State of the Art on the Use of Genetic Algorithms in Design of Steel Structures

S. Pezeshk,[1] M. ASCE and C.V. Camp[2]

INTRODUCTION

During the last three decades, many mathematical programming methods have been developed for solving optimization problems (Gallagher and Zienkiewicz, 1973; Hillier and Lieberman, 1990). However, no single method has been found to be entirely efficient and robust for the wide range of engineering optimization problems (Rajeev and Krishnamoorthy, 1992). Most design applications in civil engineering involve selecting values for a set of design variables that best describe the behavior and performance of the particular problem while satisfying the requirements and specifications imposed by codes of practice. Mathematically these design variables are discrete for most practical design problems. However, most mathematical optimization applications are suited and developed for continuous design variables. In discrete optimization problems, searching for the global or a local optimal solution becomes a difficult task. A few mathematical methods have been developed for solving problems in discrete optimization. These methods include complete enumeration techniques, integer programming, branch and bound algorithms, and dynamic programming. All these methods use mathematical programming techniques.

This chapter presents a genetic algorithm (GA) approach for optimized design of structural systems using both continuous and discrete structural elements. GAs are efficient and broadly applicable global search procedures based on a stochastic approach which relies on a "survival of the fittest" strategy (Holland, 1975). In recent years, GAs have been used in structural optimization by many researchers (Goldberg and Samtani, 1986; Jenkins, 1991a and 1991b; Hajela, 1992; Rajeev and Krishnamoorthy, 1992; Adeli and Cheng, 1993, 1994a, and 1994b; Koumousis and Georgiou, 1994; Rajan, 1995; Camp et al., 1997; Rajeev and Krishnamoorthy; 1997; Pezeshk et al., 1997; Jenkins, 1997; Camp et al., 1998; and Pezeshk et al., 2000 [see the reference list for others]). All these studies have demonstrated that GAs can be powerful design tools for discrete optimization. Because GAs do not require gradient information, they offer a very general approach. This does not mean that useful gradients cannot be

[1]Professor, Department of Civil Engineering, The University of Memphis, Memphis, TN 38152
[2]Associate Professor, Department of Civil Engineering, The University of Memphis, Memphis, TN 38152

exploited by GAs. However, GAs can handle unsmooth, or even randomly ordered data (Jenkins, 1997).

GAs are search algorithms that are based on the concepts of natural selection and natural genetics. GAs differ from traditional optimization methods in the following aspects: (1) GAs work with a coding set of variables and not with the variables themselves; (2) GAs operate on a population of potential solutions rather than improve a single solution; (3) GAs use objective function information without any gradient information; and (4) GAs use a transition scheme that is probabilistic, whereas traditional methods use gradient information (Goldberg, 1989; Hajela, 1992; Rajeev and Krishnamoorthy, 1992; and Pezeshk et al., 2000).

This chapter presents an overview of GAs, discusses various aspects of GAs, and provides a summary of selected research related to structural optimization.

INTRODUCTION TO GENETIC ALGORITHMS

GAs utilize a strategy that models the mechanisms of genetic evolution (Holland, 1975; Goldberg, 1989). The core characteristics of GAs are based on the principles of *survival of the fittest* and *adaptation*. The advantages of applying GAs to the optimized design of structures include discrete design variables, open format for constraint statements, and multiple load cases. GAs do not require an explicit relationship between the objective function and the constraints. Instead, the value of the objective function for a set of design variables is adjusted to reflect any violation of the constraints.

GAs operate on a *population* of design variable sets, with each design variable set defining a potential solution is called a *string*. Each string is made up of a series of characters, typically binary numbers, representing the values of the discrete design variables for a particular solution. The *fitness* of each string is a measurement of performance of the design variables as defined by the objective function and the constraints.

GAs basically consist of a series of three processes: (1) coding and decoding design variables into strings, (2) evaluating the fitness of each solution string, and (3) applying genetic operators to generate the next *generation* of solution strings. The fitness of each string is evaluated by performing some type of system analysis to compute a value of the objective function. If the solution violates constraints, the value of the objective function is penalized.

Most genetic algorithms are variations of the simple genetic algorithm (SGA) proposed by Goldberg (1989). Goldberg's SGA consists of three basic genetic operators: reproduction, crossover, and mutation. The reproduction operation in the SGA is the basic engine of Darwin-

ian natural selection and survival of the fittest (Koza, 1992). The crossover operation creates variations in the solution population by producing new solution strings that consist of parts taken from selected *parent* solution strings. The mutation operation introduces random changes in the solution population. In a GA, the mutation operation can be beneficial in reintroducing diversity in a population.

The objective of the reproduction process is to allow the information stored in strings with good fitness values to survive into the next generation. Typically, each string in the population is assigned a probability of being selected as a parent string based on the string's fitness. However, reproduction does not change the features of parent strings. The next generation of solution strings are developed from selected pairs of parents strings and the application of other explorative operators such as crossover and mutation.

Crossover is a procedure wherein a selected parent string is broken into segments and some of these segments are exchanged with corresponding segments of the another parent string. The one-point crossover implemented in Goldberg's SGA breaks each string of a selected parent string set into two segments and interchanges the second segment to create two new strings.

Mutation is usually used as an *insurance policy* (Goldberg, 1989). Mutation allows for the possibility that non-existing features from both parent strings may be created and passed to their children. Without an operator of this type, some possibly important regions of the search space may never be explored.

Genetic algorithms develop solutions based on the payoff or quality of the fitness of solution strings. A scheme for properly evaluating the fitness is very important in a genetic algorithm. For many structural engineering design problems, a minimum value of the structural weight (related to cost) is desirable. Traditional GAs are designed to work directly with maximum problems; therefore, the minimum problem (low cost of structures) will be converted to a maximum problem.

The Fundamental Theorem of Genetic Algorithms

To explore why simple operators such as reproduction, crossover, and mutation would provide genetic algorithms with robust search power, Holland (1975) proposed a model called the *schema theorem*, or fundamental theorem of genetic algorithms. The schema theorem says that "short, low order schemata are given exponentially increasing or decreasing numbers of

samples depending on a schema's average fitness" (Goldberg, 1989). This theorem can be expressed by the following equation:

$$m(H,t+1) \geq m(H,t) \times \frac{f(H)}{f_{avg}} \left[1 - p_c \frac{\delta(H)}{L-1} - O(H) p_m \right] \tag{1}$$

where $m(H, t+1)$ and $m(H, t)$ are the number of schema H in generation $t+1$ and t, respectively, $f(H)$ is the average fitness value of strings that include schema H, f_{avg} is the average fitness value of the whole population, $\delta(H)$ is the length of schema H, L is the total length of the string, $O(H)$ is the order of schema H, and p_c and p_m are the probabilities of crossover and mutation, respectively. Interested readers are referred to Goldberg (1989) for details of the theorem and a simple example of schema processing.

Formulation of Structural Optimization Problems

The most popular optimization criterion in structural design is cost. Typically, cost is a function of the total structural weight. Other factors that may be involved in estimating the cost of a structure include maintenance (related to the total surface area of a steel structure) and connection costs. An objective function in terms of the properties of both the structure as a whole and individual structural members can be expressed as:

$$F = f(p_m, p_c, p_s) \tag{2}$$

where F is the objective function, p_m are the material properties, p_c are the connection characteristics, and p_s are the structural characteristics.

The general form of structural optimization may be expressed as:

$$\textit{Minimize} \quad F = f(p_m, p_c, p_s) \tag{3}$$

$$\textit{Subject to} \quad g_1 \geq 0, g_2 \geq 0, \quad \ldots \; g_n \geq 0 \tag{4}$$

where g_1, g_2, . . . , and g_n are the constraint functions.

For example, consider a framed structure, where the structural weight is the only term in the objective function and is subjected to stress, displacement, and fabrication constraints. The optimization problem may be expressed as:

$$\textit{Minimize} \quad F = \sum_{e}^{N} \rho_e L_e A(\eta_e) \tag{5}$$

$$\textit{Subject to} \quad F = \begin{cases} s^l \le s \le s^u \\ d^l \le d \le d^u \\ A^l \le A \le A^u \end{cases} \tag{6}$$

where η_e and L_e are the material density and the length of element e, respectively, η_e is the index number referencing an AISC-ASD W-section, as given in Table 1 for element e, $A(\eta_e)$ is the cross-sectional area of the element e, and N is the total number of elements. The vectors s, d, and A contain values of stress, displacement, and cross-sectional area, respectively. The superscripts l and u refer to the prescribed lower and upper boundaries of each constraint, respectively.

Table 1. Relationship between index number η and AISC W-sections.

η	W-section	$A(\eta)$ cm^2 (in^2)	$I(\eta)$ $(10^6)\text{mm}^4$ (in^4)
1	W44 x 335	634.16 (98.3)	12,943 (31,100)
2	W44 x 290	553.55 (85.8)	11,279 (27,100)
⋮	⋮	⋮	⋮
268	W4 x 13	24.71 (3.83)	4.703 (11.3)

For a framed structure, more complicated performance constraints may be imposed on the objective function. For example, the internal forces acting on a member are a function of the properties of that member and any other structural members to which it is connected. Practical serviceability and strength constraints may be more complicated than stress and displacement constraints.

Penalized Objective Function

To evaluate the performance or fitness of a particular solution string, the string's characters are decoded into values of the design variables. Using these design variables, an analysis is performed and a value is computed for the objective function. If any constraints are violated, a penalty is applied to the objective function, with the value of the penalty related to the degree in which the constraints are violated. The resulting penalized objective function quantitatively represents the extent of the violation of constraints and provides a relatively meaningful measurement of the performance of each solution string. In the following sections, several penalty function schemes are proposed for structural design.

Multiple Segment Penalty Function

One of the simplest penalty functions is a multiple linear segment function. Consider a problem where displacement and stress constraints are imposed. Each structural element is checked for stress violation, and each model node is checked for displacement violation. If no violation is found, then no penalty is imposed on the objective function. If a constraint is violated, then the penalty is defined as:

$$\Phi_i = \begin{cases} 1 & if \quad \dfrac{|p_i|}{p_{\max}} \le 1 \\ \dfrac{k_1 |p_i|}{p_{\max}} & if \quad \dfrac{|p_i|}{p_{\max}} > 1 \end{cases} \tag{7}$$

where Φ_i is a penalty value for constraint i, p_i is a structural parameter or response (deflection, stress, etc.), $p_{\max}$ is the maximum allowable value of each p_i, and k_1 is the penalty rate. Figure 1 shows a linear multiple-segment penalty function.

Nonlinear Penalty Function

Another type of penalty function is a nonlinear function defined as:

$$\Phi_i = 1 + k_2 \left(q_i - 1 \right)^n \tag{8}$$

where k_2 is the nonlinear penalty rate, n is the order of nonlinearity, and q_i is defined as:

$$q_i = \begin{cases} 1 & if \quad \dfrac{|p_i|}{p_{\max}} \le 1 \\ \dfrac{|p_i|}{p_{\max}} & if \quad \dfrac{|p_i|}{p_{\max}} > 1 \end{cases} \tag{9}$$

Having obtained penalty function factors, the fitness value of a particular solution string is obtained by multiplying the objective function (structural weight) by the corresponding penalty factors:

$$F = W \prod_{i=1}^{m} \Phi_i \tag{10}$$

$$W = \sum_{e=1}^{N} \rho_e L_e A\left(\eta_e\right) \tag{11}$$

where F is the string fitness (penalized objective function), m is the total number of points where the constraints are checked, W is the weight of the entire structure, and the product represents the total penalty.

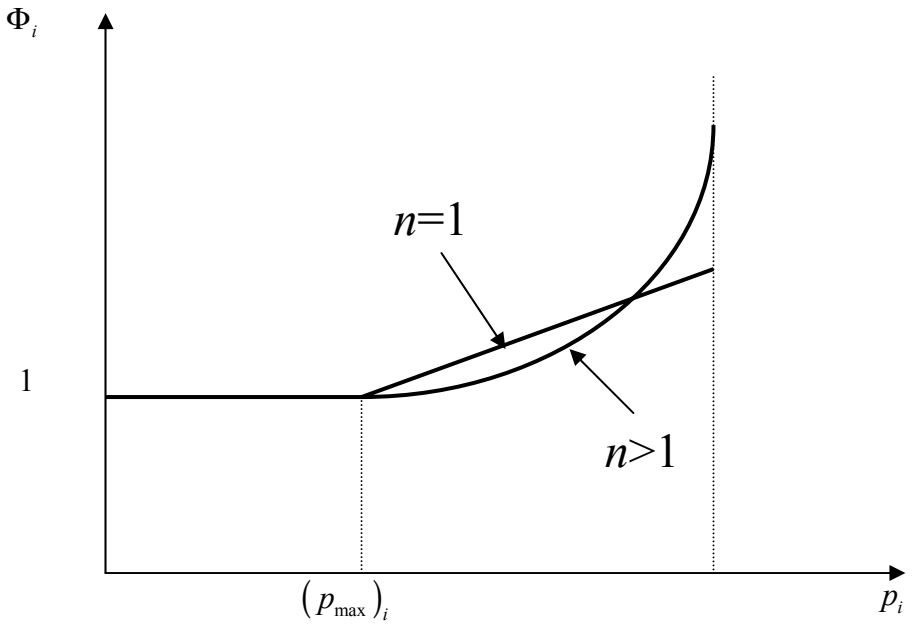

Figure 1. Typical penalty functions.

Coding and Decoding

An essential characteristic of GAs is the coding of the variables that describe the problem. There are many coding methods available, such as binary, gray, non-binary, etc. (Jenkins, 1991a and 1991b; Hajela, 1992; and Reeves, 1993). The most common coding method is to transform the variables to a binary string of specific length.

For a specific problem that depends on more than one variable, the coding is constructed by concatenating as many single variable codings as the number of the variables in the problem. The length of the coded representation of a variable corresponds to its range and precision. By decoding the individuals of the initial population, the solution for each specific in-

stance is determined and the value of the objective function that corresponds to this individual is evaluated. This applies to all members of the population.

Binary Coding

According to Hajela (1992) an *m*-digit binary number representation of a continuous variable allows for 2^m distinct variations of that design variable to be considered. If a design variable is required to a precision of A_c, then the number of digits in the binary string may be estimated from the following relationship:

$$2^m \geq \frac{(X_U - X_L)}{(A_C + 1)} \tag{12}$$

where X_U and X_L are the lower and upper bounds of a continuous variable X. Hajela (1992) suggests that although a higher degree of precision may be obtained by increasing string length, higher degree of schema disruption can be expected. He further mentions that larger defining length schema clearly are at a disadvantage in dominating the population pool. For example, a real variable X whose range is $0.0 < X < 5.0$ can be coded as a 3-digit string using Equation (12):

$$000 \leq X \leq 111$$

There are a total of $2^3 = 8$ points in this range. Of the 2^m possible *m*-digit binary strings, a unique string is assigned to each of the *n* integer variables. In this example, there are six integers between 0 and 5; therefore, there are 2^3 binary strings. Hajela (1992) suggests that the two extra binary values assigned to out-of-bound variables 6 and 7 as follows:

[0, 1, 2, 3, 4, 5, 6*, 7*]
[000,001,010,011,100,101,110*,111*]

where * indicates an out-of-bound variable. Hajela suggests that a penalty measure is then applied to the fitness function of a design variable that includes the out-of-bound integer variable. Another approach would be an one-to-one correspondence between the integer variables and their binary representation. Decoding from a binary number to a real number can be performed using the following equation (Adeli and Cheng, 1994):

$$C = C_{\min} + \frac{B(C_{\max} - C_{\min})}{2^L} \tag{13}$$

where C is the real value of the string, $C_{\min}$ and $C_{\max}$ are the lower and upper bounds of C, L is the length of the binary string, and B is the decimal integer value of the binary string.

Selection Strategies

A simple GA proceeds by first randomly generating a population of solution strings. A pseudo random number generator is used to generate the initial population. From this population, the next generation is evolved by performing three distinct operations: selection, crossover, and mutation. Based on the statistics of this population, the next generation is reproduced according to probabilities assigned to the members. This means that poor designs will be assigned low probabilities, and good designs will be assigned high probabilities of surviving in the next generation. In this way, the next generation evolves, where strings with higher fitness survive and increase their presence, as strings with lower fitness die out and disappear from the generation.

The reproduction process is essentially a selection process. There are a number of selection schemes commonly used in modern genetic algorithms which include proportionate reproduction, ranking selection, tournament selection, Genitor (or "steady state") selection, and greedy over selection. A comparison of the varying schemes has been performed by Goldberg and Deb (1991). As a brief introduction to selection strategies, proportional selection and group selection are discussed. In proportional selection, strings with higher fitness strings receive higher reproduction rates. In group selection, the whole population is divided into groups according to their fitness values. Every member in a group is assigned the same reproduction rate.

To give GAs enough opportunity and information to explore the selection domain, the population should be kept as diversified as possible during the early generations. Extraordinary strings (with high fitness) may exist in initial generations, and these strings could dominate the population. To prevent this possible domination, the fitness of extraordinary strings may be scaled down and the fitness of poor strings may be scaled up. As GAs proceed, the difference of the fitness between extraordinary strings and poor strings will narrow. In this situation, the selection procedure may be like a random walk, with a potential increase in the selection pressure. The above strategy is shown in Figure 2.

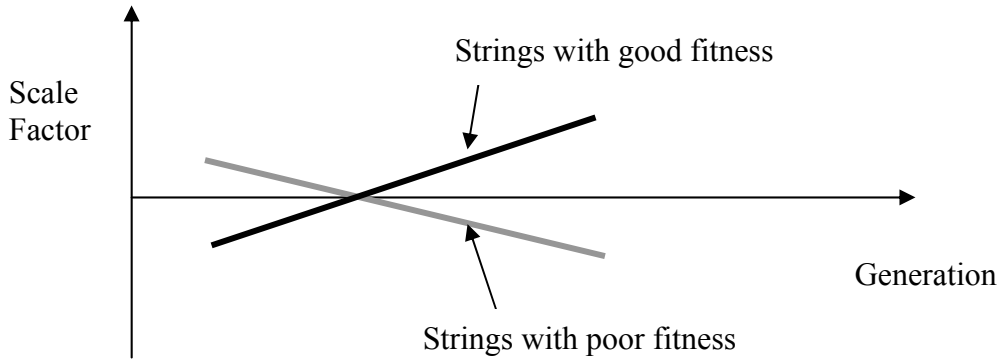

Figure 2. Scaling Reproduction Strategy.

The following is a description of various selection strategies:

Proportional Selection

In proportional selection, the selection probability (P_{si}) is calculated as (Jenkins, 1991):

$$P_{si} = \frac{f_i}{\sum f_i} \tag{14}$$

where f_i is the fitness of the i*th* string and Σf_i is the summation of all of the fitnesses of the population. The result is that strings with higher fitness have higher selection probabilities during reproduction.

Inverse Fitness

The objective function is converted to a fitness function by taking the reciprocal of the objective function. The inherent nonnegative objective functions allow this. The raw fitness will be:

$$f = \frac{1}{W} \tag{15}$$

where W is the objective function. For example, W is the weight of a structure that is to be minimized.

Fitness Scaling

In early generations, it is common to have few extraordinary strings in a population. For example, if the selection rule given by Equation (14) is utilized, extraordinary strings may take over a significant proportion of the finite population in a single generation, and this can be undesirable and may result in premature convergence (Dhingra and Lee, 1994). In later generations, there may still be significant diversity within the population; however, the population's average fitness may be close to the population's best fitness. If this situation is unchanged, a string with average fitness and a string with high fitness will have nearly the same numbers in future generations. In this case, the survival of the fittest strategy necessary for improvement becomes a random walk.

One useful scaling procedure is the linear scaling. The linear scaling procedure requires a linear relationship between the scaled fitness, f', and the raw fitness, f, as (Dhingra and Lee, 1994):

$$f' = af + b \tag{16}$$

The coefficients a and b may be chosen in a number of ways; however, in all cases it is required that the average scaled fitness f'_{avg} be equal to the average raw fitness f_{avg} because subsequent use of the selection procedure will insure that each average population member contributes one expected offspring to the next generation. To control the number of offspring from a parent string with maximum raw fitness, the other scaling relationship to obtain a scaled maximum fitness, $f'_{max} = C_{mult}\, f_{avg}$ is chosen, where C_{mult} is the number of expected numbers desired for the best population member. For small populations (50 to 100) $C_{mult} = 1.2$ to 2 has been used successfully (Chen, 1997). In later generations, this choice of C_{mult} stretches the raw fitness significantly. This in turn causes difficulty in applying the linear scaling rule as shown in Figure 3. It can be observed during early generations, that there is no problem applying the linear scaling rule, because the few extraordinary strings get scaled down and the poor strings of the population get scaled up. The more difficult situation is shown in Figure 4. If the scaling rule is applied in this situation, the stretching required on the relatively close average and maximum raw fitness values causes the low fitness values to become negative after scaling. To circumvent this scaling problem, Forrest (1985) suggested using population variance information to preprocess raw fitness values prior to scaling. In this procedure, which is called sigma (σ) truncation, a constant is subtracted from raw fitness values as follows:

$$f' = f - \left(f_{avg} - c\sigma\right) \tag{17}$$

where the constant *c* is chosen as a reasonable multiple of the population standard deviation σ (between 1 and 3) and negative results ($f' < 0$) are set to zero. Following sigma truncation, fitness scaling can proceed as described without the danger of negative results.

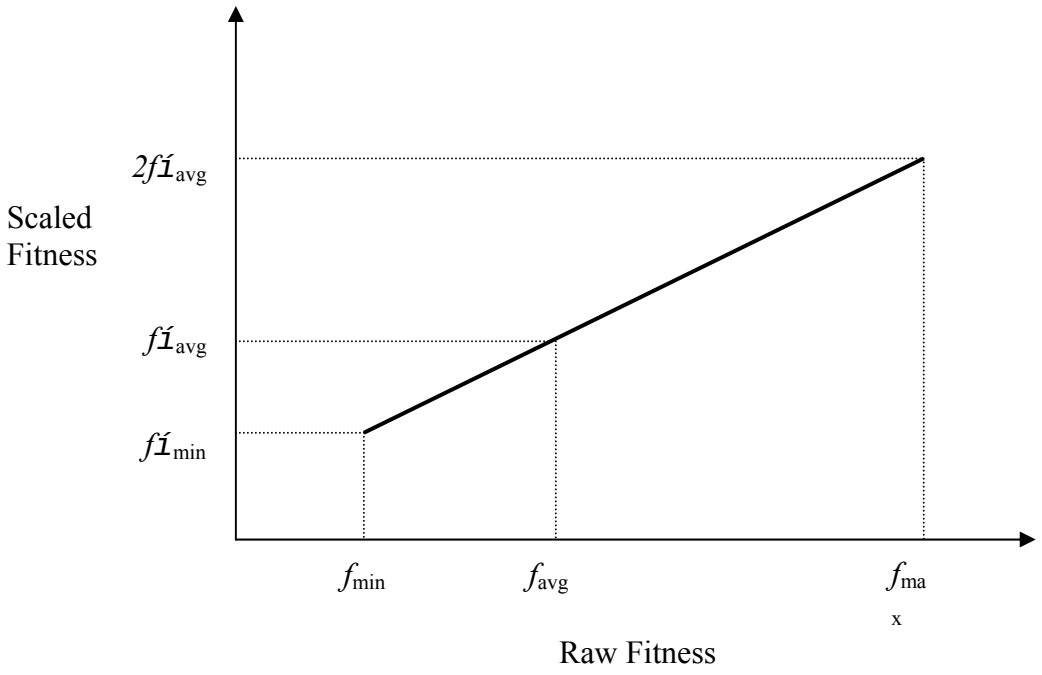

Figure 3. Linear scaling under normal conditions.

The scaling steps can be summarized as:

Step 1: Calculate the objective function *W;*

Step 2: Calculate the fitness function $f = 1/W;$

Step 3: Shift the fitness $f' = f - (f_{avg} - c\,\sigma);$

Step 4: Calculate the coefficients *a* and *b* (also see Dhingra and Lee, 1994) where

$$a = \frac{(c_{mult} - 1)}{f'_{\max} - f'_{avg}} f'_{avg} \tag{18}$$

$$b = \frac{(f'_{\max} - C_{mult} f'_{avg})}{f'_{\max} - f'_{avg}} f'_{avg} \tag{19}$$

Step 5: Calculate the scaled fitness:

$$f''_i = af'_i + b \tag{20}$$

Step 6: Calculate the selection probability:

$$P_{si} = \frac{f''_i}{\sum f''} \tag{21}$$

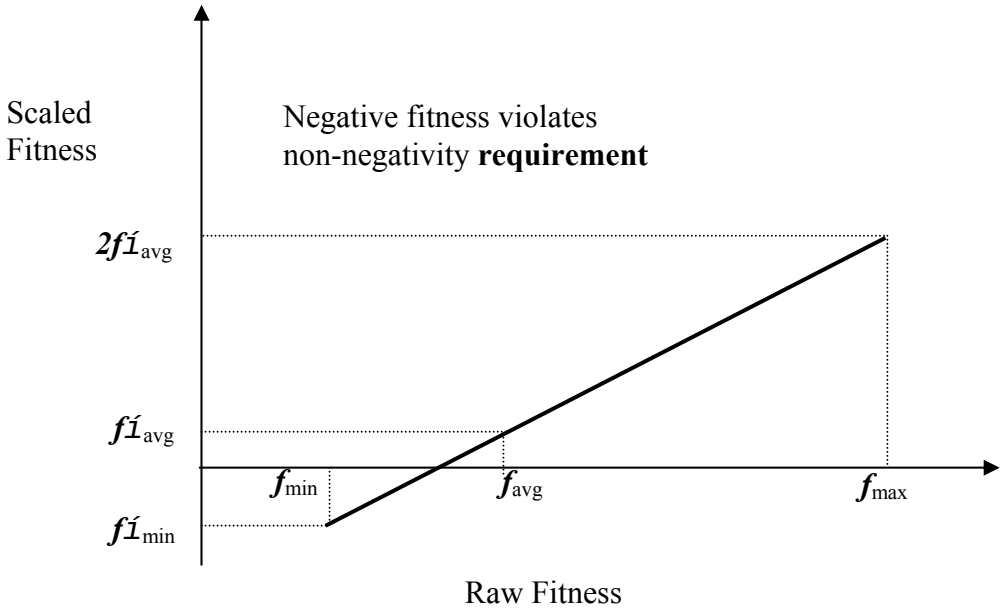

Figure 4. Difficulty with linear scaling procedure in later generations.

Group Selection

If the fitness values of strings in the population are close to each, then a proportional selection scheme may cause slow convergence. In group selection, the population is sorted according to fitness. Then, the sorted population is divided into several groups, where each group is assigned a selection probability. The selection probability for a specific group equals the group selection probability divided by the number of string in that group. The group selection

scheme is illustrated in Figure 5. In this figure, the whole population is divided into two groups, and a specific selection probability is assigned to each group. For example, the first group occupies 30% of the whole population and is assigned a 0.75 group selection probability, the second group occupies 70% of the whole population with the selection probability of 0.25. If there are 10 strings in the population, the selection probability of an individual in the first group will equal 0.75/(0.3x10) = 0.25. In essence, group selection assigns greater probabilities of existence to the best individuals of the population for the next generation.

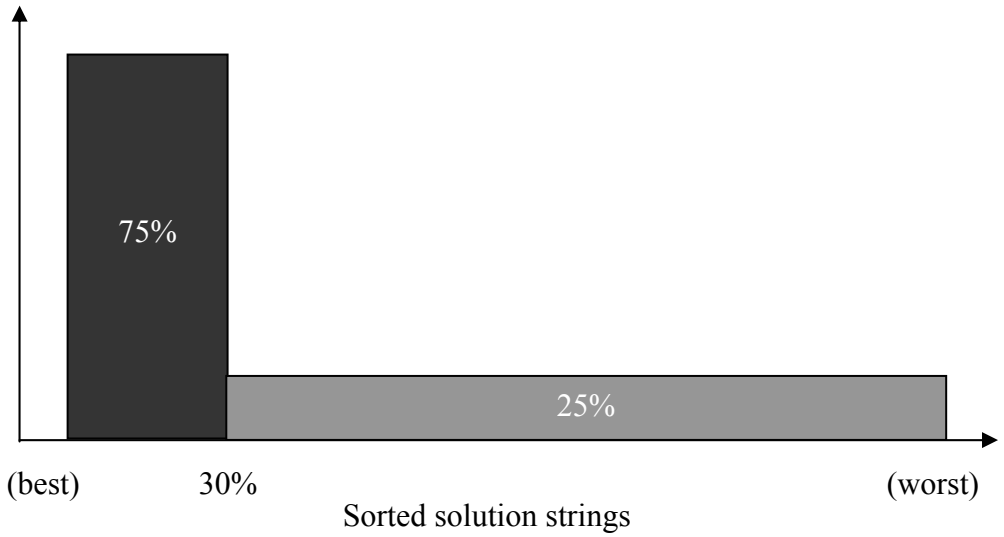

Figure 5. Illustration of group selection.

Crossover

One of the most important operators in GAs is crossover. Crossover is a means for two strings (parents) to produce two offspring by mixing and matching their desirable qualities through a random process. After reproduction, crossover proceeds in two steps: (1) two strings are selected; (2) segments of each string are chosen at random (segment length and location), and the information contained in these segments is exchanged between the two strings. Several methods can be used for choosing the length and location of exchange sites. In this chapter, one-point, two-point, and uniform crossover methods are presented. Figure 6 shows two chromosomes that will be used to illustrate different crossover methods.

String 1 ———— 0101010101000011

String 2 ———— 1111111000111000

Figure 6. Example Strings.

One-Point Crossover

To perform one-point crossover, one crossing site along the string is selected at random. Figure 7 illustrates an example crossover where the crossover site is at the sixth bit.

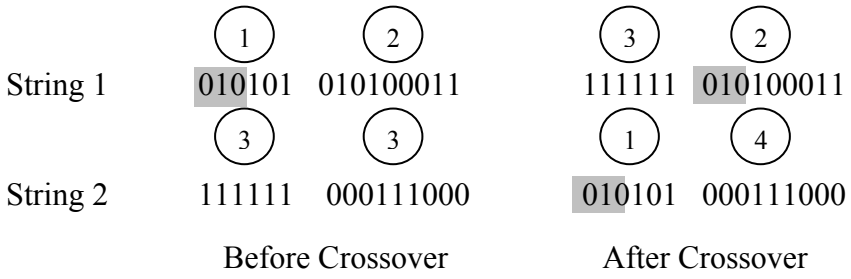

Figure 7. One-point crossover.

Two-Point Crossover

For two-point crossover, two crossing sites are selected randomly. Figure 8 illustrates a two-point crossover using crossing sites at the fourth and tenth bits.

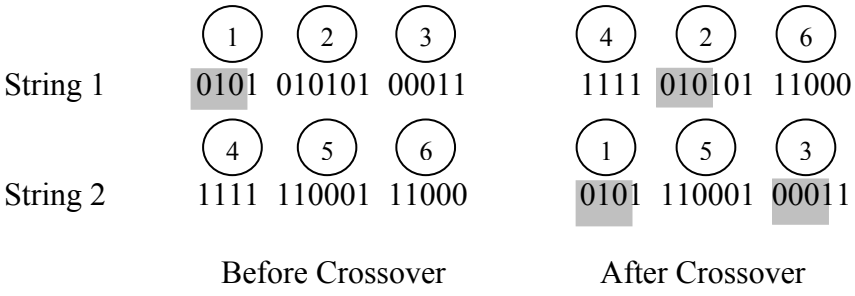

Figure 8. Two-point crossover.

Uniform Crossover

Uniform crossover is based on a randomly created binary string, called a mask (Syswerda, 1989). A mask acts like a sieve. Parent strings exchange their bits at the positions where the corresponding position in the mask is zero. Otherwise, no exchange of bits is performed. The percentage of exchanged bits between two parent strings can be varied from 0% to 50% by selecting the percentage of zeros in the mask string. Figure 9 illustrates how the 40% uniform crossover operation works (note that the mask string contains 40% zeros).

Figure 9. Two-point crossover.

Although the uniform crossover site is chosen by random selection, crossover is not the same as a random solution through the search space. Since crossover is based on the reproduction process just described, it is an effective means of exchanging information and combining portions of strings.

Reproduction and crossover are very simple operations. Their implementation simply requires generating random strings, making copies of the strings (reproduction), and swapping portions of the strings (crossover). However, the reproduction and crossover operators together give GAs the ability to perform complex and difficult optimizations.

Adapting Crossover

In GAs, there are many different forms of crossover. Traditionally, GAs have relied upon one- and two-point crossover operators, but there are many situations in which having a higher number of crossover points is beneficial (Syswerda, 1989 and Eshelman et al., 1989). Perhaps the most surprising result (from a traditional schema-based perspective) is the effec-

tiveness of uniform crossover. Uniform crossover produces, on average, *L/2* crossings on string length *L* (Syswerda, 1989 and Spears and De Jong, 1991).

In addition to empirical investigations, considerable effort has been directed toward theoretical comparisons between different crossover operators (De Jong and Spears, 1992). But these theories are not sufficiently general to predict when to use crossover, or what form of crossover to use. For example, the theories do not consider population size, yet population size can affect the relative utility of crossover operators (De Jong and Spears, 1990).

The current theory of GAs is inadequate for determining which types of operators to use for a particular problem *a priori*. There are at least two possible approaches to this problem: the first approach is to extend the current theories to take into account all facets of GAs (Nix and Vose, 1992; and De Jong et al., 1994; the second approach is to have an adaptive mechanism in which GAs select the type of operator to be used. For example, an adaptive mechanism could choose between four different forms of crossover: one-, two-, three-point, and uniform. One- and two-point crossovers are the least disruptive to the population, while uniform crossover is the most disruptive operator (De Jong and Spears, 1992). Thus, it is natural to allow GAs to explore a relative mixture of these operators.

One obvious way for the GA to self-adapt its use of different crossover operators is to append two bits to the end of every individual in the population. Suppose "00" refers to one-point crossover, "01" to two-point crossover, "10" to three-point crossover, and "11" to uniform crossover. Then, the last two columns of the population (the last two bits of every individual) are used to sample the crossover operator space. If uniform crossover moves the search into solution spaces with high fitness, then more "11"s should appear in the last two columns as the GA evolves. If higher fitness solutions are found using two-point crossover, more "01"s should appear accordingly. Because the approach is self-adaptive, crossover and mutation are allowed to manipulate these extra two columns of bits.

Techniques for Adapting Crossover

There are two possible techniques for using these extra bits: local and global adaptation. In local adaptation, the last two bits of each string are used to select which type of crossover is used. For example, suppose two strings chosen for crossover. The last two bits of each string are then examined. If these bits are identical, "00", then one-point crossover is performed. If the two bits in each string are "11", uniform crossover is performed. If the last two bits are different, the crossover operator is randomly chosen from the four methods (Chen, 1997).

With local adaptation, the choice of crossover operator is determined by a particular string. With global adaptation, the choice of crossover operator is determined by the entire population. For example, suppose that 40% of the population has "00" as their last two bits, and 25% of the population has "11". Then, when crossover is performed on the population, one-point crossover should be applied 40% of the time, and uniform crossover should be applied 25% of the time.

Since the extra two bits are used to determine which crossover operator to apply, the mechanism should give greater reward to the crossover operator that produces superior offspring. Note that this mechanism allows GAs to adjust the relative mixture of crossover operators. Consequently, adaptive crossover can reward a sequence of operators that cooperate to produce a good offspring.

Mutation

Another important operator used in GAs is called mutation, which mimics the phenomenon of natural mutation. When mutation is applied to a string, it sweeps down the string of bits, and changes the bit from 0 to 1 or from 1 to 0 if a probability test is passed. Mutation has an associated probability parameter that is typically quite low. By itself, mutation is a random walk through the string space. When used sparingly with reproduction and crossover, it is an insurance policy against premature loss of important information (Goldberg, 1989). Table 2 contains an example of the operation of mutation. From this table, it can be observed that two parent chromosomes of length 5, randomly generated numbers used for the mutation probability check, and the resulting mutated chromosomes. From Table 2, it can observed that for the first chromosome, the probability test is never passed, and so in this case, the output of mutation is the same as the input. In the second case, the probability test is passed for the fourth bit. Thus, this bit is changed from 0 to 1.

Table 2. Examples of mutation.

Old chromosomes				Random numbers				New chromosomes			
1	0	1	0	0.702	0.325	0.245	0.373	1	0	1	0
0	0	1	0	0.802	0.471	0.023	**0.001**	0	0	1	1
Note: Mutation probability 0.002											

BASIC PARAMETERS OF GENETIC ALGORITHMS

Basic parameters of GAs include: population size, probability and type of crossover, and probability and type of mutation. By varying these parameters, the convergence of the problem may be altered. Thus, to maintain the robustness of the algorithm, it is important to assign appropriate values for these parameters. Much attention has been focused on finding the theoretical relationship among these parameters. Schwefel (1981) developed theoretical models for optimal mutation rates with respect to convergence and convergence rates in the context of function optimization. De Jong and Spears (1990) presented theoretical and empirical results on the interacting roles of population size and crossover in genetic algorithms. Cvetkovic and Muhlenbein (1994) investigated the optimal population size for uniform crossover and truncation selection.

One problem is that the relationships presented in these works are based on specific simplified problems. Thus, the relationships cannot be used in practical problems. In view of the mathematical difficulties involved, experience and experimentations are needed to determine these parameters. Usually, the population size and the probability of mutation are related. With a larger population size, the probability of mutation is smaller. For a wide range of problems, the following values are good estimates for an initial run. For a population size of 30–50, a probability of crossover P_c of about 0.6 and a probability of mutation P_m less than 0.01 is typical. In general, mutation is important in evolutionary computing, because it can bring new strings into the population. Therefore, the initial population, which might have been very far from the satisfactory solution, can adapt itself toward the optimized solution. Conversely, mutation tends to disorganize the convergence of the problem; therefore, the mutation rate, in conjunction with the population size, is crucial to the overall performance of GAs.

RELATED RESEARCH FINDINGS

In recent years, GAs have been used for solving a variety of engineering design problems, and the focus of this section is to provide a summary of some of new developments in structural optimization using GAs. The first contribution to optimal design of structures is the work of Goldberg and Samtani (1986). They discussed the optimal design of a 10-bar truss structure. Jenkins (1991a and 1991b) discussed the optimization of truss-beam roof structures using GAs. Hajela (1992) presented the use of GAs and simulated annealing (SA) methods in structural optimization. Hajela presented various features of both procedures and presented optimal design of relatively large structural systems such as trusses. In addition, Hajela discussed the application of GAs to generate a family of Pareto optimal designs for multi-criterion optimization problems. Lin and Hajela (1992 and 1993) used discrete design variables to find

the minimum weight of an 8-bar truss subjected to displacement constraints. In addition, they discussed the design of 25-member and 72-member truss structures with stress constraints.

Rajeev and Krishnamoorthy (1992) discussed the concept of optimization using a GA for a 3-bar truss problem. They presented all the computations for three successive generations in the form of tables for easy understanding of the problem. In addition, they used GAs to design a 160-bar transmission tower.

Sakamoto and Oda (1993) presented an optimization technique for layout of truss structures using a GA. They used a hybrid method composed of a GA and the generalized optimality criteria method to optimize the layout and cross-sectional area of truss members simultaneously. The objective function is to minimize weight subjected to displacement constraints. In this approach, a GA was used to handle the layout of the truss and the optimality criteria was used to find the member cross-sectional area.

Hajela (1993), Hajela and Lee (1993), and Hajela and Lee (1995) used GAs in the topological design of grillage structures. They used a two-stage optimization process: (1) in the first stage, they used kinematic stability requirements to identify stable topological configurations, and (2) in the second stage, they considered member adding/removal and resizing.

Dhingra and Lee (1994) used a GA in obtaining single- and multiple-objective design problems. They presented several examples dealing with optimum design of truss structures with discrete-continuous variables.

Adeli and Cheng (1993) used a GA for optimal design of space truss structures. Later, Adeli and Cheng (1994a) presented a GA procedure for optimization of three-dimensional truss structures using the augmented Lagrangian to transform the constrained problems to an unconstrained problem.

Grierson and Pak (1993) addressed the sizing, shape, and topology of frame structures. This was one of the first papers to examine the design of framed structures using GAs, and their research used re-analysis techniques to reduce computational effort during any generation, with rigorous analysis conducted on the best design for each successive generation.

Koumousis and Georgiou (1994) used a GA in discrete optimization of steel truss roofs. In their approach, Koumousis and Georgiou designed steel roofs using mixed layout and sizing optimization, and determined that a population size between 10 to 50, crossover ranging between 0.6 and 0.8, and mutation less than 0.01 will provide satisfactory designs for a typical steel roof. Using these parameters, Koumousis and Georgiou were able to locate the area of global extremum satisfactory for design purposes.

Prakkash et al. (1995) used a GA for optimal design of ribbed Ferrocement roofing/flooring elements, with emphasis given to realistic optimal design modeling of this type of structure. In this study, two types of structural elements were considered: double-T and triple-T elements, and examples were presented to illustrate the advantages of using GAs in optimal design of this type of structure.

Rajan (1995) presented a design procedure for simultaneous consideration of sizing, shape, and topology design of space trusses using a GA, using discrete and continuous variables to define the cross-sectional areas of the members. The nodal locations were treated as continuous design variables using the hybrid natural approach for shape optimal design. Element connectivity and boundary conditions were treated as Boolean design variables in the context of topology design. The design procedure reduced the computational expense by using restarts and comparing the chromosome to be evaluated with the database of unique generated chromosomes.

Ohsaki (1995) presented a global search method for topology optimization of truss structures subjected to stress constraints using a GA. Wu and Chow (1995) applied a GA to integrated discrete and configuration optimization of trusses. Their work is similar to the previously mentioned work by the other researchers. Galanate (1996) applied a GA to design two-dimensional truss structures and designed a 10-bar truss and a transmission tower, considering buckling effects.

Soh and Yang (1996) presented a fuzzy controlled genetic-based search technique coupled with expert knowledge and experience for structural shape optimization of two and three-dimensional trusses. The authors provided a survey of papers related to Gas using fuzzy logic techniques. Using an automated GA-based simulation procedure in conjunction with expert knowledge a hybrid fuzzy approach is proposed for the least weight design of structures. This paper is an extension of previous work by the authors in which they provided a series of small transition parameters used as a buffer between the satisfied constraints and the unsatisfied constraints to soften the various constraints in shape optimization problems. Example problems presented show that hybrid fuzzy-GA approach can reduce the required computational time and improve search efficiencies.

Ramasamy and Rajasekaran (1996) designed truss systems using an expert system for multiple loading conditions using a GA by using an artificial neural network to obtain initial areas of the members for a truss system, and a truss was analyzed and designed.

Yang and Soh (1997) developed an optimization procedure of truss structures using GAs with a tournament selection strategy. The authors concluded that their approach using the tour-

nament selection strategy was able to search for an optimum solution in a more efficient manner.

Rajeev and Krishnamoorthy (1997) presented a GA-based methodology for optimal design which simultaneously considers topology, configuration, and cross-sectional parameters in a unified manner. Their methodology is a two-phased approach and can handle both discrete and continuous design variables. The main objective of a two-phased approach is to reduce the size of search space by automatically arriving at lower bound values for design variables. This resulted in improved efficiency of the optimization process. In this procedure, Rajeev and Krishnamoorthy used a GA to arrive at appropriate lower-bound indices for each design variable in Phase 1. In Phase 2, these indices were then improved in an adaptive manner, in order to arrive at the optimal solution. The method used a variable string length GA, which allowed variations in topology, size, and configuration.

Huang and Arora (1997) discussed optimal design of engineering systems having linked discrete design variables. Such problems in structural engineering are encountered when standard sections are used for design of steel structures. Haug and Arora used and compared three strategies: a continuous variable optimization method using a GA, simulated annealing, and branch and bound methods. The authors considered three structural problems, studied the performance of each method, and compared the required computational time of each method. The results indicated that the three strategies performed well for all test problems. In addition, the computational times required to solve the problems were greater compared to those for the continuous variable problems.

Jenkins (1997) provided an overview of GAs and optimized a multistory frame with truss-supported hangers, and suggested that enhancements in GAs could be obtained if crossover and mutation controls were adapted.

Parmee et al. (1997) introduced various strategies which incorporated evolutionary and adaptive search techniques by addressing the problems associated with the decision support aspects of preliminary and global optimal designs.

Leite and Topping (1998) consolidated GAs as an engineering tool for a general-purpose optimization; therefore, they implemented modifications to one-point crossover by defining the effective crossover site and using multiple offspring tournament. Modifications in the GA were designed to increase the exploratory power to allow major reduction in computational time and improve of results. Their basic idea was to improve the efficiency of the GA operators in ways that preserve the balance of exploration and exploitation of solutions. Leite and Topping demonstrated how the improved GA operators may remove the convergence speed while enhancing

the quality of the solution through several examples consisting of a welded beam, ten-bar truss structure, a three-span continuous beam, and prestressed I-sections.

Camp et al. (1998) developed a GA-based design procedure FEAPGEN as a module in the Finite Element Analysis Program (FEAP). Special features of FEAPGEN included: discrete design variables, an open format for prescribing constraints, design checking using the American Institute of Steel Construction Allowable Stress Design (AISC-ASD) specifications, multiple loading conditions, and a comprehensive AISC database of available structural steel members. Several strategies for reproduction and crossover were investigated. In particular, a group selection scheme for reproduction that does not require fitness scaling was applied. Various fitness and penalty functions were investigated for their appropriateness to the ASD design of two-dimensional structures. A comparison of designs of truss and frame structures using the FEAPGEN genetic search design procedure and a classical continuous optimization method based on the optimality criterion were compared.

Chen and Rajan (1998) developed a frame design software system based on a simple GA that incorporated a specialized one-point crossover applied to each design variable individually. The crossover strategy was implemented using an additional string called an association string. Each design variable was matched with a portion of the binary association string. The appropriate segment of the association string was evaluated using a special crossover parameter. If the value of the crossover parameter for a particular design variable exceeded a specified threshold, one-point crossover was applied to that design variable. The major objective was to reduce disruptive crossovers. Results from numerous truss designs showed that, in general, the crossover strategy using association strings preformed efficiently and generated acceptable designs.

Ohmori and Kito (1998) proposed a new methodology for structural optimization of truss topology where the truss topology is expressed as a combination of joined triangles. This concept attempted to avoid topologies that included needless members, undesirable colinear members, or unstable structures. In addition, a parallel GA strategy was proposed based on the concept on environmental effects on evolutionary solutions. In other words, a good environment effects the evolution of desirable solutions. In this study, respective GAs were used to optimize the topology of structures with the associated size and shape of structural members as the environment. Designs using the triangle-based encoding and the proposed parallel GA strategy for 2-D and 3-D trusses were presented.

Cheng and Li (1997 and 1998) developed a constrained multi-objective optimization methodology by combining a Pareto GA with a fuzzy penalty function. The Pareto GA consists of five basic operators: reproduction, crossover, mutation, niche, and Pareto-set filter. The

niche operator was derived from Cavicchio's concept (Cavicchio, 1972) that offspring solutions replace parent solutions only if their rank in the parent's population is equal to or greater than the parents rank. In effect, offspring solutions were produced only around parent solutions or new positions not dominated by old ones. In evolutionary reproduction, the best traits of the parents are always passed on to their offspring, but some of the lost information or traits may be optimal points. The Pareto-set filter attempts to store nondominated points at each generation and eliminate dominated points. Therefore, solutions from a Pareto GA are developed from the entire evolutionary process and not solely from the last generation. The results from several multi-objective optimization problems involving truss structures indicated that the Pareto GA handled complicated response surfaces and achieved the global optimum more often than classical search methods.

Nair et al. (1998) developed an approach that combined approximation models with a GA-based optimization procedure. The objective was to use empirical information to gain asymptotic convergence to the optima using a limited number of exact analyses. The resulting procedure was posed as a dynamic optimization problem with a variable fitness function and a mechanism to select design points where exact analysis should be performed. In addition, an adaptive selection operator was developed to efficiently search the complex design space. Results presented for the design of a 10-bar truss indicated that the number of exact analyses required to determine the optima can be reduced by more than 97% from the original problem.

Crossley et al. (1998) developed a two-branch tournament GA for multi-objective design problems. The authors applied their two-branch GA to the design of a 10-bar truss and an optimization problem that used both discrete and continuous variables. Results from these examples were compared with single-objective approaches to measure the Pareto-optimal set estimated by the two-branch GA.

Topping and Leite (1998) presented a detailed analysis and discussion of strategies and topologies of parallel genetic algorithms (*p*GAs). In particular, the authors presented strategies for *p*GAs using a model based on population, subpopulation, individual, and parallel processing. Parallelism was employed to achieve an increase in computation speed and to allow the solution of much larger optimization problems. The advantages and applicability of each of these approaches was demonstrated in the design of a cable-stayed bridge.

Saka (1998) used a simple GA to minimize the weight in the design of a grillage system constrained by deflection limitations and allowable stresses. In addition, the effects of warping and shear were taken into account. Convergence was obtained when 80% of the current generation was dominated by the best solution.

Hajela et al. (1998) applied a simple GA to determine the optimal layout and size of 2-D and 3-D grillage structures for displacement, stress, and element buckling constraints. A two-level GA was used where the stability constraint was imposed on one level, and the stress and displacement constraints were applied on a second level optimization. Results for a 2-D ten-element grillage structure and for a 3-D helicopter tail-boom structure were compared to solutions obtained using equilibrium linear programming.

Soh and Yang (1998) developed a two-stage GA for the design of bridge trusses that utilized domain knowledge. In the first stage, geometric and sizing variables for the truss are simultaneously optimized for an initial topology pattern. In the second stage, topologic variables are optimized depending on some cognitive topologic patterns and then simultaneous optimization of geometric and sizing variables under the new topology is performed. The cognitive topology patterns represent a set of all the stable topologies when some members and joints are removed for the current topology. Results indicate that designs developed using the GA were lighter than design obtained using traditional numerical optimization techniques.

Groenwold et al. (1999) used a regional genetic algorithm (R-GA) for the discrete design of truss structures. The R-GA used a selection operator with some similarity with genetic rebirth, where the GA was restarted with a random population selected from a new subset or region centered on the best solution in the current generation. Results from several truss designs indicated that the R-GA was computationally efficient in obtaining good approximations to the global optimum. In addition, the procedure was highly suited for the design of large truss structures.

Botello et al. (1999) proposed combining SA and GA by inserting an acceptance operator after the mutation step in a simple GA. In this strategy, the population obtained after the selection step in the GA was compared to the one modified by the crossover and mutation steps before proceeding to the next generation. The effects of selected values of free parameters associated with the SA acceptance operator were studied. Results using the hybrid algorithm were compared with variable length GA (Rajeev and Krishamoorthy, 1997), Monte Carlo SA (Elperin, 1998), SA with automatic reduction of search range (Tran and Pantelides, 1996), iterative SA, and state space optimization (Huang and Arora, 1979).

CONCLUSIONS

REFERENCES

Adeli, H. and Cheng, N.T. (1993). "Integrated Genetic Algorithm for Optimization of Space Structures." *J. Aero. Engrg.* 6(4), 315-328.

Adeli, H. and Cheng, N.T. (1994a). "Augmented Lagrangian Genetic Algorithm for Structural Optimization." *J. Struct. Engrg.* 7(3), 104-118.

Adeli, H. and Cheng, N.T. (1994b). "Concurrent Genetic Algorithms for Optimization of Large Structures." *J. Aero. Engrg.*, 7(3), 276-296.

Booker, L.B. (1987). "Improving Search in Genetic Algorithms." *Genetic Algorithms and Simulated Annealing* (ed Davis, L.), London: Pitman, 61-73.

Booker, L.B. (1993). "Recombination Distributions for Genetic Algorithms." *Proceedings of the 2nd Foundations of Genetic Algorithms Workshop* (ed. Whitley, L.D.), San Mateo, CA: Morgan Kaufmann, 29-44.

Botello, S., Marroquin, J.L., Oñate, E., and Van Horebeek, J. (1999). "Solving Structural Optimization Problems with Genetic Algorithms and Simulated Annealing." *Int. J. Numer. Meth. Engrg.*, Vol. 45, 1069-1084.

Cai, J. and Thierauf, G. (1996). "Evolution Strategies for Solving Discrete Optimization Problems." *Advances in Engineering Software*, 25, 177-183.

Camp, C.V., Pezeshk, S. and Cao, G. (1997). "Design of Framed Structures Using a Genetic Algorithm*."* Chapter in *Advances in Structural Optimization,* Edited by D.M. Frangopol and F.Y. Cheng, ASCE , NY, 19-30.

Camp, C.V., S. Pezeshk, and G. Cao. (1998). "Optimized Design of Two–Dimensional Structures Using a Genetic Algorithm." *ASCE J. of Struct. Engrg.*, 124(5), May.

Cao, G. (1996). *Optimized Design of Framed Structures Using a Genetic Algorithm*. Ph.D. dissertation, The University of Memphis, Memphis, Tennessee.

Cavicchio, D.J. (1972). "Reproductive Adaptive Plans." *Proceedings, ACM 1972 Annual Conference.* ,Association of Computing Machinery, Boston, MA, 1-11.

Chen, D. (1997). *Least Weight Design of 2-D and 3-D Geometrically Nonlinear Structures Using a Genetic Algorithm*. Ph.D. dissertation, The University of Memphis, Memphis, Tennessee.

Chen, S.-Y. and Rajan, S.D. (1998). "Improving the Efficiency of Genetic Algorithms for Frame Designs." *Eng. Opt.*, Vol. 30, 281-307.

Cheng, F.Y. and Li, D. (1997). "Multi-objective Optimization Design with Pareto Genetic Algorithm." *ASCE J. of Struct. Engrg.,* 123(9), 1252-1261.

Cheng, F.Y. and Li, D. (1998). "Genetic Algorithm Development for Multi-objective Optimization of Structures." *AIAA Journal*, Vol 36(5), 1105-1112.

Crossley, W.A, Cook, A.M, Fanjoy, D.W. and Venkayya, V.B. (1998). "Using Two-Brach Tournament Genetic Algorithm for Multiobjective Optimization." *AISS/ASME/ASCE/AHS/ASC Structures, Structural Dyanmics & Materials Conf.*, Vol 2., AIAA, Reston VA, USA, 1752-1762 AIAA-98-1914.

Cvetkovic, D. and Muhlenbein, H. (1994). *The Optimal Population Size for Uniform Crossover and Truncation Selection*, Technical Report GMD-AS-TR-94-11.

De Jong, K.A. and Spears, W.M. (1990). "An Analysis of the Interacting Roles of Population Size and Crossover in Genetic Algorithms." *Proceedings of the International Conference on Parallel Problems Solving from Nature* (eds. Schwefel, H. P. and Manner, R.), Springer-Verlag, 38-47.

De Jong, K.A. and Spears, W.M. (1992). "A Formal Analysis of the Role of Multi-point Crossover in Genetic Algorithms." *Annals of Mathematics and Artificial Intelligence Journal*, 5(1), 1-26.

De Jong, K.A., Spears, W.M., and Gordon, D.F. (1994). "Using Markov Chains to Analyze GAFOs." *Proceedings of the 3rd Foundations of Genetic Algorithms Workshop*, San Mateo: Morgan Kaufmann.

Deb, K. (1990). "Optimal Design of a Class of Welded Structures via Genetic Algorithm." *Proc. 31st AIAA/ASME/ASCE/ASH/ASCS Structures, Structural Dynamics, and Materials Conf.*, 444-453.

Dhingra, A.K., and Lee, B.H. (1994). "A Genetic Algorithm Approach to Single and Multiobjective Structural Optimization with Discrete-Continuous Variables." *Int. J. Numer. Meth. Engrg.,* Vol. 37, 4059-4080.

Eshelman, L., Caruana, R., and Schaffer, D. (1989). "Biases in the Crossover Landscape." *Proc. of 3rd Int. Conf. on Genetic Algorithms* (Ed. Schaffer, J.D.), Morgan Kaufmann, San Mateo, CA, 10-19.

Elperin, T. (1998). "Monte-Carlo Structural Optimizaton in Discrete Varibles with Annealing Algorithm." *Int. J. Numer. Meth. Eng.*, Vol. 26, 815-821.

Forrest, S. (1985). *Documentation for Prisoners Dilemma and Norms Programs That Use the Genetic Algorithm*. University of Michigan, Ann Arbor.

Galante, M. (1996). *"*Genetic Algorithms as an Approach to Optimize Real-World Trusses." *Int. J. Numer. Meth. Engrg.*, Vol. 39, 361-382.

Gallagher, R.H. and Zienkiewicz, O.C. (1973). *Optimum Structural Design: Theory and Applications*, John Wiley & Sons.

Glover, F., Kelly, J.P., and Laguna, M. (1995). "Genetic Algorithms and Tabu Search: Hybrids for Optimization." *Computers Ops Res.*, 22(1), 111-134.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley.

Goldberg, D.E. and Deb, K. (1991). "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms." *Proceedings of the Foundations of Genetic Algorithms Workshop*, Bloomington, Indiana, 69-93.

Goldberg, D.E. and Samtani, M.P. (1986). "Engineering Optimization via Genetic Algorithms." *Proc. of 9th Conf. on Electronic Computation*, ASCE, New York, N. Y., 471-482.

Grierson, D.E. and Pak, W.H. (1993). "Optimal Sizing, Geometrical and Topological Design Using Genetic Algorithms." *Structural Optimization*, Vol. 6, 151-159.

Groenwold, A.A., Stander, N., and Snyman, J.A. (1999). "A Regional Genetic Algorithm for the Discrete Optimal Design of Truss Structures." *Int. J. Numer. Meth. In Eng.*, Vol. 44, 749-766.

Hajela, P. (1992). "Stochastic Search in Structural Optimization: Genetic Algorithms and Simulated Annealing.." Chapter 22, 611-635.

Hajela, P. and Lee, E. (1993). "Genetic Algorithms in Topological Design of Grillage Structures." *Proc., IUTAM Symp. on Discrete Structural Systems, IUTAm,* Zakopane, Poland.

Hajela, P. and Lee, E. (1993). "Genetic Algorithms in Structural Topology Optimization." *Topology Design of Structures*, Bendsoe and Mota Sores, Eds., 117-134, Luwer Academic Publishers, Boston, Mass.

Hajela, P. and Lee, E. (1995). "Genetic Algorithms in Truss Topological Optimization." *Int. J. Solids Structures*, Vol. 32, No. 22, 3341-3357.

Hajela, P. and Yoo, J. (1995). "Constraint Handling in Genetic Search – A Comparative Study." *AIAA Paper No. 95-1143*, New Orleans, LA.

Hajela, P., LEE, E., and Cho, H. (1998). "Genetic Algorithms in Topologic Design of Grillage Structures." *Computer-Aided Civil and Infrastructure Engineering,* Vol. 13, 13-22.

Hillier, F.S. and Lieberman, G.J. (1990). *Introduction to Mathematical Programming*. McGraw-Hill Publishing Company.

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

Homaifar, A., Qi, C.X., and Lai, S.H. (1994). "Constrained Optimization via Genetic Algorithms." *Simulation*, April, 242-251.

Huang, M.-W. and Arora, J.S. (1979). *Applied Optimal Design of Mechanical and Structural Systems,* Wiley, New York.

Huang, M.-W. and Arora, J.S. (1997). "Optimal Design of Steel Structures Using Standard Sections." *Structural Optimization*, 14, 24-35.

Jenkins, W.M. (1991a). "Towards Structural Optimization Via The Genetic Algorithm." *Computers & Structures*, 40(5), 1321–1327.

Jenkins, W.M. (1991b). "Structural Optimisation with The Genetic Algorithm." *The Structural Engineer*, 69(24), 418-422, December.

Jenkins, W.M. (1997). "On the Application of Natural Algorithms to Structural Design Optimisation." *Engineering Structures*, 19(4), 302-308.

Koumousis, V.K. and Georgiou, P.G. (1994). "Genetic Algorithms in Discrete Optimization of Steel Truss Roofs." *J. of Computing in Civil Engineering*, 8(3), 309-325.

Koza, R.R. (1992). *Genetic Programming: on the Programming of Computers by Means of Natural Selection.* The MIT Press, Cambridge, MA.

Leite, J.P. and Topping, B.H.V. (1998). "Improved Genetic Operators for Structural Engineering Optimization." *Advances in Engineering Software*, Vol. 29, No. 7/9, 529-562.

Lin, C.-Y. and Hajela, P. (1992). "Genetic Algorithms in Optimization Problems with Discrete and Integer Design Variables." *Eng. Opt.,* Vol. 19, 309-327.

Lin, C.-Y. and Hajela, P. (1993). "Genetic Search Strategies in Large Scale Optimization." *Proc. 34th AIAA/ASME/ASCE/AHS/ASC SDM Conf.*, La Jolla, Ca, ASCE, New York, NY, 2437-2447.

Lin, C.-Y. and Hajela, P. (1994). "EVOLVE: A Genetic Search Based Optimization Code with Multiple Strategies. *Proceedings of OPTI93 Computer-Aided Optimum Design of Structures,* 7-9 July, Zaragoza, Spain, Eds. S. Hernandez & C.A. Brebbia, Elsevier Science, London, 639-654.

Lu, J., Ding, Y., Wu, B., and Xiao S. (1996). "An Improved Strategy for GAs in Structural Optimization." *Computers & Structures,* 61(6), 1185-1191.

Nair, P.B., Keane, A.J., and Shimpi, R.P. (1998). "Combining Approximation Concepts with Genetic Alogrithm-Based Optimization." *AISS/ASME/ASCE/AHS/ASC Structures, Structural Dyanmics & Materials Conf.*, Vol 2., AIAA, Reston VA, USA, 1741-1751 AIAA-98-1912.

Nix, A.E. and Vose, M.D. (1992). "Modeling Genetic Algorithms with Markov Chains." *Annals of Mathematics and Artificial Intelligence Journal*, 5(1), 79-88.

Ohsaki, M. (1995). "Genetic Algorithms for Topology Optimization of Trusses." *Computers and Structures*, 57(2), 219-225.

Ohmori, H. and Kito, N. (1998). "Structural Optimization of Truss Topology by Genetic Algorithms." *Published by Publication Committee of NCTAM Proceedings,* 331-340, Tokyo, Japan.

Parmee, I.C., Vekeria, H., Bilchev, G. (1997). "The Role of Evolutionary and Adaptive Search During Whole System, Constrained and Detailed Design Optimization." *Eng. Opt.,* Vol. 29, 151-176.

Pezeshk, S., Camp, C.V. and Chen, D. (1997). "Optimal Design of 2-D Frames Using A Genetic Algorithm." *Proceedings of the NSF/ASCE Workshop on Optimal Performance of Civil Infrastructure Systems*, Portland, Oregon, April.

Pezeshk, S., Camp, C.V. and Chen, D. (2000). "Design of Nonlinear Framed Structures Using Genetic Optimization." *ASCE J. Struct. Engrg.*, to appear in March issue.

Prakkash, V.S., Rajeev, S., and Mathews, M.S. (1995). "Optimal Design of Ribbed Ferrocement Roofing/Flooring Elements Using Genetic Algorithms." *J. of Ferrocement*, Vol. 25, No. 1, 1-16, January.

Paz, E.C. (1995). "A Summary of Research on Parallel Genetic Algorithms." *IlliGAL Report No. 95007*, July 1995.

Rajan, S.D. (1995). "Sizing, Shape, and Topology Design Optimization of Trusses Using Genetic Algorithms." *ASCE J. of Struct. Engrg.*, 121(10), 1480-1487.

Rajeev, S. and Krishnamoorthy, C.S. (1992). "Discrete Optimization of Structures Using Genetic Algorithms." *ASCE J. of Struct. Engrg.*, 118(5), 1233-1250.

Rajeev, S. and Krishnamoorthy, C.S. (1997). "Genetic Algorithms–Based Methodologies for Design Optimization of Trusses." *ASCE J. of Struct. Engrg.*, 123(3), 350-358.

Ramasamy, J.V. and Rajasekaran, S. (1996). "Artificial Neural Network and Genetic Algorithm for the Design Optimization of Industrial Roofs - A Comparison." *Computers and Structures*, Vol. 58, No. 4, 747-755.

Reeves, G.R. (1993). *Modern Heuristic Techniques for Combinatorial Problems.* John Wiley & Sons, Inc.

Richardson, J.T., Palmer, M. R., Liepins, G., and Hilliard, M. (1989). "Some Guidelines for Genetic Algorithms with Penalty Functions." *Proc. of 3rd Int. Conf. on Genetic Algorithms* (ed. Schaffer, J.D.), Morgan Kaufmann, San Mateo, CA, 191-197.

Saka, M.P. (1998). "Optimum Design of Grillage Systems Using Genetic Algorithms." *Computer Aided Civil and Infrastructure Engineering*, Vol. 13, 297-302.

Sakamoto, J. and Oda, J. (1993). "Technique for Optimal Layout Design for Truss Structures using Genetic Algorithms." *Collection of Technical Papers – AIAA/ASME Structures,* Structural Dynamics and Material Conference, Publ. by AIAA Washington, DC, USA, Pt. 4, 4402-2408.

Schwefel, H.P. (1981). *Numerical Optimization of Computer Models.* John Wiley & Sons, New York.

Simpson, A.R., Dandy, G.C., and Murphy, L.J. (1992). "Genetic Algorithms Compared to Other Techniques for Pipe Optimization." *J. Water Resour. Planng. and Mgmt.*, ASCE, 120(4), 423-443.

Soh, C.K. and Yang, J. (1996). "Fuzzy Controlled Genetic Algorithm Search for Shape Optimization." *J. of Computing in Civil Engrg.,* Vol 10, No. 2, April, 143-150.

Soh, C.K. and Yang, J. (1998). "Optimal Layout of Bridge Trusses by Genetic Algorithms." *Computer-Aided Civil and Infrastructure Engineering,* Vol. 13, 247-254.

Spears, W.M. (1994). "Adaptive Crossover in Genetic Algorithms." *Artificial Intelligence Center Internal Report #AIC–94–019*, Naval Research Laboratory, Washington, DC 20375.

Spears, W.M. and De Jong, D. (1990). "An Analysis of Multi–Point Crossover." *Proceedings of the Foundations of Genetic Algorithms Workshop*, Bloomington, Indiana. 301-315.

Spears, W.M. and De Jong, K.A. (1991). "On the Virtues of Parameterized Uniform Crossover." *Proc. of 4th Int. Conf. on Genetic Algorithms* (eds. Belew R. and Booker, L.), Morgan Kaufmann, San Mateo, CA, 230-236.

Syswerda, G. (1989). "Uniform Crossover in Genetic Algorithms." *Proc. of 3rd Int. Conf. on Genetic Algorithms* (ed. Schaffer, J. D.), Morgan Kaufmann, San Mateo, CA, 2-8.

Tanese, R. (1989). "Distributed Genetic Algorithms." *Proc. of 3rd Int. Conf. on Genetic Algorithms* (ed. Schaffer, J. D.), Morgan Kaufmann, San Mateo, CA, 434–439.

Thierens, D. and Goldberg, D.E. (1993). "Mixing in Genetic Algorithms." *Proc. of 5th Int. Conf. on Genetic Algorithms*, 38-45.

Topping, B.H.V. and Leite, J.P.B. (1998). "Parallel Genetic Models for Structural Optimization." *Eng. Opt.,* Vol. 31, 65-99.

Tran, S. and Pantelides, C.P. (1996). "Annealing Strategy for Optimal Design." *J. of Struct. Engrg.* 122(7), 815-827.

Whitley, D. (1993). "A Executable Model of a Simple Genetic Algorithm." *Proceedings of the 2nd Foundations of Genetic Algorithms Workshop* (ed. Whitley, L. D.), San Mateo, CA: Morgan Kaufmann, 45-62.

Wu, S.-J. and Chow, P.-T. (1995). "Integrated Discrete and Configuration Optimization of Trusses Using Genetic Algorithms." *Computers and Structures*, Vol. 44, No. 4, 695-702.

Yang, J. and Sho, C.K. (1997). "Structural Optimization by Genetic Algorithms with Tournament Selection." *ASCE J. of Struct. Engrg.*, 11(3), 195-200.

Zhu, D.M. (1986). "An Improved Templeman's Algorithm for Optimum Design of Trusses with Discrete Member Sizes." *Engineering Optimization*, No. 9, 303-312.