs in large-scale optimization problems.[8]

The handling of design constraints in a genetic-search-based approach is somewhat contrived. In a traditional GA implementation, populations of designs are evolved from one generation to another, with survival biased in favor of those members of the population that have a higher fitness in the present generation. The fitness is represented by a scalar measure—in the case of unconstrained optimization, the fitness can be represented in terms of the objective function value alone. If the traditional GA formulation is to be used in the more frequently encountered problem of constrained optimization, the scalar fitness function has to be formulated as a composite of the objective and constraint functions. The subject of constraint handling in GAs by penalty functions is discussed elsewhere.[9, 10] However, it has been shown that there is considerable sensitivity of the penalty-function approach to the choice of user-specified weighting constants, which results in biasing the search toward suboptimal designs.

Another strategy to account for design constraints has been explored in which the constraints were explicitly taken into consideration by using them to limit the range of variation in the design variables.[11] This approach assumed that the constraints were linear and that the design space was convex. The approach, however, was of limited value, because a number of realistic design problems involve nonlinear equality and inequality constraints. Furthermore, the principal motivation for using GAs is in the usefulness of this approach in problems where the design space may be multimodal.

The present paper describes an alternative approach for handling nonlinear, equality and inequality constrained design optimization problems. This approach is heuristic in nature, and derives from an observation that the chromosome-like structure of both feasible and infeasible designs is present at any generation of the simulated evolution process. In a traditional GA approach of simulating the biological evolution process, the chromosome structure is represented by a single strand (a haploid representation). The biological chromosomal structure is, in fact, a double-strand helix (a diploid representation). Genes at specific locations on either string may be of a dominant or a recessive type, and the expressed gene at a location is what characterizes a particular trait. The present idea draws upon this concept, forming pairs from a given population to represent a diploid chromosome structure, and using a probabilistic expression operator to develop a constraint-conditioned population to which a traditional unconstrained genetic search is applied. Two

Received Nov. 25, 1995; revision received May 3, 1996; accepted for publication July 1, 1996; also published in *AIAA Journal on Disc*, Volume 2, Number 1. Copyright © 1996 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Professor, Department of Mechanical Engineering, Aeronautical Engineering, and Mechanics. Associate Fellow AIAA.

†Graduate Research Assistant, Department of Mechanical Engineering, Aeronautical Engineering, and Mechanics.

variants of this basic approach are presented in subsequent sections and applied to representative structural optimization and multimodal problems. The numerical results are compared to a penalty-function approach[12] that has been shown to be effective in previous work.

II. Conventional GA Implementation

The basic approach in GAs is to represent possible solutions to a given problem by a population of bit strings of finite lengths, and subsequently to use transformations analogous to biological reproduction and evolution to improve and vary the coded solutions. This approach is facilitated by defining a fitness function or a measure indicating the goodness of a member of the population in a given generation during the evolution process. For unconstrained maximization problems, the objective function could serve as the fitness function. The inverse of the objective function, or the difference between a large number and the objective function, can be used as the fitness function in a minimization problem. Central to this approach is the step by which a design is coded into a stringlike structure, and a commonly used approach is to represent each variable by a fixed-length binary-number representation of the variable value. The length of the string determines the precision with which each variable is represented in the search. For a design variable x_i represented by a five-digit binary string, the lower and upper bounds of the variable can be represented as follows:

$$x_i^L = 00000 \qquad x_i^U = 11111$$

A linear scaling can be introduced to convert intermediate values of the binary number into design-variable values. For the case of n variables, the binary-string representations for x_i can be placed head-to-tail to create a $5n$-digit number, which is a chromosome-like representation of the design. Several such chromosomal strings are defined to constitute a population of designs, which would generally include a mix of feasible and infeasible designs. This population of designs then is subjected to three basic genetic transformations, including reproduction, crossover, and mutation.

The reproduction process is nothing more than biasing the population of designs by introducing multiple copies of better designs and simultaneously eliminating designs with poorer values of the fitness function. This results in the creation of a mating pool of the same size as the initial population, but with a higher average fitness-function value. Note that the GA simulation is contrived in that the population size is fixed from one generation to another. Variants of this basic GA approach have been explored where the population is allowed to change over generations. In the most common GA implementation, reproduction represents an elitist selection process that retains only the fittest members of a population for mating; it does not in any way improve or create new designs. It is the crossover transformation that allows the characteristics of the designs in the population pool to be altered, thereby exploring new designs. A simple crossover operation (two-point) consists of randomly selecting two mating parents from the pool, randomly choosing two sites on the genetic strings, and swapping strings of zeroes and ones between two chosen sites among the mating pair. An illustration of the crossover process between mating parents represented by 10-digit binary strings is as follows:

$$\text{Parent 1} = 110\underline{0}1\underline{0}0100 \qquad \text{Parent 2} = 010\underline{1}1\underline{1}0001$$

$$\text{Child 1} = 1101110100 \qquad \text{Child 2} = 0100100001$$

The crossover sites on the parent strings are indicated by an underscore. A probability of crossover p_C is defined to determine whether crossover should be implemented. The third transformation operator, mutation, safeguards the genetic search process from a premature loss of genetic information because of reproduction. The process of mutation simply involves selecting few members from the population pool, and to switch a zero to one, or vice versa, at a randomly selected mutation site on the chosen string. This operation is done with a low probability of mutation p_M. The process of reproduction, crossover, and mutation is repeated over a number of generations so that all designs in the population converge to a design with the best fitness value. More details on the use of the GA approach are available elsewhere.[6]

III. Constraint Treatment in GA

This section describes the three strategies for constraint handling in GAs for which numerical studies have been performed. The general mathematical statement of this optimization problem can be written as follows:

Minimize $F(x)$ subject to

$$\begin{aligned} g_j(x) &\le 0, & j &= 1, m \\ h_k(x) &= 0, & k &= 1, p \\ x_i^L \le x_i &\le x_i^U, & i &= 1, n \end{aligned} \tag{1}$$

Here, $F(x)$ is the function to be minimized; $g_j(x)$ and $h_k(x)$ are the m-inequality and p-equality constraints for the problem; and x_i are the n-design variables with lower and upper bounds denoted as x_i^L and x_i^U, respectively. The penalty-function formulation used in the present work is similar to that described by Rao et al.[5] and is summarized here for completeness.

A. Penalty-Function Approach

To use genetic search in the problem, the constrained minimization problem is first converted into an unconstrained problem, using the exterior penalty-function formulation, resulting in the following problem:

$$\text{Minimize } \bar{F} = F + \bar{P} \tag{2}$$

Here, $\bar{F}$ is the modified objective function that also contains the penalty term $\bar{P}$, which brings the constraint functions into the problem. Careful consideration must be given to this selection, and in the present work, the following bounding strategy was adopted. If the average fitness of feasible designs is F_{av}, then a limiter value of the penalty $\bar{L}$ is selected as $\bar{L} = kF_{\text{av}}$, where k is typically selected as 2; the penalty $\bar{P}$ that is appended to an infeasible design then is obtained as follows:

$$\bar{P} = \begin{cases} G & \text{if} \quad (G \le \bar{L}) \\ [\bar{L} + \alpha(G - \bar{L})] & \text{if} \quad (G > \bar{L}) \end{cases} \tag{3}$$

where

$$G = r\left(\sum_{j=1}^{m} \langle g_j \rangle\right) \tag{4}$$

where r is a penalty parameter of the form encountered in the exterior penalty-function approach and $\langle g_j \rangle$ represents the violated inequality constraints. A similar function also could be obtained for the equality constraints. The effect of the aforementioned scaling operation is to prevent radical departures in the value of the penalty term from the specified $\bar{L}$ value. If $\alpha = 0.0$, the penalty for all violated designs is limited to $\bar{L}$. If instead, α is assigned a small value of the order 0.2, then the extent of constraint violation attributable to severely violated and less violated designs varies linearly from $\bar{L}$, albeit with a small slope. To convert this function minimization into a fitness maximization as required by GAs, the following fitness function is created:

$$f_i = \bar{F}_{\max} - \bar{F} \tag{5}$$

where f_i is the fitness of the ith design and $\bar{F}_{\max}$ is the maximum value of $\bar{F}$.

The performance of the penalty-function approach described here is clearly dependent on a number of user-specified constants, such as the penalty parameter r, the factor k used to establish $\bar{L}$, and the slope parameter α. The two alternative approaches examined in this work were an attempt to reduce the dependency of the process on the selection of these parameters.

B. Expression-Based Strategies

The basis of this approach is to combine the favorable characteristics of both feasible and infeasible designs present in a given population at any generation through the use of an expression operator, which, like the crossover and mutation operations in genetic search, is probabilistic in nature. In a simplistic view of the biological analog of genetic search, the chromosome structure is a double helix, containing two strands of chromosomes. Each of these strands has genes at specific locations (they can be thought of simplistically as beads on a string if one ignores the idea that genes actually form and exist in linkage groups) that describe a particular trait. These genes can be of dominant or recessive type, and the expressed gene at a particular site is what determines the trait that is manifested in an individual. Complex rules for determining the expressed gene have been developed in genetics.[13] In GAs, each design is represented only by a single-strand chromosome structure. However, this does not prevent the use of simplified expression operators if one considers selected pairs of genes in the population as a temporarily assembled diploid chromosome from which an expressed haploid chromosome is developed. In particular, if one considers such pairs to be assembled from a combination of feasible and infeasible designs, the infeasible design could be replaced by an expressed chromosome that incorporates features of the feasible design. The use of such an operation would have the natural effect of eliminating constraint violations from the population.

Consider two 10-digit binary strings representing a feasible and an infeasible design as follows:

String A (feasible):1110010110

String B (infeasible):1000011011

The expression operator is applied on a bit-by-bit basis to determine an expressed chromosome that would replace string B in the population. This operator simply substitutes the zero or one at a specific location in string B with the corresponding value from string A with some prescribed probability, p_E. The manner in which the strings are selected for the expression operation, and the specification of the probability of carrying out this expression, distinguishes the two approaches that were studied. A stepwise description of the two alternative strategies is summarized below.

1. *Strategy 1*

Step 1) The population of designs is first generated at random. A uniform distribution of the design variables between specified lower and upper bounds was the objective of this random initialization.

Step 2) The population of designs generated in step 1 was evaluated to determine the objective and constraint functions.

Step 3) The infeasible designs were combined with feasible designs in the population in the following manner:

a) The best feasible design in the population was identified as x_{best}.

b) All infeasible designs were ranked on the basis of the constraint value, with a higher rank given to the more feasible designs. For N infeasible designs, the ranks would range from one to N, with rank N assigned to the design with the most constraint violation.

c) Each infeasible design was combined with x_{best} through the use of the expression operation on a bit-by-bit basis. The probability of expression p_E was determined by the individual rank of the design. Uniform random integers were generated between one and population size. If the specific gene at the ith location of the expressed chromosome is represented by g_i^E, and those of the best and jth violated design are denoted as g_i^B and g_{ij}^V, respectively, then the expressed gene is obtained as

$$g_i^E = \begin{cases} g_i^B & \text{if} \quad r_i < R_j \\ \\ g_{ij}^V & \text{if} \quad r_i \geq R_j \end{cases} \qquad j = 1, \ldots, N \qquad (6)$$

where r_i is a randomly generated integer between one and population size and R_j is the rank of the jth infeasible design. For designs with a higher constraint violation, i.e., lower rating in the ranking scale, the expression operator would tend to make the chromosomes of these designs more similar to the best design.

Step 4) The expressed designs are reevaluated to determine the objective- and constraint-function values.

Step 5) The selection operation of the traditional GA approach is applied to the feasible designs in the population at this stage of the computation. This selection operation creates multiple copies of the better feasible designs and purges some of the less attractive feasible designs. These selected designs then are combined with all existing infeasible designs to create a population of the same size as before (this is necessary only if we work with fixed population size).

Step 6) The crossover and mutation operations are performed on this new population as in a traditional GA implementation.

The process is repeated from step 2 through step 6 until a converged design is obtained, or a prescribed maximum number of function evaluations has been performed.

2. *Strategy 2*

The stepwise implementation of this strategy is identical to strategy 1 with the exception of step 3, i.e., the manner in which the expression operator is invoked. This strategy is more selective in determining the pair of designs that is subjected to the expression operation. This selectivity was imposed by matching infeasible designs to feasible designs with the most similar objective-function value. If the difference in objective-function values between an infeasible design J and a feasible design I is defined as δ_{IJ}, where

$$\delta_{IJ} = \text{Obj}(I) - \text{Obj}(J) \qquad (7)$$

then design I, which yielded the smallest absolute value of δ_{IJ}, was selected for expression; the exception to this rule was that a negative δ_{IJ} was preferred over a positive δ_{IJ} even if the absolute value of the latter was smaller. This rule was imposed on the premise that such a choice would not only improve the constraint response of the expressed infeasible design, but also would improve its objective-function value. Once the pair of strings used for expression was identified, the expression operation was carried out on a bit-by-bit basis with a fixed probability of expression p_E, as follows:

$$g_i^E = \begin{cases} g_i^{BF} & \text{if} \quad r_i \leq p_E \\ \\ g_{ij}^V & \text{if} \quad r_i > p_E \end{cases} \qquad j = 1, \ldots, N \qquad (8)$$

Here, g_i^{BF} is the ith bit on the selected feasible design string, g_{ij}^V is the corresponding bit on the jth infeasible string, and r_i is a random number (based on a uniform distribution) between zero and one. The expressed designs were reevaluated, and subsequent steps of the GA were carried out as in strategy 1.

IV. Equality Constraints

The foregoing discussion has focused on handling inequality constraints only. In the case of explicit equality constraints, it is possible to use these constraints to eliminate some design variables from the problem. This option is unavailable for implicit constraints, and a somewhat modified approach is required in including these constraints in the expression-based method. Note that the expression-based approach requires sorting of designs as infeasible or feasible at every generation to implement the expression operation. To accommodate this requirement in the presence of equality constraints, the strict equality is replaced by including a generous band around the equality constraints (about 100% violation beyond a specified bound) in which the design is considered to be near feasible. The width of this band is gradually reduced as more designs move inside the band. If the design space contains widely spaced discrete variables, a wider band of feasibility is recommended to prevent oscillations in the convergence pattern. The Kreisselmeir–Steinhauser (KS) function was used to represent the equality constraints as suggested elsewhere.[14] If h_i is the ith equality constraint, then it can be represented by a pair of inequality constraints as

$$h_i \leq 0 \qquad -h_i \leq 0 \qquad (9)$$

The KS function can be used to fold these constraints into a cumulative measure Ω as follows:

$$\Omega = (1/\rho) \ln(e^{\rho h_i} + e^{-\rho h_i}) - (1/\rho) \ln 2 + c_1 \qquad (10)$$

where c_1 denotes the width of the band and ρ is a user-specified constant. For $h_i = 0$, the cumulative measure would take on a value c_1. By reducing the width of the band c_1, the designs are forced to move closer to the equality constraint.

V. Test Problems

The strategies for constraint handling described in preceding sections were applied to the optimal sizing of a truss structure for strength- and stiffness-related constraints, and a multimodal algebraic function.

Ten-bar truss: The geometry of this truss and the applied loads are shown in Fig. 1. The cross-sectional areas of each bar element were sized to obtain a minimum weight structure for prescribed limits on stresses and displacements. The problem was considered in three parts. First, the structure was sized for stress constraints only, and an allowable stress limit (both tension and compression) of 25 ksi was prescribed. As a variation of this problem, constraints on nodal displacements were imposed in addition to the stress constraints in the previous problem. Next, the vertical displacement at node A was restricted to $\leq w_{\mathrm{al}} = 2.0$ in. Last, an inequality constraint was converted to an equality constraint by requiring the stress in member 9 of Fig. 1 to be exactly equal to 25 ksi. In all of these problems, each cross-sectional area of the truss elements was considered as an independent design variable.

Multimodal algebraic function: The expression-based approach has strongly convergent characteristics, which requires setting the probabilities of mutation to higher values than normal. To test how the proposed approach behaves in the presence of nonconvexities in the design space, a constrained multimodal function was used as a second test problem. The mathematical statement of this optimization problem can be written as follows:

Minimize

$$F(X) = 0.25X_1^4 - 3X_1^3 + 11X_1^2 - 13X_1 + 0.25X_2^4 - 3X_2^3 + 11X_2^2 - 13X_2 \tag{11}$$

Subject to

$$g_1(X) \equiv 4 - X_1 - X_2 \leq 0$$

As shown in Figs. 2a and 2b, this function has multiple relative minima; the linear constraint for the problem only excludes one relative minimum from the feasible space. The global minimum of this problem is located at (5.3301, 5.3301) and corresponds to an objective function value of -18.568.

The first test problem is discussed in greater detail in Ref. 15, which also presents the known optimal solutions for this problem obtained through the use of mathematical programming and optimality criteria approaches. It must be emphasized that it is not our intent to compare the computational efficiency of the GA approach to that of the mathematical programming methods. We simply wish to compare the final solutions obtained by the different strategies. Note further that the solutions for the 10-bar truss listed[14] correspond to a continuous variation in the design variables, and a direct comparison of results is not possible, because the GA approach works on a discrete representation of the design space; solutions obtained through that approach would depend upon the granularity of the discrete representation—the continuous solution represents a lower bound to the solution obtained through GA. For the algebraic function, however, the exact solution could be very nearly obtained in the GA approach, using a design-variable representational precision of 0.01.

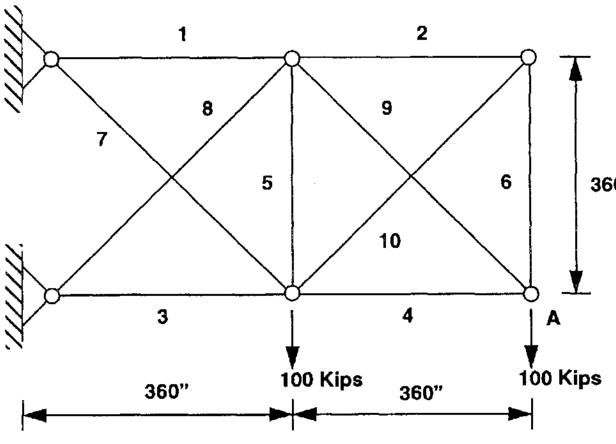

Fig. 1 Ten-bar truss.

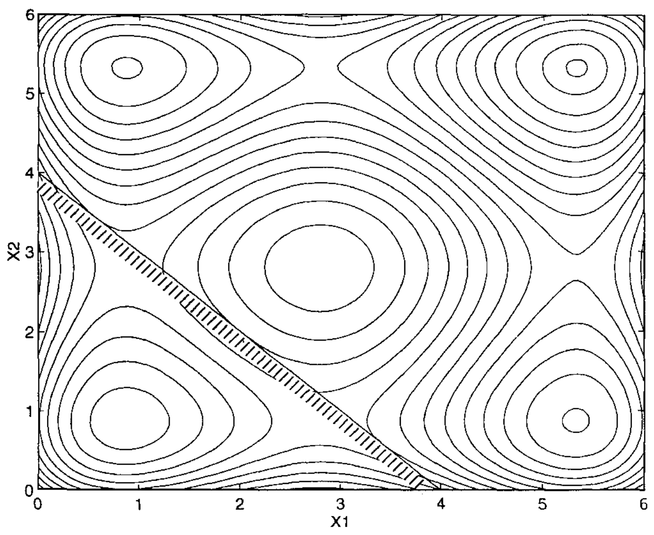

Fig. 2a Contour plot of the multimodal algebraic function.

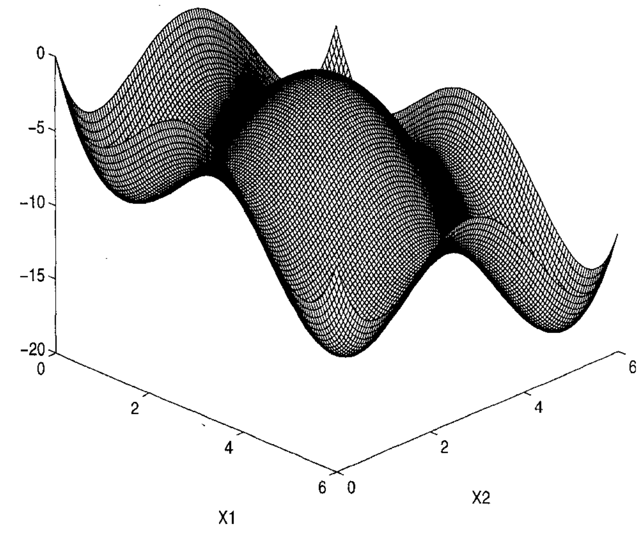

Fig. 2b Three-dimensional plot of the multimodal algebraic function.

VI. Discussion of Results

A number of numerical experiments were conducted to provide a reasonable basis for comparison between the traditional penalty-function-based implementation of constrained GA search and the newly proposed strategies based on the expression operator. The GA code EVOLVE[16] was used in each of these experiments. Note that the stochastic nature of GA search renders such a comparative study somewhat difficult. The progression of GA search is determined by a pseudo-random-number generator, which in turn depends on a user-specified seed. To minimize the effect of these variations, each strategy was implemented using the same seed, which resulted in identical initial populations for each case. Furthermore, each test case was run several times, with different values of the seed number. The schemes were then ranked in their ability to find the best design and with what consistency, and how the scheme performed on average. This comparison was done on the basis of both the final solution and the computational effort required to obtain the best solution.

Table 1 Results for 10-bar truss problem with stress constraints only

	Original GA		Strategy 1		Strategy 2	
Trial no.	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations
1	1,685	32,229	1,688	6,351	1,678	23,721
2	1,713	32,278	1,639	27,083	1,677	34,971
3	1,821	30,337	1,639	25,028	1,639	14,803
4	1,901	21,240	1,685	18,982	1,635	17,906
5	1,717	32,707	1,724	6,642	1,688	16,862
6	1,773	16,602	1,740	11,347	1,666	26,401
7	1,776	26,364	1,639	25,415	1,692	23,399
8	1,914	33,964	1,768	9,160	1,725	33,871
Best results	1,685	16,602	1,639	6,351	1,635	14,803
Average of all runs	1,788	28,215	1,690	16,251	1,675	23,992
Deviation from best	229	17,362	129	20,732	90	20,168

Table 2 Results for 10-bar truss problem with stress and displacement constraints

	Original GA		Strategy 1		Strategy 2	
Trial no.	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations
1	1,921	33,107	1,996	29,527	1,942	20,683
2	1,849	30,995	1,846	29,602	1,840	30,576
3	1,879	25,804	1,879	20,600	1,852	34,665
4	1,855	16,181	1,844	19,948	1,844	26,720
5	1,936	26,053	1,844	13,742	1,928	19,740
6	1,958	26,623	1,925	25,981	19,02	27,088
7	1,910	35,520	1,862	10,044	1,844	22,949
8	1,903	30,022	1,894	36,125	19,41	35,879
Best results	1,849	16,181	1,844	10,044	1,840	19,740
Average of all runs	1,901	28,038	1,886	23,196	1,887	27,288
Deviation from best	109	19,339	152	26,081	102	16,139

The strategies were first implemented for the 10-bar truss problem with stress constraints only. Lower and upper bounds on the design variables were prescribed as $x_i^L = 0.1$ in.^{2} and $x_i^U = 11.0$ in.^{2}, respectively. A precision of design variable representation of 0.2 in.^{2} was stipulated, which resulted in a total string length of 60 for the problem. Probabilities of crossover and mutation for the plain GA were prescribed as 0.8 and 0.02, respectively. Based on prior experience with using GAs in problems with comparable string lengths, a population size of 120 was selected for this problem. Because both expression operators force a more rapid convergence of the population, these schemes performed better with higher probabilities of mutation, and the results presented for expression-based strategies 1 and 2 used a probability of mutation of 0.1 and 0.08, respectively. This higher mutation probability retains diversity in the population for a larger number of generations. A summary of the results obtained for eight different seed numbers is presented in Table 1. In each of these cases, strategies 1 and 2 performed better than the plain GA, resulting in lower weight, and with less computational effort. The deviations in the solutions were, in general, lower when using the expression strategies as opposed to the plain GA solution. Figure 3 shows the convergence patterns of each of the strategies, plotting the objective-function values as a function of the number of function evaluations performed. By the very nature of the expression-based strategies, all objective-function values in this plot corresponding to these strategies represent feasible designs. In the plain GA approach, the objective function is a composite of the weight and any constraint violation. It is clear from Fig. 3 that both expression-based strategies converge to the optimal solution faster than the plain GA. Furthermore, the expression-based strategies produce lower-weight feasible solutions very early in the process; a much higher investment of computational resource is required to obtain converged feasible solutions with the penalty-function-based approach. Figure 4 shows the convergence patterns of the plain GA approach for different choices of the penalty-enforcing scheme. The

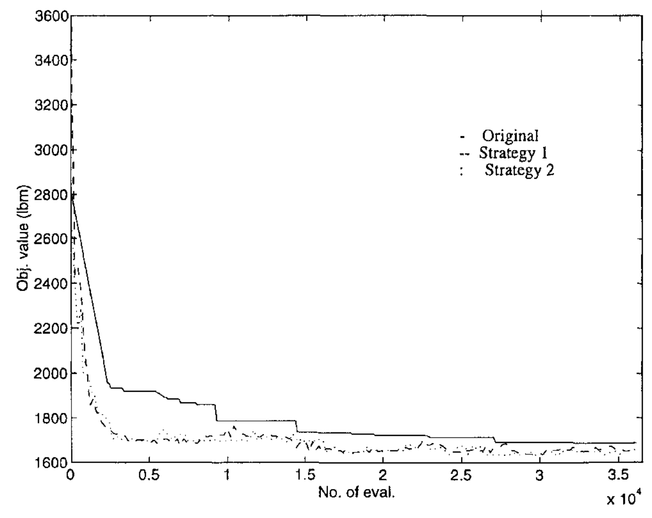

Fig. 3 Iteration history of 10-bar truss problem with stress constraints only.

sensitivity of the GA process to the choice of these parameters is obvious from this plot.

Similar results were obtained for the 10-bar truss problem with stress and displacement constraints. These results are summarized in Table 2 and show trends similar to those obtained for stress constraints only. The upper and lower bounds on the design variable, as well as the GA search parameters, were the same as in the 10-bar truss problem with stress constraints. Convergence histories for each strategy are depicted in Fig. 5, and, from which, conclusions similar to those in the previous case may be derived. Results for the algebraic-function problem are summarized in Table 3. It is clear from the table that all three methods, on average, produced similar results in roughly the same number of function evaluations. The strongly convergent characteristics of the expression-based strategies did not force these strategies to converge to one of the other relative optima, and the global optimum was always identified. Indeed, the manner in which the constraint is specified does not allow for the expression strategies to show dominant performance over the penalty-function approach. If one examines Fig. 2a, it is quite obvious that the global optimum is far removed from the constraint boundary, and once the designs are rendered feasible, the expression-based approaches behave much like the plain GA on an unconstrained problem. In this case, the higher probabilities of mutation for the expression strategies simply retard the expression strategies from achieving quick convergence. In a typical execution, 100% feasibility in the expression-based approaches was reached in one to two generations of evolution; a completely feasible population was typically obtained in the plain GA approach after 10 generations of evolution. Lower and upper bounds on design variables of 0.0 and 6.0, respectively, were used. A string length of 10 digits for each variable allowed these variables to be represented to a precision of 0.01. A population size of 40 was used in the GA simulations.

The 10-bar truss problem also was used to study the performance of the approach for equality constraints. The stress in member 9 of the structure was required to be equal to 25 ksi; all other stress and displacement constraints were retained as before. In the plain GA approach, a 5% constraint violation was allowed throughout the search process. In both expression-based methods, a 100% constraint violation was permitted initially, and then was reduced sequentially to 50, 30, 10, and 5% after every two generations of evolution. Table 4 summarizes the results of eight trials in all three strategies. Both expression-based strategies seem to perform better than the penalty-function method in handling equality constraints. The average results indicate that strategy 2 is more efficient in finding the optimal solution with fewer number of function evaluations than strategy 1. It is clear in Fig. 6 that the expression-based approach can locate the designs near the optimum at a much earlier stage of the search process.

For the 10-bar truss problem, strategy 2 seems to perform marginally better than strategy 1, but at an increased computational cost. This can be attributed to the manner in which the expression operator is invoked. The bit string for each infeasible design is conditioned against a feasible design that most closely resembles the infeasible design in terms of the objective-function value. The motivation for this selection is that those segments of the bit string that contribute to calculating the objective function will be similar on both strings, and would therefore be minimally altered; however, differences in strings would be the ones contributing to constraint violations, and segments of these strings from the feasible design would be adapted into the infeasible design.

Table 3 Results for multimodal function problem

	No. of function evaluations		
Trial no.	Original GA	Strategy 1	Strategy 2
1	1376	1143	1071
2	1593	1343	1975
3	3396	4066	5263
4	5664	4546	1355
5	1407	2629	724
6	1724	1009	4850
7	611	252	752
8	1221	3789	645
Average	2124	2347	2079

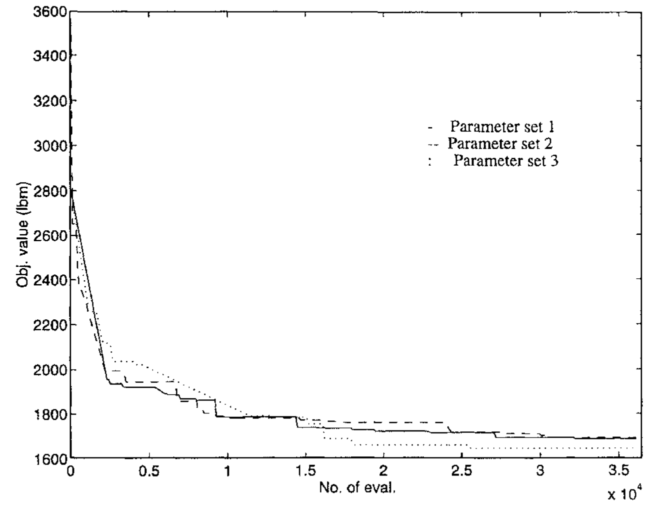

Fig. 4 Iteration history of 10-bar truss problem using different sets of penalty parameters.

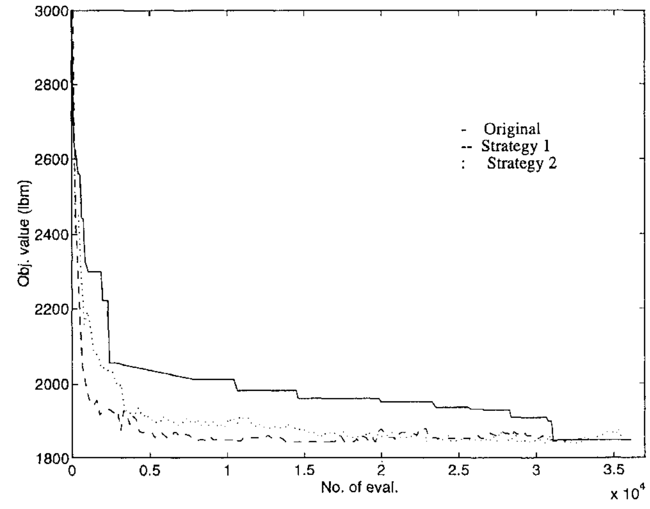

Fig. 5 Iteration history of 10-bar truss problem with stress and displacement constraints.

Table 4 Results for 10-bar truss problem with an equality constraint

	Original GA		Strategy 1		Strategy 2	
Trial no.	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations	Optimal objective, lb	No. of function evaluations
1	1,958	27,051	1,978	35,364	1,958	26,716
2	2,039	24,258	1,862	31,537	1,891	38,918
3	1,927	32,807	1,872	35,642	1,856	24,088
4	1,918	22,553	1,861	29,436	1,856	11,127
5	1,935	32,510	1,967	18,754	1,937	19,805
6	1,925	27,485	1,914	19,654	1,920	31,423
7	1,981	31,836	1,854	34,114	1,862	27,578
8	2,026	7,894	1,996	26,181	1,849	24,985
Best results	1,918	22,553	1,854	34,114	1,849	24,985
Average of all runs	1,964	25,799	1,913	28,835	1,891	25,205
Deviation from best	121	24,913	142	16,888	109	24,791

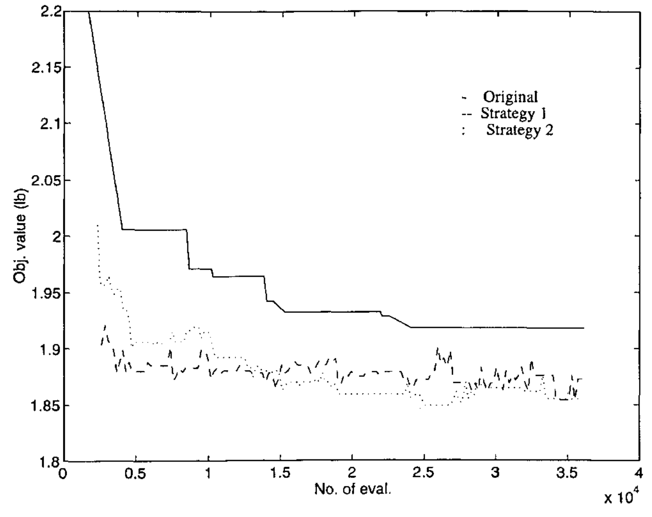

Fig. 6 Iteration history of 10-bar truss problem with an equality constraint.

VII. Conclusions

The present paper explores alternative methods to account for nonlinear constraints in GA-based optimal design. The traditional penalty-function-based formulation is shown to be sensitive to user-specified schedule for penalizing constraint violations. This sensitivity results in suboptimal solutions and in increased computational effort. The alternative strategies are based on the use of an expression operator that preconditions the infeasible designs in the population by forcing its bit strings to resemble those of feasible designs. The alternative strategies performed better than the plain GA with the penalty-function treatment of constraints, generally producing better results with less computational effort. These strategies do not require the specification of a number of user-specified parameters that determine the penalty schedule, and are therefore considered more robust for general GA applications. Continuing efforts in this area are focused on examining the effect of increased problem dimensionality on this approach.

Appendix: Modifications in Schema Growth Equation

The rapid convergence of the expression-based approach can be explained by modifying the well-known schema growth equation[6] to include the effect of the expression operation. The number of a particular schema of type H at a time step $t + 1$ is defined as

$$m(H, t+1) \geq m(H, t) \cdot [f(H)/\bar{f}]$$
$$\times \{1 - p_C[\delta(H)/l - 1] - \mathcal{O}(H)p_M\} \qquad \text{(A1)}$$

Here, $f(H)$ is the fitness of the schema H; $\bar{f}$ is the average fitness of the population; l is the string length; $\delta(H)$ and $\mathcal{O}(H)$ are the schema defining length and order; and p_C and p_M are the probabilities of crossover and mutation, respectively. This equation indicates that a schema with higher-than-average fitness, short defining length, and low order will survive, and its numbers will grow exponentially in subsequent generations. In the expression-based approach, the expression operation modifies this equation. Let us denote the schema of the best feasible design as H_1, a feasible design as H_2, and of the infeasible design as H_3. If the infeasible design is acted upon by the best feasible design through a probability of expression p_E, then the probability that it becomes similar to schema H_1 is given as $p_E^{\mathcal{O}(H_1)}$. The number of schema of type H_3 that are converted to type H_1 through the expression operation is given as $m(H_3, t)p_E^{\mathcal{O}(H_1)}$, and this number adds to the existing schema of type H_1 that are then subjected to the crossover and mutation operation. The schema growth equation can be applied to schema of type H_1 as follows:

$$m(H_1, t+1) \geq \left[m(H_3, t)p_E^{\mathcal{O}(H_1)} + m(H_1, t)\right]$$
$$\times [f(H_1)/\bar{f}]\left\{1 - p_C[\delta(H_1)/l - 1] - p_M^{\mathcal{O}(H_1)}\right\} \qquad \text{(A2)}$$

Note that the effect of mutation must be modified somewhat from the original schema growth equation if large probabilities of mutation must be used to prevent premature convergence. The other feasible designs characterized by schema H_2 will increase or decrease as determined by the ratio of their fitness to the average population fitness, as follows:

$$m(H_2, t+1) \geq m(H_2, t)[f(H_2)/\bar{f}]$$
$$\times \left\{1 - p_C[\delta(H_2)/l - 1] - p_M^{\mathcal{O}(H_2)}\right\} \qquad \text{(A3)}$$

Finally, infeasible schema of type H_3 will be eliminated as indicated in the following equation, where $(1 - p_E)^{\mathcal{O}(H_3)}$ is the probability that schema H_3 will survive the expression operation and is always less than unity:

$$m(H_3, t+1) \geq m(H_3, t)(1 - p_E)^{\mathcal{O}(H_3)}$$
$$\times \left\{1 - p_C[\delta(H_1)/l - 1] - p_M^{\mathcal{O}(H_3)}\right\} \qquad \text{(A4)}$$

Acknowledgment

Partial support for this work received under U.S. Army Research Office Contract DAAH-04-93-G-0003 to the Rensselaer Rotorcraft Technology Center is gratefully acknowledged.

References

[1]Hajela, P., and Lee, E., "Genetic Algorithms in Topological Design of Grillage Structures," *Discrete Structural Optimization*, edited by W. Gutkowski and J. Bauer, Springer–Verlag, Berlin, 1994, pp. 30–39.

[2]Hajela, P., "Genetic Search—An Approach to the Nonconvex Optimization Problem," *AIAA Journal*, Vol. 26, No. 7, 1990, pp. 1205–1210.

[3]Lin, C.-Y., and Hajela, P., "Genetic Algorithms in Optimization Problems with Discrete and Integer Design Variables," *Journal of Engineering Optimization*, Vol. 19, No. 4, 1992, pp. 309–327.

[4]Le Riche, R., and Haftka, R. T., "Optimization of Laminate Stacking Sequence for Buckling Load Maximization by Genetic Algorithms," *AIAA Journal*, Vol. 31, No. 5, 1995, pp. 951–956.

[5]Rao, S. S., Pan, T.-S., and Venkayya, V. B., "Optimal Placement of Actuators in Actively Controlled Structures Using Genetic Algorithms," *AIAA Journal*, Vol. 29, No. 6, 1991, pp. 942, 943.

[6]Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, MA, 1989, Chap. 2.

[7]Holland, J. H., "Schemata and Intrinsically Parallel Adaptation," *Proceedings of the NSF Workshop on Learning Systems Theory and Its Applications*, edited by K. S. Fu and J. S. Tou, Gainesville, FL, 1973, pp. 43–46.

[8]Lin, C.-Y., and Hajela, P., "Genetic Search Strategies in Large Scale Optimization," *Proceedings of the AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics, and Materials Conference* (La Jolla, CA), AIAA, Washington, DC, 1993, pp. 2437–2447.

[9]Smith, A. E., and Tate, D. M., "Genetic Optimization Using a Penalty Function," *Proceedings of the 5th International Conference on Genetic Algorithms*, edited by S. Forrest, Morgan Kaufmann, San Mateo, CA, 1993, pp. 409–505.

[10]Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M., "Some Guidelines for Genetic Algorithms with Penalty Functions," *Proceedings of the 3rd International Conference on Genetic Algorithms*, edited by J. D. Schaffer, Morgan Kaufmann, San Mateo, CA, 1989, pp. 191–197.

[11]Michalewicz, Z., and Janikow, C. Z., "Handling Constraints in Genetic Algorithms," *Proceedings of the 4th International Conference on Genetic Algorithms*, edited by R. K. Belew, Morgan Kaufmann, San Mateo, CA, 1991, pp. 151–157.

[12]Lin, C.-Y., "Genetic Search Methods for Multicriterion Optimal Design of Viscoelastically Damped Structures," Ph.D. Dissertation, Dept. of Aerospace Engineering, Engineering Science and Mechanics, Univ. of Florida, Gainesville, FL, 1991.

[13]Stansfield, W. D., *Genetics—Theory and Problems*, McGraw–Hill, New York, 1991, pp. 27, 365.

[14]Sobieszanski-Sobieski, J., "A Technique for Locating Function Roots and for Satisfying Equality Constraints in Optimization," *Structural Optimization Journal*, Vol. 4, Nos. 3–4, 1992, pp. 241–243.

[15]Schmit, L. A., and Miura, H., "Approximation Concepts for Efficient Structural Synthesis," NASA CR-2552, 1976.

[16]Lin, C.-Y., and Hajela, P., "EVOLVE: A Genetic Search Based Optimization Code with Multiple Strategies," *Proceedings of OPTI'93, Computer-Aided Optimum Design of Structures* (Zaragoza, Spain), edited by S. Hernandez and C. A. Brebbia, Elsevier Applied Science, London, 1993, pp. 639–654.