Genetic Search—An Approach to the Nonconvex Optimization Problem

P. Hajela*
University of Florida, Gainesville, Florida

Principles of genetics and natural selection are adapted into a search procedure for function optimization. Such methods are based on a randomized selection from that restricted region of the design space that yields an improvement in the objective function. Their lack of dependence on function gradients makes these methods less susceptible to pitfalls of convergence to a local optimum. An implementation of the approach to a class of problems in structural optimization with demonstrated nonconvexities or disjointness is discussed in the paper. These examples suggest the effectiveness of the proposed method for such problems. The principal drawback of the method is an increase in function evaluations necessary to locate an optimum. Possible strategies to overcome this limitation are presented.

Introduction

MATHEMATICAL nonlinear programming algorithms provide a significant capability for the automated optimal structural design problem. Several recent publications attest to the generality and versatility of these algorithms in the design of structures subjected to both static and dynamic loads.[1-4] Typical optimum designs require a minimization or maximization of a stated objective and simultaneous satisfaction of several design constraints. The more efficient nonlinear programming algorithms for this class of problems are gradient-based, and require at least the first derivative of the objective function and constraints with respect to the design variables.[5] Such "hill-climbing" algorithms are extremely efficient in locating a relative optimum closest to the starting point in the design space. In designs applications where the design space is known to be nonconvex, the optimum may be obtained by starting the search from several initial points in the design space. Even then, no guarantee exists of obtaining the global optimum.

There is a critical need to examine alternate strategies for optimal design that are not susceptible to the pitfalls of methods of nonlinear programming, and the present paper is perceived as an attempt in this direction. Genetic search methods provide such a capability, and their successful adaptation and implementation in a series of optimal design problems, with either disjoint or nonconvex design spaces, is the subject of the present paper.

The development of the field of genetic algorithms is generally attributed to Holland.[6] However, ideas of analysis and design based on biological evolution may be traced to the early efforts of Rechenberg.[7] These methods have since been adapted for a large number of applications in game theory, induction systems, and other aspects of human cognition, such as pattern recognition and natural language processing.[8-11] Such algorithms are generally regarded in the same category of stochastic search methods as simulated annealing,[12] in that both approaches have their basis in natural processes (simulated annealing derives from principles of statistical mechanics). The potential of genetic algorithms as function optimizers has been a subject of more recent interest and is examined closely in the present paper in the context of structural optimization.

Genetic search methods have their philosophical basis in Darwin's theory of survival of the fittest.[13] A set of design alternatives representing a population in a given generation is allowed to reproduce and cross among the alternatives, with bias allocated to the most fit members of the population. Combination of the most desirable characteristics of mating members of the population results in progeny that are more fit than the parents. Hence, if a measure that indicates the fitness of a generation is also the desired goal of a design process, successive generations produce better values of the objective function. An obvious advantage in this approach is that the search is not based on gradient information, and has, therefore, no requirements on the continuity or convexity of the design space.

The literature in structural design contains numerous examples of problems for which the design space is either disjoint or nonconvex. Admittedly, some of these problems are rather contrived. However, most are physically realistic, and must be approached by methods that represent a departure from the traditional gradient-based algorithms that are widely used. Some specific examples of such problems are encountered in optimal configuration and layout,[14] in beam grillage structures,[15] and in structures that are subjected to harmonic loads and sized explicitly for dynamic response constraints.[16,17] Subsequent sections of the paper present a description of the genetic search algorithm, including its principal components and the commonly accepted terminology. Implementation of the method on a series of problems drawn from the literature serves to illustrate both the advantages and shortcomings of the proposed approach. A discussion of current work aimed at enhancing the efficiency of the method and expanding its applicability in structural design is also included.

Elements of Genetic Search

In Holland's[6] original work, genetic algorithms were characterized by bit string representations of possible solutions to a given problem, and by transformations used to vary and improve these coded solutions. An analogy may be drawn with the natural process of reproduction in biological populations, where genetic information stored in chromosomal strings evolves over generations to adapt favorably to a static or

Received March 2, 1989; revision received Sept. 11, 1989.

*Associate Professor, Department of Aerospace Engineering, Mechanics and Engineering Science. Member AIAA.

changing environment. This chromosomal structure represents the generational memory, and is altered when members of the population reproduce. The three basic processes that affect the chromosomal makeup in natural evolution are inversions of chromosome strings, an occasional mutation of genetic information, and a crossover of genetic information between the reproducing parents. The last process is an exchange of genetic material between the parents, and allows for beneficial genes to be represented in the progeny. Genetic algorithms, in a manner similar to their natural counterpart, use chromosome type representations of possible solutions of the problem to search for improved solutions.

There are essentially three basic components necessary for the successful implementation of a genetic algorithm. At the outset, there must be a code or scheme that allows for a bit string representation of possible solutions to the problem. Next, a suitable function must be devised that allows for a ranking or fitness assessment of any solution. The final, and perhaps most significant component, is the development of transformation functions that mimic the biological evolution process when applied to a population of chromosomal representation of solutions to the problem. Central to these components are questions pertaining to optimal population sizes, lengths of chromosome strings, and frequency with which the transformation functions are invoked. Some aspects of these problems are presented in Ref. 18. These issues are best discussed by relating them to a physical design problem.

Let us consider the design of a statically loaded three-member truss for minimum weight, with constraints on allowable values of stress levels in the members. The design variables are the three cross-sectional areas of the bar elements. To obtain a bit string representation for the problem, each cross-sectional area can be expressed as a fixed length string of 0's and 1's that comprise components of a binary coded number. Other representation codes are possible, although the binary representation was used in the present work. For purposes of discussion, we chose the representation code as a 10-digit binary number, with maximum and minimum values of the design variable corresponding to the maximum and minimum of the binary number. Such a bit string representation is as follows:

$$A^i_{min} = 0000000000$$

$$A^i_{max} = 1111111111$$

A linear scaling can be introduced to convert intermediate values of the binary number into design variable values. Three such binary numbers, each corresponding to a design variable, are then placed end-to-end to create a 30-digit string (string length = 30) of 0's and 1's. This 30-digit string represents one design of the three-member truss. A sequence of such strings can be introduced to construct a population of designs, with each design having a corresponding fitness factor. This fitness factor could be a function of the sum of the structural weight of the design and a penalty associated with any constraint violation in the design.

Once a population of designs is generated—and this could be known designs or designs generated at random—the genetic search can proceed to produce new designs with a higher level of fitness than members of the current population. The first concept in this process is one of reproduction, which is primarily meant to bias the population to contain more fit members and to eradicate the less fit ones. A discussion on how this is implemented in the current work is postponed until a later section.

The second component of genetic search is referred to as crossover, and corresponds to allowing select members of the population to exchange characteristics of the design among themselves. Crossover entails selecting a start and end position on a pair of mating strings at random, and simply exchanging the string of 0's and 1's between these positions on one string with that from the mating string. This is akin to transfer of genetic material in biological reproduction processes facilitated by DNA strings. For string lengths of 20, and crossover sites indicated by an underline, two parents cross to produce two new members. This is illustrated as follows:

$$\text{Parent } 1 = 1101\underline{1}0011110\underline{0}1100001$$

$$\text{Parent } 2 = 0011\underline{0}0101010\underline{1}1100011$$

$$\text{Child } 1 = 11010010101011100001$$

$$\text{Child } 2 = 00111001111001100011$$

Mutation is the third concept in the genetic refinement process, and is one that safeguards the process from a complete premature loss of valuable genetic material during the reproduction and crossover. Stated in terms of the problem under discussion, this corresponds to selecting a few members of the population, determining at random a location on the strings, and switching the 0 or 1 at that location.

The steps described above are repeated for successive generations of the population, until no further improvement in the fitness is attainable. The member in this generation with the highest level of fitness is the optimal design. Implementation of this basic approach for the work reported in this publication is presented in the following section. However, before proceeding with a discussion of the implementation, it is important to draw a distinction between the proposed method and a random walk or random search method. The latter generally reduces to an exhaustive enumeration of the design space. Even adaptive random search methods, which identify favorable regions of the design space, essentially search from a point in the design space. In contrast, the method described here simply uses random choice as a tool to manipulate and direct the search in a region that is most likely to contain the global optimum. Furthermore, the search is conducted from a series of design points simultaneously.

Implementation of Genetic Search

A better understanding of the three basic transformations corresponding to reproduction, crossover, and mutation may be obtained by describing simplistic implementations of each. We assume that a scalar measure of fitness (such as the one described in the previous section) has been assigned and a population pool has been fully characterized. The definition of the population pool includes a specification of the total number of strings in the population, the length of each string, and the maximum number of generations over which the population is allowed to evolve. Additional parameters that must be defined are probabilities of crossover and mutation in the evolution process, p_c and p_m. Studies by DeJong[19] show that values of p_c and p_m of 0.6–0.8 and 0.01–0.02, respectively, perform adequately for most problems.

Let the fitness associated with the ith string be denoted by f_i. We can obtain a sum of these fitness values as $f_{\text{sum}} = \Sigma f_i$. The ratio of individual fitness to the fitness sum denotes a ranking of that string in the population. This ratio is used to construct a weighted roulette wheel, with each string occupying an area on the wheel in proportion to this ratio. The wheel is then employed to determine the strings that participate in the reproduction. A random number generator that determines a pseudorandom number between 0 and 1 is invoked to determine the location of the spin on the roulette wheel. Other forms of string selection are currently under implementation, and will be reported elsewhere. Pairs of strings selected in this manner are denoted as a mating pair, which is then subjected to operations of crossover and mutation to produce a pair of offspring.

Clearly, the selection process in reproduction is biased towards strings with better fitness in the population. The crossover transformation is then invoked on the mating pair.

Since crossover is possible with a probability of p_c only, the decision to actually carry out the operation is made on the basis of a biased coin toss. If a crossover is confirmed by this operation, another random number generator, which returns integer numbers between 1 and stringlength-1, is invoked to determine the crossover sites. The string of numbers between these sites are simply swapped between the parents, as illustrated in a previous section, to create the offspring. The final step is that of mutation and is conducted on a bit-by-bit basis. Once again, the probability of this event is prescribed in p_m, and a biased coin toss is used to determine if this operation must be carried out. If the coin toss calls for a mutation, the 0 or 1 at that site is simply reversed.

This operation is repeated until the total number of strings equals the specified population size. General questions pertaining to the desirability of a fixed population size are currently under study, but do not affect the results presented herein. A code for this implementation of the genetic algorithm was developed in FORTRAN. The program is fairly modular and needs external routines that provide population fitness statistics. For the purposes of using this algorithm in structural optimization, it was linked with a finite element analysis program EAL[20] through a series of pre- and post-processors.

With the description of the data structures and transformation operators central to the genetic algorithm complete, one can proceed to write a concise mathematical statement of the optimization problem. In weight minimization problems of the type discussed in this paper, a typical problem statement is as follows:

$$\text{Minimize } W(d) \tag{1}$$

$$\text{Subject to } g_j \leq 0, \qquad j = 1,2,\ldots,m \tag{2}$$

$$\text{and } d_i^l \leq d_i \leq d_i^u, \qquad i = 1,2,\ldots,n \tag{3}$$

In the present work, the constraints were appended to the objective function $W(d)$ in the form of a penalty function defined as follows:

$$W'(d) = W(d) + R * \Sigma < g_j >^2 \qquad j = 1,2,\ldots,m \tag{4}$$

$$\begin{aligned} <g_j> &= g_j \qquad \text{if } g_j > 0 \\ &= 0 \qquad \text{if } g_j \leq 0 \end{aligned} \tag{5}$$

Here, R is a penalty multiplier that was chosen in successive generations to make the penalty term of the same order of magnitude as the weight. The minimization of weight is converted to a fitness maximization problem by defining a transformation $W^*(d)$ of the form

$$W^*(d) = W_{max} - W'(d) \tag{6}$$

where W_{max} is chosen to be greater than the largest $W'(d)$ in the population; $W^*(d)$ is the fitness of any string.

Nonconvex Problems in Structural Design

The genetic algorithm implementation was successful on a wide range of structural design problems. Results of this study, including extensive numerical experimentation with the algorithm parameters and a more rigorous mathematical treatment of the subject are presented in Ref. 21. Discussions in this paper are restricted to those problems that were used to assess the applicability of the proposed method in the optimal design of structures with known nonconvexities in the design space.

The first problem is a two-beam grillage structure (Fig. 1) subjected to distributed static loads. The design variables are the cross-sectional areas of each beam X_i. These are related to the moments of inertia I_i and section modulus Z_i, by empirical relations obtained by Clarkson.[15]

$$Z_i = (X_i/1.48)^{1.82} \tag{7}$$

$$I_i = 1.007(X_i/1.48)^{2.65} \tag{8}$$

The beams were sized for minimum weight to accommodate constraints on allowable stresses σ^* at the center point and at the location of maximum bending moment along the span. Thus, there are four constraints in the problem with two design variables. The design space for this two-beam grillage is shown in Fig. 2. The nonconvexity in the design space is clearly evident.

A second test problem involves a two-element thin-walled cantilever torsional rod (Fig. 3) subjected to sinusoidal excitation. The wall thickness of each element t_i was chosen as the design variable for the problem. The rod was sized for minimum weight with constraints on the maximum stresses produced during the dynamic torsional displacement. In the absence of damping, the design space is disjoint (Figs. 4 and 5), as a natural frequency of the structure close to the forcing frequency would produce an unbounded response. The disjoint design space is defined in the following expressions for the stress constraints:

$$g_i = 1 - (s_i/s_{max})^2 \tag{9}$$

where

$$s_1/s_{max} = 3T_n t_2(1 - \lambda_e)/\mu \tag{10}$$

$$s_2/s_{max} = T_n[3\lambda_e t_2 + t_1(1 - 2\lambda_e)]/\mu \tag{11}$$

$$\mu = t_2[t_1(1 - 2\lambda_e)^2 - t_2(6\lambda_e - 3\lambda_e^2)] \tag{12}$$

Here λ_e is a nondimensional measure of the forcing frequency, t_i is the thickness of the ith element, and T_n is a function of the amplitude of the torsional load. This problem was first discussed by Johnson.[16] Depending on the natural frequencies of the initial structure, the starting design would be in one of these pockets. A nonlinear programming algorithm converges to the best design inside that pocket.

A third test problem is an extension of the dynamic response problem to a ten-bar truss, shown in Fig. 6. A modal superposition approach was used to obtain the damped dynamic response of the structure to a harmonic excitation. Design variable linking shown in Table 1 was used to reduce the number of design variables in the problem to five. These design variables were cross-sectional areas of the bar elements and were sized for minimum weight, with constraints on the maximum

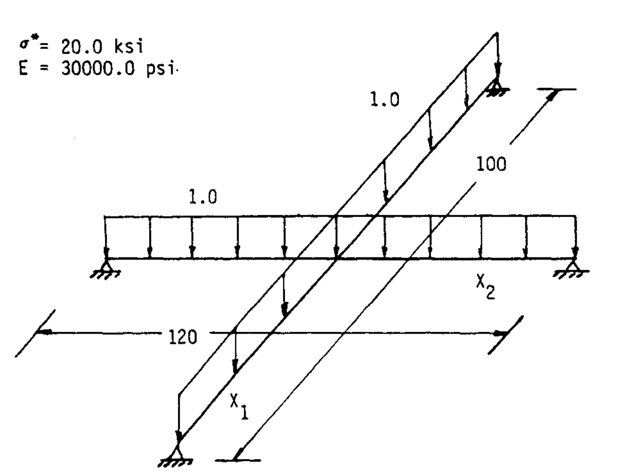

Fig. 1 A two-beam grillage structure.

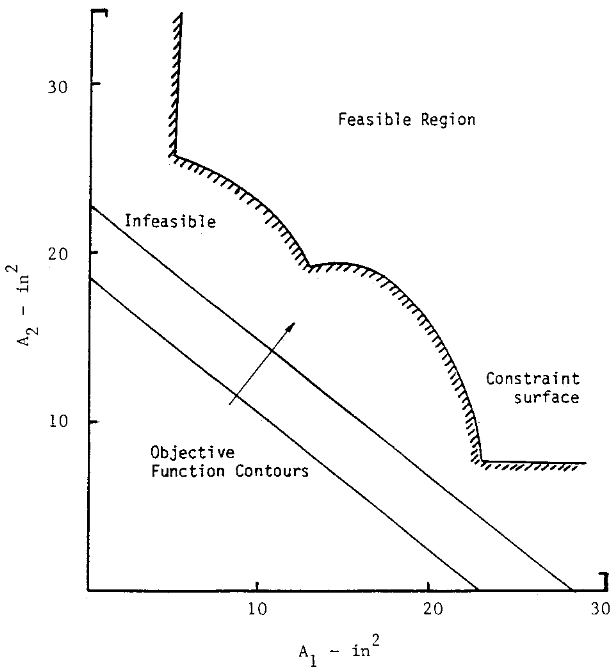

Fig. 2 Nonconvex design space for the grillage structure (Ref. 15).

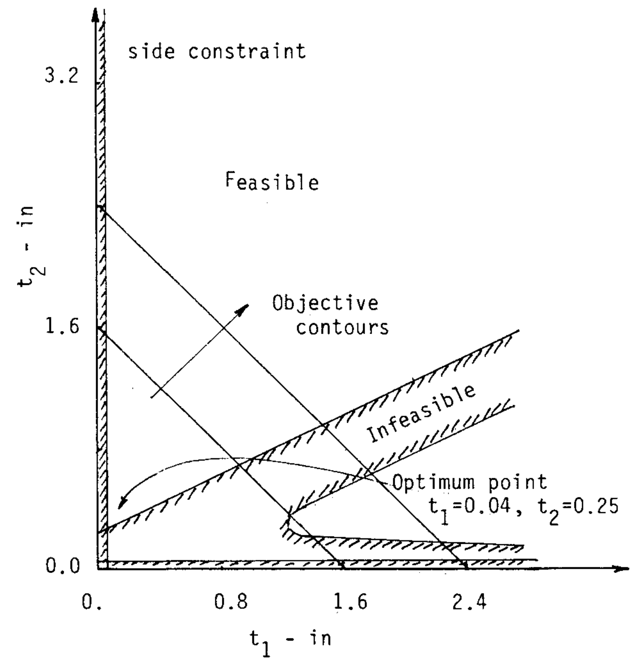

Fig. 4 Disjoint design space for the torsional rod (λ_e = 0.1666667, Ref. 16).

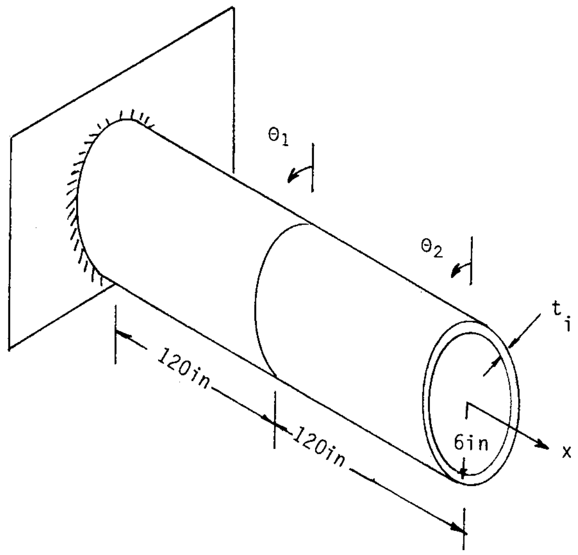

Fig. 3 A cantilever torsional rod.

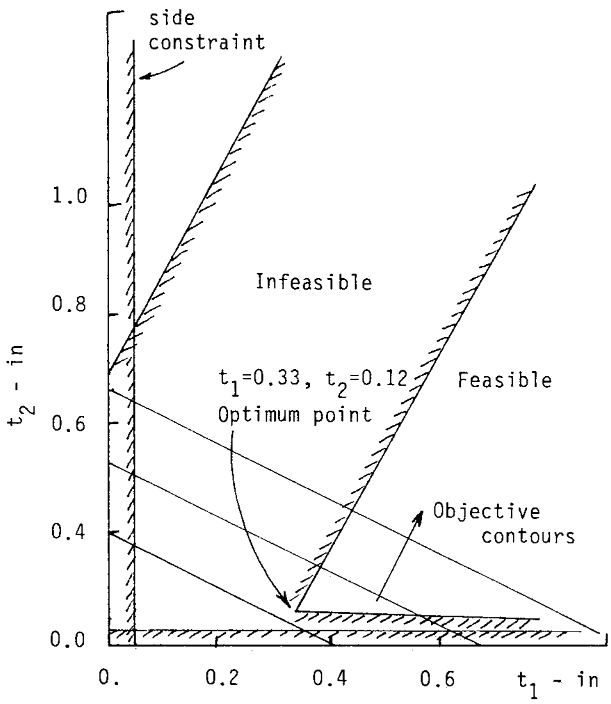

Fig. 5 Disjoint design space for the torsional rod (λ_e = 0.0416667, Ref. 16).

allowable vertical displacements at node 3. The time-variant behavior of the displacements was accommodated in the design constraints by adopting a conservative upper-bound approximation proposed by Mills-Curran.[17] In this problem, the design space is no longer disjoint, as structural damping precludes an unbounded response. However, large responses in regions where the natural frequencies of the structure and the forcing frequency coincide results in severe nonconvexities.

Discussion of Results

The problems described in the previous section were solved using an implementation of the genetic search algorithm discussed earlier. The first problem was the two-beam grillage subjected to a uniformly distributed load. Crossover and mutation probabilities of 0.8 and 0.009, respectively, were used in the problem. Two distinct cases, with fixed population sizes of 60 and 70, and with each member of the population represented by a 60-digit string of 0's and 1's, were implemented in the computer runs. For a two-design variable problem, the latter translates into each cross-sectional area represented by a 30-digit binary coded number. The four stress constraints were appended to the objective function by an initial penalty parameter R = 1000. In subsequent runs, this parameter was increased to a value of 100,000. The initial population was selected at random, and maximum and minimum values of the

Table 1 Design variable definition and final designs for the ten bar truss

Design variable	Element connectivities	Final design-in.^{2}
1	1-2, 5-6	0.6061
2	2-3, 4-5	0.6397
3	2-5, 3-4	3.6671
4	1-5, 2-6	0.8405
5	2-4, 3-5	4.3016

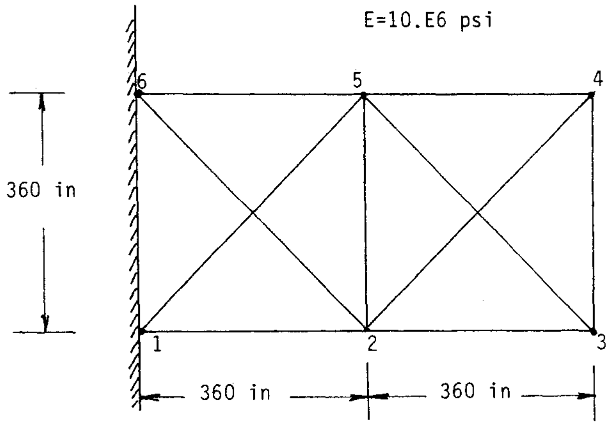

Fig. 6 Ten-bar planar truss.

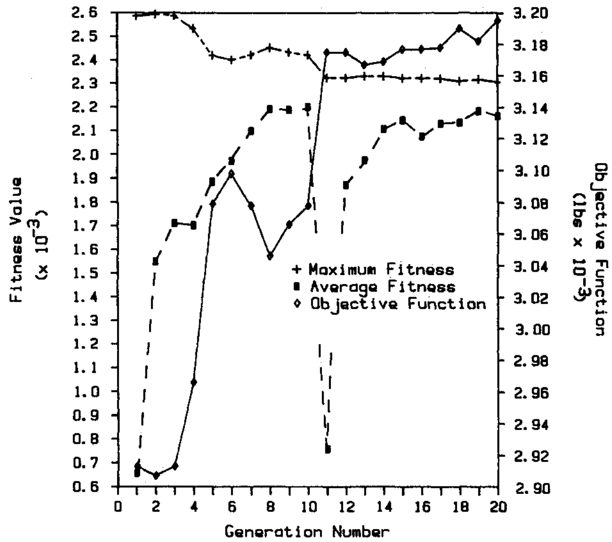

Fig. 7 Convergence histories for the grillage structure.

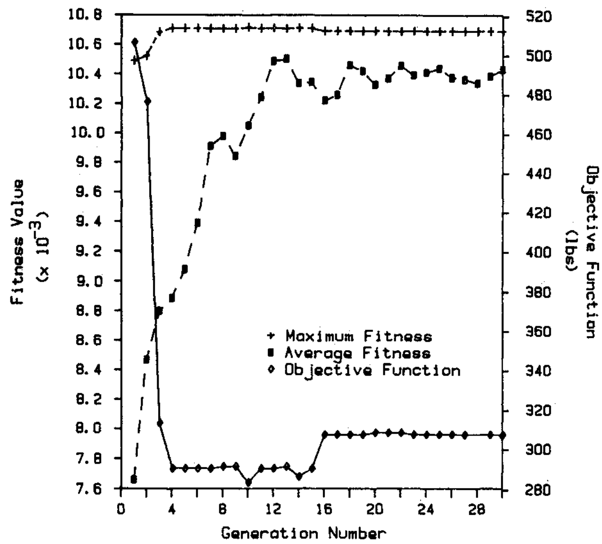

Fig. 8 Convergence histories for the torsional rod (t_{max} = 2.5 in., t_{min} = 0.04 in., λ_e = 0.1666667).

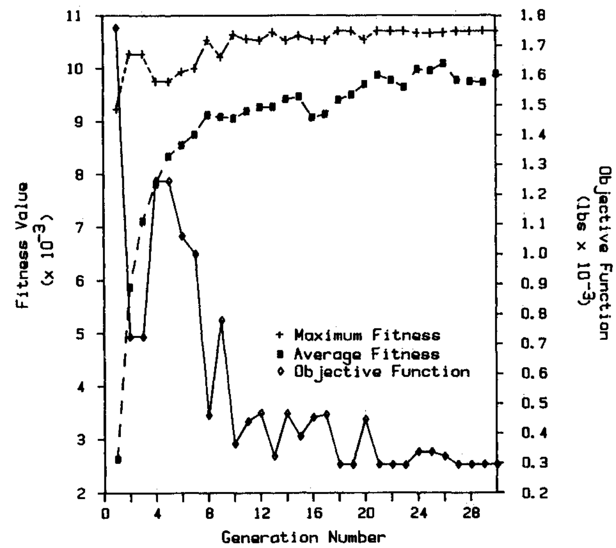

Fig. 9 Convergence histories for the torsional rod (t_{max} = 10 in., t_{min} = 0.04 in., λ_e = 0.1666667).

design variable of 30 in.^{2} and 0.05 in.^{2} were specified. In order to express the weight minimization as a fitness maximization problem, the value of W_{max} was chosen as 10,000 initially, and after 10 generations of evolution, was reduced to 5000.

For both population sizes, a converged result was obtained in about 20 generations. Figure 7 shows typical convergence trends for the maximum fitness, the population average fitness, and the objective function over the generations of evolution. The sharp drop in the average fitness at the eleventh generation is due to a quantum increase in the penalty parameter at that step. A design close to the actual optimum is located rather early in the search process. The optimal design is clearly approached from the infeasible side.

For the second example of a torsional shaft subjected to a harmonic excitation, two design variables were coded for string representation in an identical manner to the first problem. Here, a fixed population size of 70 was chosen for the runs. The penalty parameter was modified from an initial value of $R = 100$ to $R = 10{,}000$ over 30 generations of evolution. The probabilities for crossover, mutation, and W_{max} were kept the same as in the previous problem.

Two cases of initial populations were obtained at random, one by choosing the maximum and minimum values of the design variables as 2.5 in. and 0.04 in. and the other as 10.0 in. and 0.04 in., respectively. Results for these cases were obtained at two different values of $\lambda_e = 1/6$ and $\lambda_e = 1/24$, and are summarized in the convergence histories shown in Figs. 8–10. Figure 10 shows an evolution towards an infeasible design in the first few generations. Significant increase in the penalty parameter corrects this drift back to the feasible side. However, this initial drift towards an infeasible design was directed in the vicinity of the globally optimal design.

The final example is the ten-bar truss, sized for a dynamic displacement constraint. A sinusoidal load of amplitude $F = 5 \times 10^3$ lbs and at a frequency of 14.14 rad/s was applied in the vertical direction at node 3. The first four modes of the undamped structure were used in a modal superposition approach to determine the displacement response. A structural damping of $g = 0.01$ was assumed in the problem, as was an allowable displacement of 0.05 in. A fixed population size of 70 was chosen for the problem, and a starting population was initialized at random. The five cross-sectional area dimensions had prescribed minimum and maximum values of 0.04 in.^{2} and 20 in.^{2}, respectively. The penalty parameter R was chosen as in the second example, as were the probabilities of crossover and mutation. The parameter W_{max} was selected as 10,000 and kept constant for the entire run. The convergence history for the best weight in any given generation is as shown in Fig. 11. The design variables corresponding to the optimal design are shown in Table 1.

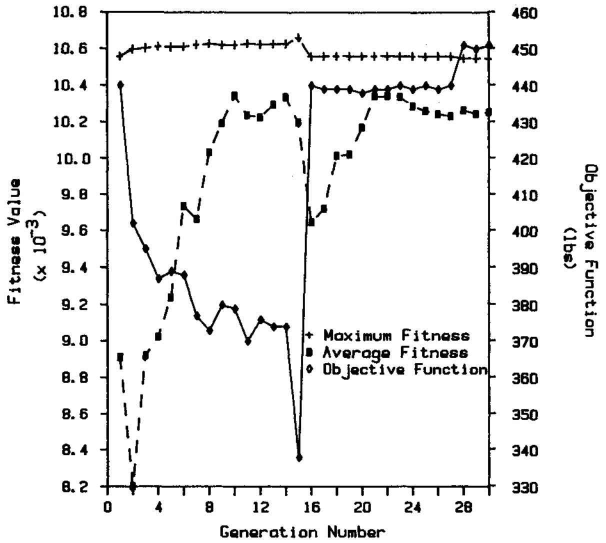

Fig. 10 Convergence histories for the torsional rod (t_{max} = 2.5 in., t_{min} = 0.04 in., λ_e = 0.0416667).

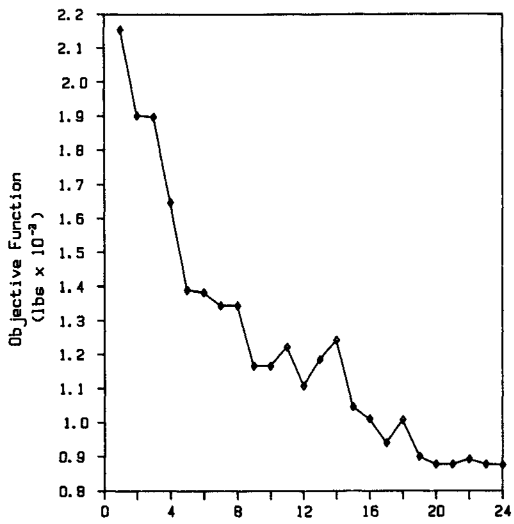

Fig. 11 Convergence history for the ten-bar truss example.

Concluding Remarks

The proposed paper presents an adaptation of genetic algorithms to the optimal structural design environment. Genetic algorithms provide a significant capability for optimal sizing of structural systems. The principal advantages of the approach reside in the fact that no gradients are required in the search process, and, hence, the method is not susceptible to pitfalls of the nonlinear programming hill-climbing schemes. It is important to recognize that even though the method is not plagued by the typical pitfalls of the hill-climbing schemes, it does not guarantee location of the global optimum. Unlike the nonlinear programming approach, the increased dimensionality, due to additional design variables and constraints, does not impose a proportional computational cost. The number of function analyses necessary is directly dependent upon the population size. In the problems attempted in this work, for an average population size of 70, and a typical requirement of 6-7 generations for locating the region of the optimum, approximately 450 function analyses were required. Two to three times the number of such analyses are typically required to obtain a fully converged optimum design. These advantages have been demonstrated by numerical experimentation on problems with known nonconvex and disjoint design spaces.

Extension of this work to large-scale structural systems with discrete design variables, for multiobjective design, and applications in shape synthesis are currently under study. Numerical experimentation has shown that computational requirements for the method can be severe. However, the method has been shown to quickly identify the region of the global optimum. A hybrid scheme that switches from the genetic search to a conventional nonlinear programming approach after a few generations has shown promise in preliminary implementations. Another strategy, wherein the response variables are simply treated as additional design variables and the analysis equations as equality constraints, has also been successful in reducing computational requirements in simple problems.[21]

Acknowledgments

The author gratefully acknowledges the assistance of graduate students Francisco Soeiro and Vidish Rao in preparation of this paper.

References

[1]Hajela, P., "Geometric Programming Strategies for Large-Scale Structural Synthesis," *AIAA Journal*, Vol. 24, No. 7, July 1986.

[2]Fleury, C., "Shape Optimal Design by the Convex Linearization Method," *The Optimum Shape—Automated Structural Design*, edited by J. A. Bennett and M. E. Botkin, Plenum Press, NY, 1986.

[3]Sobieszcanski-Sobieski, J., "Structural Optimization Challenges and Opportunities," International Conference on Modern Vehicle Design Analysis, London, U.K., June 1983.

[4]Hajela, P., and Lamb, A., "Automated Structural Synthesis for Nondeterministic Loads," *Computer Methods in Applied Mechanics and Engineering,* 57, 1986.

[5]Rao, S. S., *Optimization—Theory and Applications,* Wiley Eastern, New Delhi, India, 1979.

[6]Holland, J. H., *Adaptation in Natural and Artificial Systems,* Univ. of Michigan Press, Ann Arbor, MI, 1975.

[7]Rechenberg, I., "Cybernetic Solution Path of an Experimental Problem," Royal Aircraft Establishment, Library Translation 1122, Farnborough, England, 1965.

[8]Stadnyk, I., "Schema Recombination in Pattern Recognition Problems," *Proceedings of the 2nd International Conference on Genetic Algorithms*, Cambridge, MA, July 28-31, 1987.

[9]Sannier II, A. V., and Goodman, E. D., "Genetic Learning Procedures in Distributed Environments," *Proceedings of the 2nd International Conference on Genetic Algorithms*, Cambridge, MA, July 28-31, 1987.

[10]Goldberg, D. E., and Smith, R. E., "AI Meets OR: Blind Inferential Search Using Genetic Algorithms," TCGA Rept. 86002, Univ. of Alabama, Tuscaloosa, AL, Oct. 1986.

[11]Glover, D. E., *Solving a Complex Keyboard Configuration Problem Through Generalized Adaptive Search, Genetic Algorithms and Simulated Annealing,* Lawrence Davis and Morgan Kauffman, publishers, Los Altos, CA, 1987.

[12]Davis, L., and Steenstrup, M., "Genetic Algorithms and Simulated Annealing: An Overview," *Genetic Algorithms and Simulated Annealing,* Lawrence Davis and Morgan Kauffman, publishers, Los Altos, 1987, pp. 1-11.

[13]Sinnott, E. W., Dunn, L. C., and Dobhansky, T., *Principles of Genetics,* McGraw Hill, NY, 1950.

[14]Rozvany, G. I. N., "Optimality Criteria and Layout Theory in Structural Design: Recent Developments and Applications," *Structural Optimization,* edited by G. I. N. Rozvany and B. L. Karihaloo, Kluwer Academic Publishers, 1988, pp. 265-272.

[15]Clarkson, J., *The Elastic Analysis of Flat Grillages,* Cambridge Univ. Press, Cambridge, England, 1965.

[16]Johnson, E. H., "Optimization of Structures Undergoing Harmonic or Stochastic Excitation," *AIAA Journal,* Vol. 14, Feb. 1976, pp. 259-261.

[17]Mills-Curran, W. C., "Optimization of Structures Subjected to Periodic Loads," Ph.D. Thesis, Univ. of California, Los Angeles, CA, 1983.

[18]Goldberg, D. E., "Optimal Initial Population Size for Binary-Coded Genetic Algorithms," TCGA Rept. 85001, Univ. of Alabama, Tuscaloosa, AL, Nov. 1985.

[19]DeJong, K. A., "Analysis of the Behavior of a Class of Genetic Adaptive Systems," Ph.D. Thesis, Univ. of Michigan, Ann Arbor, MI, 1975.

[20]Whetstone, D., "SPAR—Reference Manual," NASA CR-145098-1, Feb. 1977.

[21]Hajela, P., "Genetic Algorithms in Automated Structural Synthesis," NATO Advanced Study Institute on Optimization and Decision Support Systems, Edinburgh, U.K., June 26-July 7, 1989.