Transactions on the Built Environment vol 2, © 1993 WIT Press, www.witpress.com, ISSN 1743-3509

EVOLVE: A genetic search based optimization code with multiple strategies

C.-Y. Lin[a], P. Hajela[b]

[a]*Mechanical Engineering, National Taiwan Institute of Technology, Taipei, Taiwan, Peoples Republic of China*

[b]*Mechanical Engineering, Aeronautical Engineering and Mechanics, Rensselaer Polytechnic Institute, Troy, New York, USA*

ABSTRACT

The present paper describes the capabilities of a modern design optimization tool based on the method of genetic search. This stochastic search technique offers a significantly increased probability of locating the global optimum in a design space with multiple relative optima. The program includes an advanced search technique referred to as directed crossover wherein bit positions on the design strings that offer a higher gain during crossover are assigned higher probabilities of selection as crossover sites. Directed crossover is based on bitwise generational gradient to identify critical bit positions on the string, and provides for reduced schema disruption. This optimization code also includes a multistage genetic search plan which is useful in problems where the design space is large. Multistage search involves successive refinement in the precision with which design variables are represented in the genetic search process. Also included as options in this program are other advanced techniques such as sharing function implementation, mating restrictions, and automatic encoding and decoding of the design variables.

INTRODUCTION

Genetic search based optimization techniques have received recent attention in mechanical and structural optimization problems, and have been proven useful in optimization problems with nonconvex and disjoint design spaces. The approach is applicable to problems with a mix of continuous, discrete, and integer design variables. Genetic algorithms, first proposed by John Holland in 1975 [1], have been adapted for a large number of applications in different disciplines. These methods have their philosophical basis in Darwin's theory of survival of the fittest, and belong to a general category of stochastic search methods. A set of design alternatives which represent a generation in the natural analoque are allowed to reproduce and cross among themselves, with bias allocated to the most fit members of the population. Combinations of the most desirable characteristics of the mating pairs of the population result in progenies that are more fit than either of the parents. If the measure which

indicates the fitness of the population is also the desired goal of the design process, successive generations will result in better objective function values. One characteristic of the genetic search based optimization method is that gradients of objective and constraint functions are not required. This helps in preventing the search from being trapped in a region containing only a local optimum. The application of genetic search in problems with disjoint and nonconvex design spaces is presented in Ref. 2. Genetic algorithms use a chromosome string like binary string to represent an actual design. Irrespective of the type of design variables included in the problem, the design space is a combination of discrete points, each representing an actual design. Genetic search can, therefore, be used to solve design problems with a mix of integer, discrete and continuous design variables. The application of genetic search in mechanical and structural optimization problems with a mix of integer, discrete, and continuous design variables is presented in Reference 3.

A number of implementations of the genetic search approach (GENESIS [4] being one example), each equipped with the standard genetic transformation operators, have been developed. While some of these implementations include convenient programming tools designed to facilitate in the study of genetic search, they are generally cumbersome in their use as multi-purpose function optimization tools. A binary representation of the design variables is left up to the user. In practical engineering optimization problem with one or two dozen design variables, the effort needed to carry out this encoding/decoding process may be nontrivial, particularly in problems with mixed integer discrete variables. It is therefore desirable for a genetic search code to include an automatic encoding/decoding capability, and this is one of the features of the code EVOLVE.

In problems with multiple relative optima in the design space, traditional mathematical programming methods converge to the nearest local optimum. The sharing function approach proposed by Goldberg and Richardson [5], when combined with a mating restriction strategy [6], enables the location of the different relative optima in such problems in a single genetic search. The EVOLVE code includes these search strategies, and has been shown to be effective in locating multiple near-optimum designs in problems with a nonconvex design space. EVOLVE also includes two advanced search strategies referred to as multistage search and directed crossover [7]. The multistage genetic search is a specialized strategy for optimization problems of large dimensionality. In this approach, the precision of design variable representation is gradually increased (i.e. the granularity of design variable representation is decreased) in successive generations of genetic evolution. This is achieved by increasing the length of binary string that represents each design variable, but without the attendant increase in the population size which would be otherwise required [8]. The basic idea behind the approach is to identify promising regions in the design space in earlier generations of evolution with a relatively smaller population size; the search is then refined in those promising regions.

Directed crossover is designed to improve the effectiveness of the crossover transformation in genetic search. In a traditional implementation of the approach, crossover sites on the chromosomal string are selected at random. In directed crossover, however, a bitwise generational gradient is developed to determine which bit strings offer the most potential for improving the design fitness. A bias is developed in

the crossover operation, wherein critical bit positions receive higher probabilities of being selected as crossover sites. Directed crossover has been shown to be specially effective in problems of high dimensionality, and where some design variables are much more significant than others in their influence on the objective and constraint functions.

EVOLVE is written in FORTRAN language and has been successfully tested on a number of computational platforms such as SUN Sparc-2, IBM Risc6000/340, and the DEC 5000/25. Subsequent sections of this paper discuss the genetic search procedure and the special features of EVOLVE in greater detail. Numerical problems illustrating the effectiveness of the program are also included for completeness.

THE EVOLVE CODE

Figure 1 shows the general organization of the EVOLVE code in a schematic form. The basic operations in a genetic search based optimization procedure are indicated by the sequence of commands enclosed within the dashed lines in this figure. The specialized routines in EVOLVE modify the basic operations as necessary. In a basic genetic search, the optimization is initiated with the generation of an initial population of candidate designs. The creation of this first population is usually done with the help of a uniform random number generation algorithm. In addition to the even distribution of designs over the design space achieved by this approach, it is also possible to include a number of known good designs in the initial population. Once the initial population is formed, each design is evaluated and its corresponding fitness function value is obtained. Based on the fitness function values of a given design and the average fitness function of the present generation, each design is assigned a probability of being selected as mating parents. Genetic algorithms apply selection pressure through this process, allowing designs that are more fit in one generation to increase their presence in subsequent generations. In a fixed population simulation of the search, designs with lower fitness are eliminated from subsequent generations. After the selection of mating parents, the operation of crossover is performed on a prescribed percentage of parent designs in order to create offspring in next generation. Crossover is the primary mechanism to introduce new designs into the population. For a pair of mating designs, the popular two point crossover operation involves selecting two crossover sites on each string at random, and then interchanging the binary substrings bracketed by the crossover sites between the mating strings. Mutation can serve as a minor mechanism for the introduction of new designs, and its principal purpose is to guard against preconvergence to a false solution. The operation of mutation is the random selection of a bit on a string, and changing the 0 to a 1 or vice versa; this operation is performed with a low probability (less than 2% of the designs are typically affected). These operations comprise one generation of genetic evolution. The genetic search is typically terminated when the best design does not improve over a given number of generations or when a prescribed maximum number of function evaluations have been performed.

As noted in Figure 1, the basic block of genetic search operators within EVOLVE communicates with several specialized blocks of operations that effectively enhance the flexibility of use as well as add to the overall usefulness of genetic search in problems of practical interest. These special features are described briefly in the text that follows.

A. Automatic encoding/decoding of design variables

Genetic search operates on a chromosome string-like representation of the design variables. While this feature allows handling of a mix of continuous, discrete, and integer design variables, it also requires the encoding and decoding of design variables. An appropriate choice of design variable coding is essential to the success of the procedure, and generally requires some prior experience in the use of genetic search. The EVOLVE code allows the user to select their own coding scheme for design variable representation, or alternatively, the program simply requires that the user specify the expected range of design variable variations and the precision with which the design variables must be represented. The encoding/decoding operations are performed automatically within this program. If the user wants to seed some designs in the initial population, only actual design variable values, instead of the corresponding binary string, of each seeded design are necessary. All best designs in the search and designs in the final generation are reported as actual design variable values. The option does exist, however, to also request these variable in their coded form.

B. Automatic constraint handling

Most genetic search codes are designed to handle only unconstrained optimization problems. Constraints are usually included by transforming the constrained problem into an unconstrained one by the use of exterior penalty function techniques [3]. The user generally defines the pseudo-objective function which is formed by appending all possible penalty terms resulting from violated constraints to the original objective function. In EVOLVE, the user needs only to define each constraint function as an inequality $g_j \leq 0$, and the program automatically forms a pseudo-objective function for each design in the population. The rate at which the penalty due to constraint violations is increased can be specified by the user.

C. Sharing function implementation

The sharing function implementation [5] is essential for locating multiple relative optima in the design space. The sharing function approach is based on a concept of shared resources among distinct subsets of population. Each such subset converges to one relative optimum, and in doing so, maximize its payoff. The principle of sharing is implemented by degrading the fitness of each design in proportion to the number of designs located in its neighborhood through the use of sharing functions. The extent of sharing is controlled by a sharing radius σ_{sh}. The distance metric between two designs can be calculated as:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_{k,i} - x_{k,j})^2} \qquad (1)$$

and the sharing function is defined as follows:

$$\phi(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{sh}}\right)^{\alpha}, d_{ij} < \sigma_{sh} \\ 0, otherwise \end{cases} \quad (2)$$

The fitness of a design i is modified as:

$$f_{si} = \frac{f_i}{\sum_{j=1}^{m} \phi(d_{ij})}. \quad (3)$$

The raw fitness value of each design is decreased due to the presence of a large number of designs in its vicinity. The sharing function implementation will prevent all candidate designs from converging to a single optimum, and help locating multiple optima in a genetic search. A mating restriction scheme is generally used in conjunction with the sharing function implementation for locating multiple relative optima in the design space (Its use in the EVOLVE code is left as an option). As the name suggests, mating restriction prevents two remote designs from being selected as mating partners in the crossover operation. This is achieved by checking the distance metric d_{ij} between two mating designs, and permitting the genetic evolution process to continue only if the distance metric d_{ij} is smaller than a user prescribed value.

D. Multistage search

Although genetic search has been routinely used in optimization problems involving continuous design variables, it is important to bear in mind that the method essentially searches for an optimum from a set of discrete design alternatives. In a large scale optimization problem, the number of design variables is often very large. This typically results in very long binary string representations of the design. In order to search effectively in such a situation, the required population size would have to be proportionately increased. The multistage strategy enables the use of relatively smaller sized populations while retaining the effectiveness of the search. Multistage, varying granularity approach in genetic search essentially involves a successive increase in the precision with which a design space is represented. A coarser representation is first used to identify promising regions of the design space, with the underlying implication that a smaller population size can be used for this purpose. Once the promising regions of the design space are identified, a more detailed search can be conducted with a higher precision in design variable representation. This search with higher precision was implemented in EVOLVE by increasing the string length in successive steps. Even though the design space is expanded by increasing the precision of representation, a proportional attendant increase in population size may not be required, as the previous stage solutions provide good seeds in regions that offer most promise. To counter the possibility of genetic search drifting away from the most likely optimal solutions, a

relaxation of design constraints at earlier stages of the search is implemented in conjunction with the varying granularity approach. More detailed description of this approach can be found in Ref. 7.

E. Directed crossover and mutation

The very nature of binary coding of design variables in genetic search assigns different degrees of significance to different bits of the binary string. In theory, if a string of binary characters of length N is used for representing the design, a population size that is in proportion to the string length would have to be selected. If however, only a smaller fraction, c*N of the bits in the string were really significant to the search process, the population size could be reduced accordingly. The primary motivation behind the directed crossover strategy, therefore, is to identify significant bit position on the string, and to constrain the crossover operation to these bit locations. Two strategies using generational memory of bit-by-bit crossover gain as a guideline in this selection were implemented in the EVOLVE code. The directed crossover can be put into effect after some generations in which regular crossovers are performed and generational gradient information is compiled. More extensive description of these directed crossover plans can be found in Ref. 7.

ILLUSTRATIVE EXAMPLES

There are three illustrative examples included in this section. The first is used to demonstrate the effect of the sharing function implementation which is useful in locating multiple relative optima in a multimodal design space. The second example is a riveted lap joint efficiency maximization problem, where the design space consists of a mix of integer and discrete design variables. The third example involves the sizing of a 25-bar truss in which multistage genetic search and directed crossover strategies were used.

Example 1:

The first illustrative example is to solve an unconstrained multimodal optimization problem, where the objective function is the Himmeblau's function has four distinct optima. This function is defined as follows:

$$f(X) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \tag{4}$$

Lower and upper bounds of design variables x_1 and x_2 were set to -5.0 and 5.0, respectively. The precision of the design variable is 0.01, and as a consequence, the length of a binary string representing a design is 20. A population size of 20 was automatically selected and probabilities of crossover and mutation were set to 0.6 and 0.01, respectively. A regular genetic search was first performed and used as a reference for subsequent operations. The genetic search was terminated after 2,000 function evaluations. A genetic search with an identical initial population was then performed with the sharing function implementation active. Finally, a genetic search using the sharing function approach in conjunction with a mating restriction strategy, was also

performed. The latter two cases involving the sharing implementation used 1.0 as σ_{sh} and 0.2 as α as shown in Eq. (2). Furthermore, in the last case, a radius of 0.2 was used as the mating restriction radius. Both the sharing and mating restriction radii are nondimensional quantities, and the lower and upper bounds were scaled into 0.0 and 1.0. The two cases for which the sharing function implementation was used were allowed to run up to 6,000 function evaluations. The initial population distribution for all three searches is shown in Figure 2. The population distribution at the end of 1,000 and 2,000 function evaluations for the regular genetic search is shown in Figures 3 and 4, respectively. It is clear from Figure 4 that the search has converged on one of four relative optima. The population distribution for both searches involving the sharing function implementation at the end of 50 generations (1000 function evaluations) is shown in Figure 5. It can be seen that although one of four relative optima has been clearly identified, the remaining designs are still spread over the entire design space. The results of continuing this search over more generations of evolution are shown in Figure 6. Even though the trend was to move the designs towards each of the other relative optima, there was a disruptive tendency that prevented a more precise location of these relative optima. When a mating restriction was imposed in the search after the first fifty generations, this disruptive tendency was eliminated, and as shown in Figure 7, each of the four relative optima were precisely identified.

Example 2:

This mechanical design problem involved the design of a lap joint between two steel plates in which the rivet size, and the number and arrangement of the rivet pattern were considered as design variables. The configuration of the plates and the rivets is shown in Figure 8. The number of rows parallel to side AB is represented by an integer variable x_1 with permissible values between 1 and 32. The number of the rivets in each row x_2 was an integer variable and was allowed to assume values between 1 and 128. The diameter x_3 of all rivets was assumed to be the same, and was chosen from a commercially available set

$$x_3 = [6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 27, 30, 33, 36, 40, 45] \text{ mm}$$

It is clear that this variable is of the discrete type. The code EVOLVE was simply provided this physical description of the problem. The choice of binary string representations was determined automatically within the program. The objective of this optimization was to maximize the efficiency of the joint, defined as the ratio of the strength of joint to the strength of the plate. To avoid stress concentrations, two design constraints were imposed on the placement of rivets. Detailed description of the joint strength and constraints is presented in Reference 3. A population size of 30 was selected, and a maximum of 3,000 function evaluations were specified in this search. Probabilities of crossover and mutation were set as 0.8 and 0.01, respectively. In order to transform the maximization problem to a minimization problem, the objective function was defined as 1.0-efficiency. The output history of the search is shown in Figure 9. An optimum design of $x_1^* = 5, x_2^* = 13$ and $x_3^* = 27$ mm was obtained at the

66th generation (after 1410 function evaluations) and the maximum efficiency was 82.45%.

Example 3:

The final example is the 25-bar truss structure shown in Figure 10. An extensive definition of this problem can be found in Reference 8. The performance of multistage search, directed crossover, and a combination of these two strategies was evaluated against a traditional genetic search. This sizing problem includes 25 design variables each representing the cross-sectional area of one truss member. These design variables were assumed to vary discretely between lower and upper bounds of $0.01\ in^2$ and 4.01 , respectively. A spacing of $0.01 in^2$ was assumed in this discrete variation. In Ref. 8, the number of independent variables was reduced to 8 by the use of design variable linking. The mass corresponding to the optimum with 8 design variables is 545.22 lb. which was used as the objective function scaling factor. In the present work, all 25 design variables were allowed to vary independently. Four search plans were executed five times each. The multistage search was divided into four stages. While lower and upper bounds were unchanged for each stage, precision for stage 1 to 4 were set to 0.4, 0.1, 0.05, and $0.01 in^2$, and required binary lengths 100, 150, 175, and 225, respectively. Population sizes were selected as 400 for both single stage searches, and 100 for both multistage search plans. The probabilities of crossover and mutation were prescribed as 0.8 and 0.001, respectively. In each case, the search was terminated after 80,000 function evaluations. In the varying granularity approach, each stage was terminated after 20,000 function evaluations.

The output histories for genetic searches using a normal crossover, the two directed crossover schemes available in EVOLVE, a multistage search, and a combined directed crossover and multistage search are shown in Figures 11a-e, respectively. Figures 11a-c show a similar trend, wherein the average best solutions obtained at 80,000 function evaluations as 1.027 for the normal crossover plan, and 1.015 and 0.983 for the two directed crossover plans. In the multistage and combined approaches, average objective in the first stage was significantly higher than that in two single-stage approaches. The best solutions obtained were 0.996 and 0.974 for multistage and combined plan, respectively. In order to show the effect of directed crossover, the average objective function value and the best objective function value after 8,000 function evaluations for the normal crossover scheme and the directed crossover schemes are plotted together as shown in Figure 12. Both the average and the best objective function values decrease more rapidly with the help of directed crossover, after both share the same history for the first 8,000 function evaluations during which the generational gradients were compiled. It is noted that both multistage and directed crossover approaches result in a further decrease in the minimum weight of about 3.5% in comparison to the result obtained by the traditional genetic search, and the combined approach reduces the minimum weight by about 5%. As shown in Table 1, these strategies obtain the minimum weight obtained at the end of traditional genetic search in 1/2 to 1/10th the number of function evaluations.

CLOSING REMARKS

The present paper describes the features of a general purpose optimization code EVOLVE that is based on the principles of genetic search. The program offers flexibility of use in a large number of routine optimization problems. In addition to the more basic and widely used genetic search operators, the program offers several advanced features that extend the use of genetic search to optimization problems of large dimensionality. The structure of the code is designed to also allow the use of the program in a research environment, particularly to facilitate the testing of new concepts in genetic search. The addition of a structural analysis capability to the EVOLVE programming environment is presently under consideration with the intent of creating an integrated structural design optimization tool.

REFERENCES

1. Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, USA, 1975.

2. Hajela, P., "Genetic Search-An Approach to the Nonconvex Optimization Problem," *AIAA Journal*, 26(7), pp. 1205-1210, July 1990.

3. Lin, C.-Y. and Hajela, P., "Genetic Algorithms in Structural Optimization Problems with Discrete and Integer Design Variables," *Journal of Engineering Optimization*, 19(3), pp. 309-327, 1992

4. Grefenstette, J.J., A User's Guide to Genesis, Technical Report No. CS-84-11, Computer Science Department, Vanderbilt University, Nashville, Tennessee, USA, 1984.

5. Goldberg, D.E., and Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization," Genetic Algorithms and Their Applications: *Proceedings of the 2nd Int. Conf. on Genetic Algorithms*, pp. 41-49, Cambridge, MA, USA, 1987.

6. Deb, K., and Goldberg, D.E., " An investigation of Niche and Species Formation in Genetic Function Optimization," Genetic Algorithms and Their Applications: *Proceedings of the 3rd Int. Conf. on Genetic Algorithms*, pp. 42-50, George Mason University, USA, 1989.

7. Lin, C.-Y. and Hajela, P., "Genetic Search Strategies in Large Scale Optimization,", *Proceedings of the AIAA/ASME/ASCE/AHS/ASC SDM Conference*, LaJolla, California, USA, April 1993.

8. Haftka, R.T., Gurdal, Z., and Kamat, M, *Elements of Structural Optimization*, Kluwer Academic Publications, Dordrecht, 1990.

Table 1: Comparison of Strategy Performance for 25-bar Truss Problem

Strategy	Final Best Objective Function Evaluation	Function Evaluations to Obtain Objective Function of Plain GA
Plain GA	1.027	80,000
Multistage GA	0.996	16,000
Directed Crossover - A	1.015	71,000
Directed Crossover - B	0.983	37,000
Multistage + Directed Crossover - A	0.974	8,000

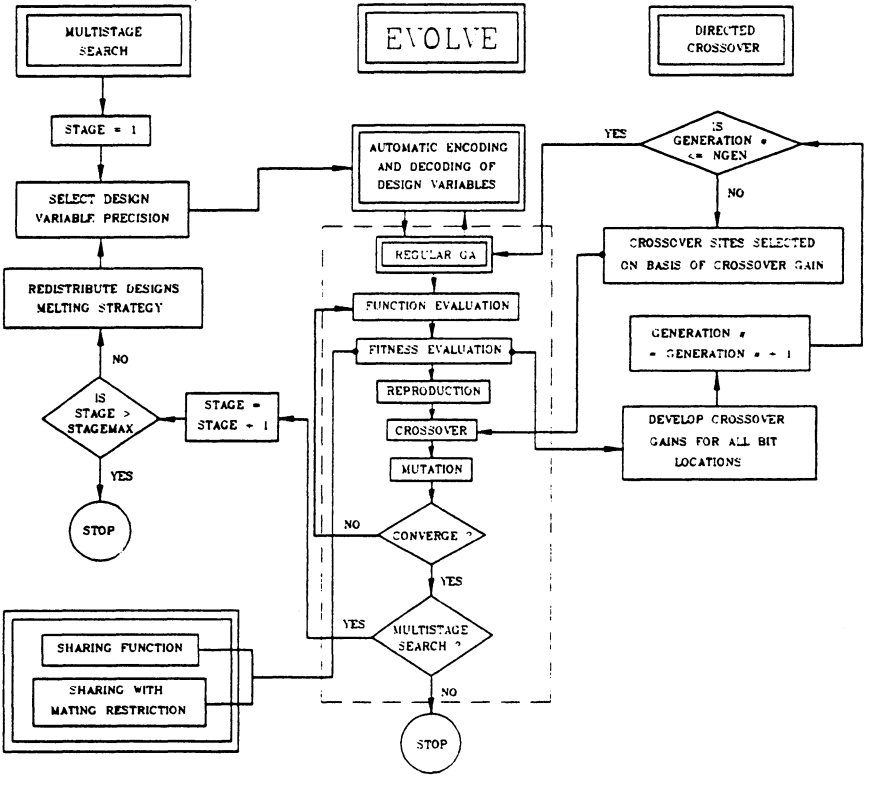

Figure 1. A schematic layout of the EVOLVE code.

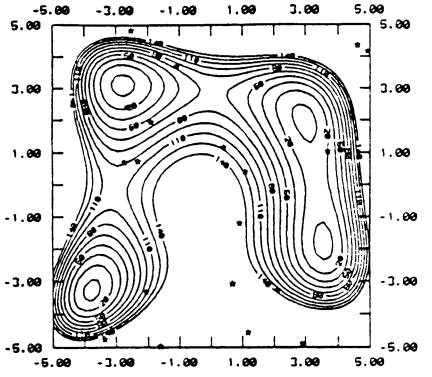

Figure 2. Distribution of designs in the first generation for all genetic searches.

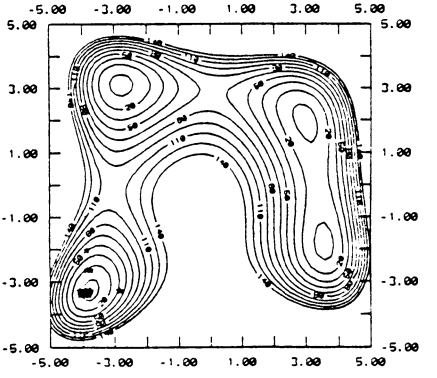

Figure 3. Distribution of designs at the end of 1,000 function evaluations for the regular genetic search.

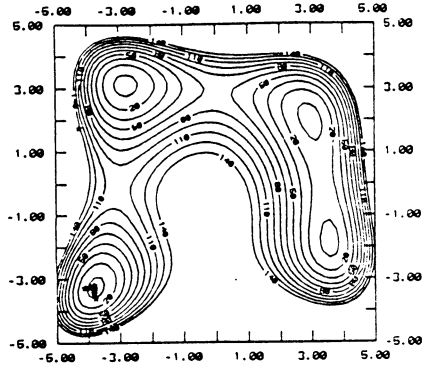

Figure 4. Distribution of designs at the end of 2,000 function evaluations for the regular genetic search.

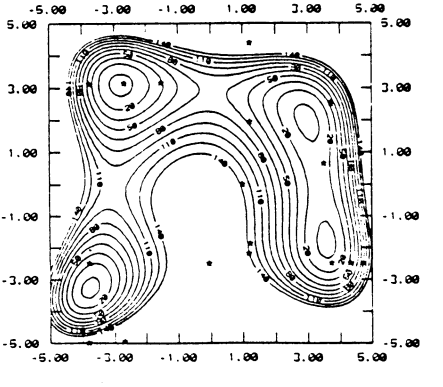

Figure 5. Distribution of designs at the end of 1,000 function evaluations for both genetic searches with sharing implementation.

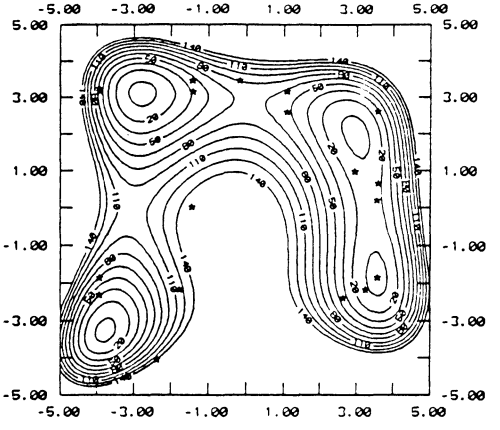

Figure 6. Distribution of designs at the end of 6,000 function evaluations for the genetic search with only sharing implementation.

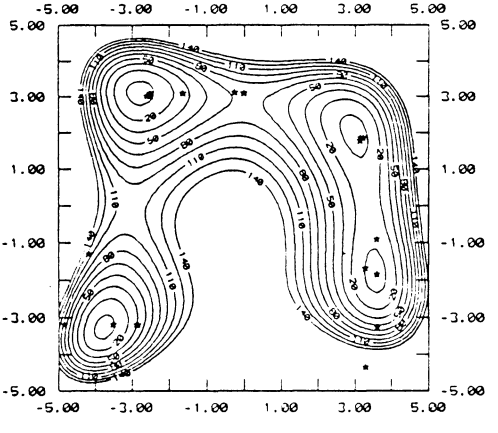

Figure 7. Distribution of designs at the end of 6,000 function evaluations for the genetic search with both sharing implementation and mating restriction.

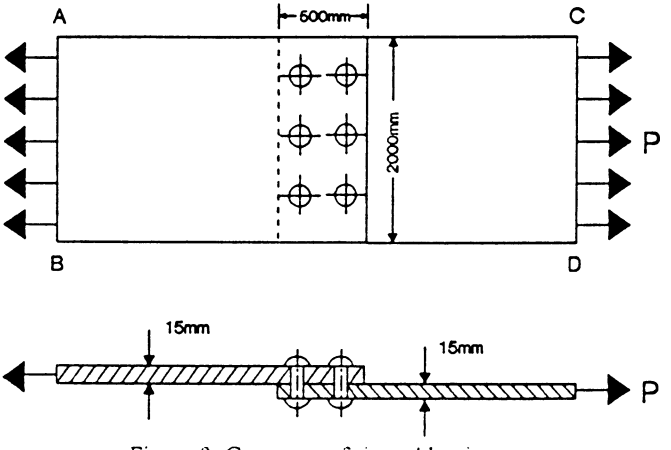

Figure 8. Geometry of riveted lap joint.

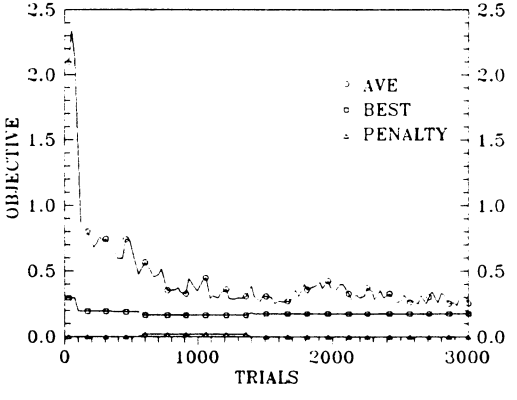

Figure 9. Convergence histories for average and best fitness values for the riveted lap joint.

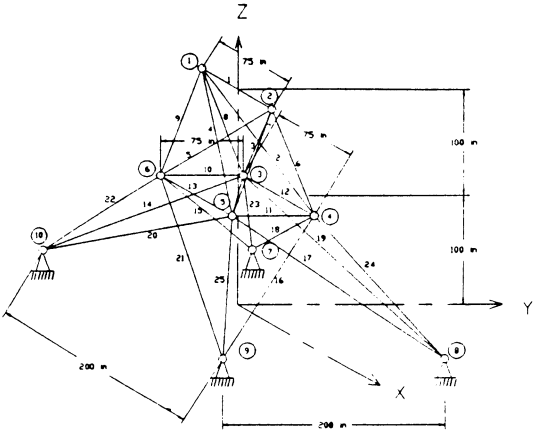

Figure 10. The 25 bar space truss.

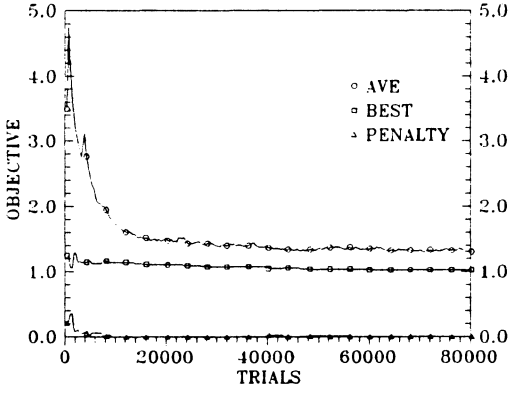

Figure 11a. Convergence histories for average and best fitness values for the 25 bar truss with a regular crossover.

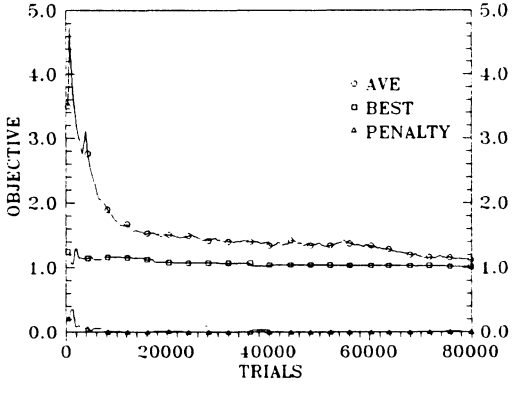

Figure 11b. Convergence histories for average and best fitness values for the 25 bar truss with the directed crossover A plan.

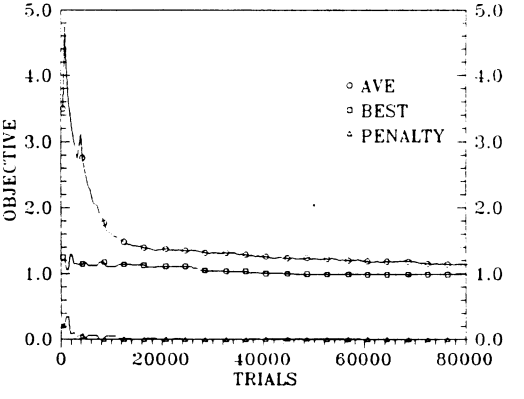

Figure 11c. Convergence histories for average and best fitness values for the 25 bar truss with the directed crossover B plan.

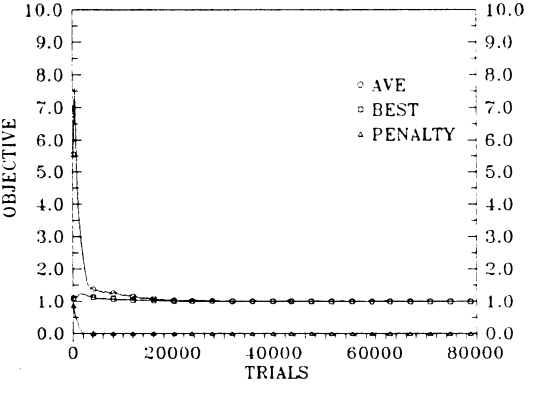

Figure 11d. Convergence histories for average and best fitness values for the 25 bar truss with the multistage search plan.

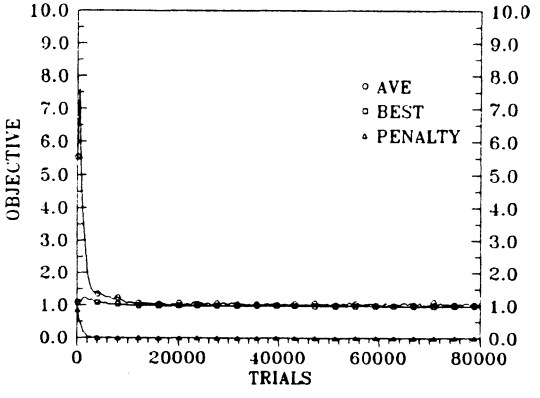

Figure 11e. Convergence histories for average and best fitness values for the 25 bar truss with a combined directed crossover B and multistage search.

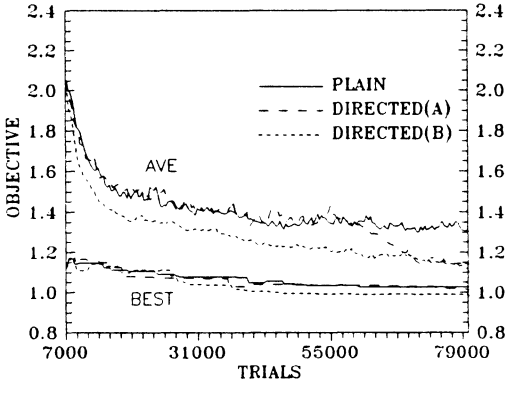

Figure 12. Comparison of convergence histories for average and best fitness values between the regular crossover, directed crossover A, and directed crossover B plans.