AIAA JOURNAL
Vol. 36, No. 6, June 1998

Genetic Algorithm Development for Multiobjective Optimization of Structures

Franklin Y. Cheng* and Dan Li†
University of Missouri–Rolla, Rolla, Missouri 65409

Genetic algorithms (GAs) have the characteristic of maintaining a population of solutions and can search in a parallel manner for many nondominated solutions. These features coincide with the requirement of seeking a Pareto optimal set in a multiobjective (multicriteria, vector) optimization problem. The rationale for multiobjective optimization via GAs is that at each generation, the fitness of each individual is defined according to its nondominated property. Because nondominated individuals are assigned the highest fitness values, the convergence of a population will go to the nondominated zone: the Pareto optimal set. Based on this concept, a Pareto GA, whose goal is to locate the Pareto optimal set of a multiobjective optimization problem, is developed. In this GA, to avoid missing Pareto optimal points during evolutionary processes, a new concept called Pareto-set filter is adopted. At each generation, the points of rank 1 are put into the filter and undergo a nondominated check. In addition, a niche technique is provided to prevent genetic drift in population evolution. This technique sets a replacement rule for reproduction procedures. For a constrained optimization problem, a revised penalty function method is introduced to transfer a constrained problem into a nonconstrained one. The transferred function of a point contains information on a point's status (feasible or infeasible), position in a search region, and distance to the Pareto optimal set. Two multiobjective optimization examples, a 25-bar space truss optimal design (objectives: structural weight and virtual work, constraints: stresses) and a four-bar pyramid truss with control system (objectives: structural weight and control effort, constraints: closed-loop frequencies) are provided to demonstrate analysis procedures of the proposed Pareto GA.

Introduction

THE analytical procedure used to solve a single-objective or multiobjective optimization problem (MOP) is generally based on a conventional linear or nonlinear programming algorithm.[1–4] Although these methods have undergone continual development over the past 30 years, algorithms available to date are not particularly robust; no method performs uniformly well on broad classes of problems. Moreover, most algorithms make use of gradient information to guide their search. As such, they tend to find a local minimum rather than a family of solutions known as the Pareto optimal set from which a multiobjective designer can select the best compromise solution.

The genetic algorithm (GA) introduced by Holland[5] is based on the Darwinian survival of the fittest theory. In contrast to most classical optimization methods, a GA is a parallel and evolutionary search technique. It requires no gradient information and produces multiple optima rather than a single local optimum. These characteristics make the GA a powerful tool for solving multiobjective optimization problems. In view of the potential that GAs have in multiobjective optimization, Schaffer[6] developed a GA that he called the vector evaluated GA (VEGA). VEGA is an extension of a simple GA in which the selection step is modified so that, at each generation, a group of subpopulations is generated by performing equal-size selection with each objective function in turn. These subpopulations are then shuffled together to obtain a new population. In this way, each objective tries to achieve its own best value so that a tradeoff among objective functions is incorporated into the GA's evolutionary process. Although this scheme was simple to implement, the independent selection of champions in each criterion held the potential for bias against middling individuals.[7] Based on the concept of Pareto optimal set, Goldberg[7] suggested a Pareto GA that uses nondominated ranking procedures to guide a population toward the Pareto optimal set in an MOP. He also proposed using a niche technique to keep a population from drifting to some regions in a Pareto optimal space. Two studies[8,9] have tried to solve multiobjective optimization with GAs according to Goldberg's suggestions. The investigators use a sharing niche technique to maintain a uniform convergence of a population along the nondominated set (Pareto optimal set). In this technique, a new GA parameter, the niche radius σ_{share}, is introduced. Performance of the niche technique is sensitive to the setting of σ_{share}.

In this research, a new Pareto GA that solves an MOP with/without constraints is developed on the basis of nondominated ranking. The goal of this Pareto GA is to find a representative sampling of solutions along the Pareto optimal set. To accomplish this, several new techniques are investigated. A new operator called the Pareto-set filter is introduced to prevent the loss of Pareto optimum points in evolutionary progress. A niche technique is created by putting limitations on reproduction operators. Fitness is defined in terms of the rank of a point and the rank size. For constrained optimization problems, the Pareto GA adopts a fuzzy penalty function method. A penalty value is combined with its objective function to create a new means of evaluation that can be easily included in a nondominated ranking procedure. Numerical experiments show that the Pareto GA can be powerful, robust, and computationally feasible for even extremely difficult problems.

Simple GA

In nature, evolution is a natural-selection or self-optimizing process within a specified environment. More successful species survive and propagate while the less successful ones decline. GA, a computer algorithm simulating the evolutionary process, is a search procedure with a randomized yet structured information exchange in a finite space. GA uses random processes to produce an initial population, and simple operators are applied to the population to produce a new population. The new population is referred to as offspring, and the original population as parents. GA represents complex models by simple encoding. It works with a coding of the parameter set, not the parameters themselves. One encoding method is the representation of a model by binary bit strings. Considering an optimal design problem, design variables are $\{x\} = \{x_1, x_2, x_3\}$ and each variable has a specified range so that $x_{i,\min} \le x_i \le x_{i,\max}$. A variable

Received May 24, 1996; revision received Feb. 5, 1997; accepted for publication Jan. 7, 1998. Copyright © 1998 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Curators' Professor, Senior Investigator, Intelligent Systems Center, Department of Civil Engineering.

†Ph.D. Candidate, Department of Civil Engineering; currently Structural Engineer, Ludwig Buildings, Inc., Harahan, LA 70123.

can be encoded by mapping in a range $x_{\min}$–$x_{\max}$ using an n-bit, binary unsigned integer. If a 4-bit code is chosen to map a variable, then $x_{\min} \rightarrow$ (0000) and $x_{\max} \rightarrow$ (1111). There are 2^n values in this range (for this example, $n = 4$). The following equation can be used to decode a string:

$$x_j = x_{j,\min} + \frac{B_j}{2^n - 1}(x_{j,\max} - x_{j,\min}) \tag{1}$$

where B_j is the decimal integer value of binary string for variable x_j. A string usually stands for a point in a design variable space. In the preceding example, let $x_{1,\min} = x_{2,\min} = x_{3,\min} = 0$ and $x_{1,\max} = x_{2,\max} = x_{3,\max} = 4$. A design point $\{x\} = \{1.8667, 2.4, 3.2\}$ could then be represented by the string (011110011100) where the first four bits equal x_1 (1.8667), the middle four bits equal x_2 (2.4), and the last four bits equal x_3 (3.2).

In contrast to a traditional optimization method, which usually starts its search from a point $\{x\}^0 = \{x_1, x_2, x_3\}^0$, GA search works from a population of points $\{x\}^0_1, \{x\}^0_2, \ldots, \{x\}^0_m$. In a traditional method, a search moves gingerly from one point in the decision space to the next, using certain rules to proceed to the next point. In GA, the process works generation by generation (iteration) using probabilistic, not deterministic transition rules, successively generating a new population of strings. A simple genetic algorithm is composed of the following three operators: reproduction, crossover, and mutation.

Reproduction is a selection process to determine the survival potential of a string according to that string's fitness. Fitness is a nonnegative merit measure of optimization objective function with respect to a string. Strings with higher fitness values have a higher probability of proceeding to the next generation. For example, if fitness value of a string is f_j in a population (population size is m), the probability of the ith string chosen as a parent is

$$c_i = f_i \Bigg/ \frac{1}{m} \sum f_j \tag{2}$$

Crossover involves random exchange of corresponding bits between two parent strings to produce two new offspring strings. Crossover operator follows the reproduction procedure. It is a process in which all of the newly reproduced strings are grouped into pairs at random; then each pair of strings undergoes crossover based on certain rules. For example, consider a pair of parents given by previous bit string (011110011100) and another string (011001001011), representing the point (1.6, 1.0667, 2.9333). If one-point crossover is used, the strings after crossover (the crossover position is randomly selected to be 5) are (011111001011) and (011000011100), representing a pair of new points: (1.8667, 3.2, 2.9333) and (1.6, 0.2667, 3.2).

Mutation introduces diversity in a model population by occasional random change in bit values of strings. It plays a secondary role in the operation of GAs. In mutation operator, the value of the bit of a randomly chosen string at a randomly selected position between 1 and L (L is the length of the string) is changed. Mutation is potentially useful in restoring lost diversity in a population.

The goal of optimization is to search the best possible solution in a finite space with respect to one or more objectives. The mechanics in applying a simple GA to an optimization problem are as follows. The simple GA randomly produces a finite initial population (m design points) in a defined space. Then, performing the initial population with simple genetic operators, fitness evaluation, reproduction selection, crossover, and mutation, a new generation population can be generated. This process (evolution) is repeated until a satisfactory solution evolves. Simple GAs have been successfully applied to single-objective optimization problems.[10–12]

Pareto Genetic Algorithms for Multiobjective Optimization

Multiobjective optimization determines a vector of design variables within a feasible region to minimize (or maximize) a vector of objective functions that are usually in conflict with each other. It can be expressed as

$$\begin{aligned} &\text{minimize} && \{f_1(x), f_2(x), \ldots, f_n(x)\} \\ &\text{subject to} && g(x) \le 0 \end{aligned} \tag{3}$$

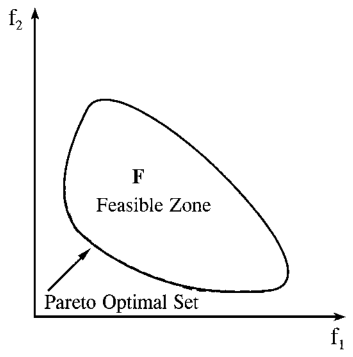

Fig. 1 Feasible region and Pareto optimal set.

where x is the vector of design variables, $f_i(x)$ is the ith objective function, and $g(x)$ is the constraint vector.

The solution for an MOP is always situated in its Pareto optimal set. Such a set can be stated as follows. A feasible vector x^* is a Pareto optimum (nondominated) for Eq. (3) if and only if there exists no feasible vector x such that[13] for all $i \in \{1, 2, \ldots, n\}$

$$f_i(x) \le f_i(x^*) \tag{4}$$

and for at least one $i \in \{1, 2, \ldots, n\}$

$$f_i(x) < f_i(x^*) \tag{5}$$

A Pareto optimization gives a set of nondominated solutions, i.e., solutions for which no criterion (objective) can be improved without worsening at least one other criterion. Figure 1 shows an optimization problem with two objective functions where dash lines indicate the Pareto optimal set in objective space.

A typical characteristic of an MOP is the absence of a unique point that would optimize all criteria simultaneously. Any point in the Pareto optimal set can become an optimum solution that varies with different decisionmakers. There is no generally accepted standard to judge the merits of a solution. Designers make their choices on the basis of tradeoff among design objectives. A Pareto optimal set conveys information on the tradeoff. Thus, a multiobjective formulation optimization can be ideally solved by obtaining an evenly distributed subset of Pareto optimal set and then exploring this subset to find a decisionmaker's preference.

Most traditional multiobjective optimum algorithms seek the best compromise solution. This solution varies according to search method in most MOPs. Furthermore, a solution from a traditional method tends to be trapped by the first local minimum encountered, which cannot be assumed to belong in the Pareto optimal set. Clearly, designers of multiobjective optimization have less choice; they cannot verify a solution as a truly optimal one with a conventional algorithm.

GA procedures are designed to locate global optimum. As noted, a GA is able to maintain a population of solutions and to conduct a parallel search for many nondominated solutions. This ability and search process meets the requirement of seeking a Pareto optimal set to solve an MOP. The basic concept of multiobjective optimization via GAs is that, in each generation, the fitness function of each individual is decided according to its nondominated property. Nondominated individuals always have a higher probability of proceeding to the next generation because they have the highest fitness values. As evolution continues, the convergence of a population goes to its Pareto optimal set zone. Utilizing the parallel search and group output properties of GAs, a Pareto GA for multiobjective optimization is proposed. Unlike a simple GA, the goal of a Pareto GA's search is a region (Pareto optimal zone) rather than an optimal point. Furthermore, this Pareto GA has two operators, niche and Pareto-set filter, besides the three basic operators, reproduction, crossover, and mutation. The analysis procedure for this Pareto GA consists of finding an evenly distributed Pareto optimal set, then choosing a solution from the set. The flow diagram of a Pareto GA is shown in Fig. 2. A detailed explanation is given in subsequent sections.

Nondominated Solutions and Rank

In this Pareto GA, a point's fitness depends on its rank and a point's rank depends on its nondominated nature. An algorithm selecting a

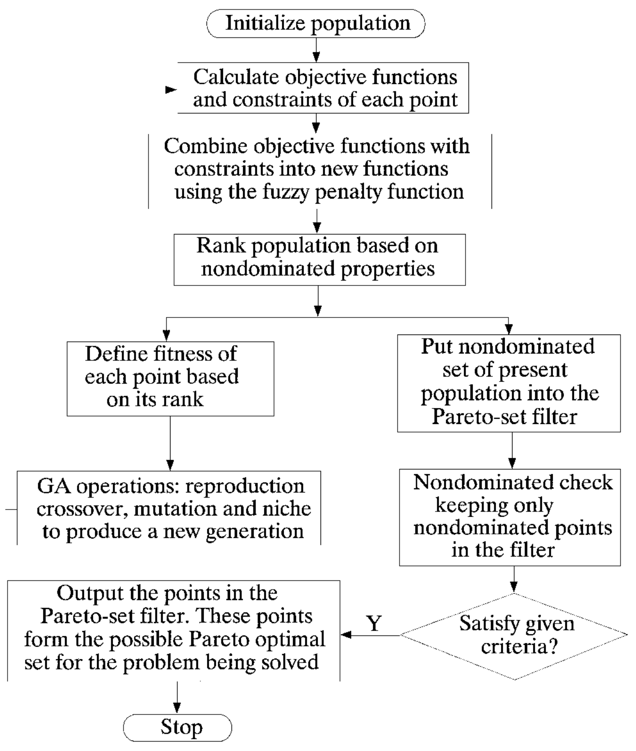
Initialize population
Calculate objective functions and constraints of each point
Combine objective functions with constraints into new functions using the fuzzy penalty function
Rank population based on nondominated properties
Define fitness of each point based on its rank
Put nondominated set of present population into the Pareto-set filter
GA operations: reproduction crossover, mutation and niche to produce a new generation
Nondominated check keeping only nondominated points in the filter
Output the points in the Pareto-set filter. These points form the possible Pareto optimal set for the problem being solved
Y
Satisfy given criteria?
Stop

Fig. 2 Pareto GA for multiobjective optimization with constraints.

nondominated solution from a given solution set $\{F\}$ can be defined as follows.

A vector f^* is a nondominated solution for a multiobjective optimization problem in a population if and only if (maximization problem)

$$f^* \cap f_i < f^*; \qquad f_i \neq f^*, \qquad f_i \in F, \qquad i = 1, 2, \ldots, P_n \tag{6}$$

or (minimization problem)

$$f^* \cap f_i \neq f^*; \qquad f_i \neq f^*, \qquad f_i \in F, \qquad i = 1, 2, \ldots, P_n \tag{7}$$

where P_n is population size.

Nondominated procedure works on the criterion domain. In practical programming, the procedure (for the minimization problem) can be implemented as follows. All items are arranged in ascending order with respect to the first criterion f_1. Where the value of f_1 is the same, relevant items are arranged in ascending order with respect to the second criterion f_2, the third criterion f_3, and so on. After organizing the entire population and eliminating those items dominated by other ones, the remaining items are nondominated points belonging to rank 1 in the present population.

Ranking a population is a continuous labeling process.[7] At each generation, nondominated points are identified and assigned rank 1. From the remaining population, nondominated points are identified and assigned rank 2. This process continues until the entire population is ranked. In reproduction, strings of rank 1 have more copies whereas strings of higher rank have less. Figure 3 illustrates ranking for a two-objective maximization problem.

When the whole population is being ranked, the fitness of each string in a rank can be determined by the following equations:

$$F_i = (N_r - i + 1)/SS \tag{8a}$$

$$SS = \sum_{i=1}^{N_r} (N_r - i + 1) P_{si} \Big/ M \tag{8b}$$

where M is the population size, N_r is the highest rank of the population, P_{si} is the population size of rank i, and F_i is the fitness of a string ranked i. In the fitness definition just given, the reproduction ratio of each rank relies on both its rank level and population size. F_i also defines the reproduction ratio of each point for $\sum F_i = M$.

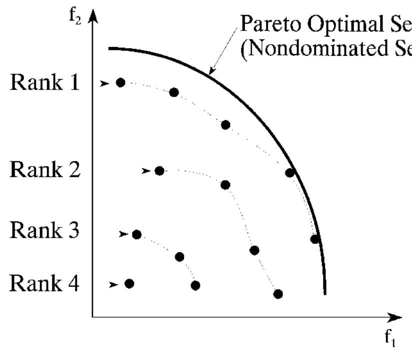

Fig. 3 Population rank (two objectives).

Consider a model population. Its size is $M = 40$ and the maximum rank at generation K is $N_r = 4$. Two cases with different population size P_{si} assumed at each rank are investigated:

rank 1, case I: $P_{s1} = 20$, case II: $P_{s1} = 4$;
rank 2, case I: $P_{s2} = 10$, case II: $P_{s2} = 6$;
rank 3, case I: $P_{s3} = 6$, case II: $P_{s3} = 10$;
rank 4, case I: $P_{s4} = 4$, case II: $P_{s4} = 20$.

Using Eq. (8),

$$SS_1 = (4 \times 20 + 3 \times 10 + 2 \times 6 + 1 \times 4)/40 = 3.15$$

$$SS_2 = (4 \times 4 + 3 \times 6 + 2 \times 10 + 1 \times 20)/40 = 1.85$$

The fitness of a point F_i at each rank is as follows:

rank 1, case I: $F_1 = 1.27$, case II: $F_1 = 2.16$;
rank 2, case I: $F_2 = 0.95$, case II: $F_2 = 1.62$;
rank 3, case I: $F_3 = 0.63$, case II: $F_3 = 1.08$;
rank 4, case I: $F_4 = 0.32$, case II: $F_4 = 0.54$.

The fitness sum of all points is as follows.
Case I:

$$\sum F_i = 1.27 \times 20 + 0.95 \times 10 + 0.63 \times 6 + 0.32 \times 4 = 40 = M$$

Case II:

$$\sum F_i = 2.16 \times 4 + 1.62 \times 6 + 1.08 \times 10 + 0.54 \times 20 = 40 = M$$

Analysis of the preceding example shows that the fitness of a lower rank point is decided by its rank level and rank population size. The larger the population size is, the smaller the fitness of a point. For example, at rank 1, $F_i = 1.27$ for case I (population size is 20 at this rank level) and $F_i = 2.16$ for case II (population size is 4 at this rank level). This fitness definition [Eq. (8)] satisfies the requirement of stable and robust convergence in the GA's evolutionary process.

Objective Functions, Constraints, and Fitness Functions

In a GA for single-objective optimization, fitness function connects with objective function in a linear or nonlinear formulation. In a Pareto GA dealing with a vector function space, fitness expresses the relative position of a point in the rank space rather than the values of objective functions. The nondominated property of a point determines its fitness value. Nondominated procedures for points operate in the objective function domain. For a constrained optimization problem, constraint conditions of a point must be included in its fitness function. Using a penalty function method with the stipulation that a feasible point is always superior to an infeasible one, a process of redefining objective functions can be constructed as follows. First, the n objective values (minimum $\{f_1, \ldots, f_m\}_j$, maximum $\{f_{m+1}, \ldots, f_n\}_j$) of point j are normalized; then, integrating with the point's penalty function, they are transformed into a minimization problem (minimum $\{f_1'', \ldots, f_n''\}_j$) according to

$$f_i'' = \begin{cases} f_i' & (f_i' \in F) \\ f_i' + R_j & (f_i' \in F) \end{cases} \quad i = 1, \ldots, n \tag{9a}$$

$$f_i' = \begin{cases} f_i/(1 + f_i) & (\text{minimum} f_i) \\ 1/(1 + f_i) & (\text{maximum} f_i) \end{cases} \tag{9b}$$

where F is the feasible zone in an objective function space, R_j is the penalty value of point j, and $R_j > 1$ (see next section). In Eq. (9b), it is assumed that $f_i \geq 0$. Equation (9) converts a general optimization problem into a minimization problem and normalizes the objective functions so that $0 \leq f_i < 1$.

Now the ranking procedure works on the new function value set $(\{f_1'', \ldots, f_n''\}_j, j = 1, \ldots, P_n$, where P_n is the population size) of the population. From Eq. (9), it can be observed that $0 \leq f_i'' \leq 1$ when the point is in the feasible zone; otherwise $f_i'' > 1$. Enforcing this principle, it is easy to separate feasible points from infeasible points. Note that infeasible points always hold higher rank than feasible points. This is because any feasible point ($f_i'' \leq 1$) is a nondominated one with respect to infeasible points ($f_i'' > 1$).

Constrained Multiobjective Optimization via GAs

A revised penalty function technique that transforms a constrained problem into a nonconstrained one is presented in this section. As distinct from traditional optimization methods, a GA knows nothing about the type of problem it handles and requires no gradient information or other internal knowledge about a problem. All it needs is the fitness evaluation of each point in a population. Fitness value of a point relates to its rank and transferred functions. A properly transferred function carries information on the point's status (feasible or infeasible), distance to Pareto optimal set, and position in an infeasible zone. Three rules for proper transfer in a Pareto GA are as follows: 1) The function easily marks the point's status as feasible or infeasible; 2) the closer a point is to the feasible zone, the higher its fitness evaluation; and 3) the closer a point is to the Pareto optimal set, the higher its fitness evaluation. Among them, rule 2 should dominate rule 3 because an infeasible point is an unacceptable solution. Pertaining to the three rules and the ranking process, a revised penalty function of point K, based on fuzzy logic theory,[14] is created as follows for zones 1–11, respectively:

$$R_K = \begin{cases} 0 & \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.001 \\ 2 & 0.001 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.01 \\ 3 & 0.01 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.02 \\ 4 & 0.02 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.05 \\ 5 & 0.05 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.1 \\ 6 & 0.1 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 0.4 \\ 7 & 0.4 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 1.0 \\ 8 & 1.0 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 2.0 \\ 9 & 2.0 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 5.0 \\ 10 & 5.0 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \leq 15.0 \\ 100 & 15.0 < \text{maximum}(d_{K1}, \ldots, d_{KM}) \end{cases} \tag{10}$$

where d_{Ki} is the violation amount of point K to the ith constraint [Eq. (11)] and M is the number of constraint conditions,

$$d_{Ki} = \begin{cases} 0 & g_i(x_k) \leq 0 \\ g_i(x_k) & \text{otherwise} \end{cases} \tag{11}$$

In Eq. (10), the constraints are normalized [for example, $g_i(x_k) = \sigma(x_k) - 25 \leq 0$ becomes $g_i(x_k) = \sigma(x_k)/25 - 1 \leq 0$] to give them the same general order of magnitude.

Unlike a traditional penalty function method, the present algorithm provides constraints with a violated quantity rather than a value. This fuzzy formulation matches GA operators that are random and probabilistic rather than precise numerical analysis and calculation. This method also reduces rank levels and satisfies the requirements of a nondominated ranking process in the Pareto GA.

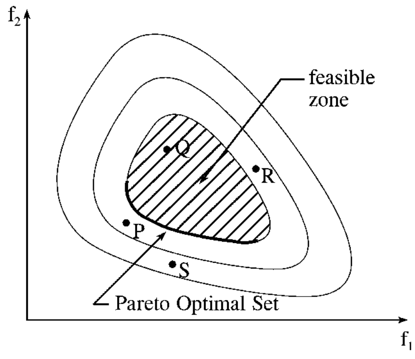

Fig. 4 Relationship of function values and positions.

In terms of penalty concept, a point's transferred function value in an infeasible zone is a comprehensive representation of its objective function plus its penalty function. For point K, transferred function values are $\{f_1'', \ldots, f_n''\}_K$, where

$$f_i'' = f_i' + R_K \tag{12}$$

From this equation it can be seen that the penalty function method obeys the three rules. The value of $f_i''(f_i'' \leq 1$ or $f_i'' > 1)$ indicates a point's status. R_K (integer part of f_i'') represents a point's violated degree in the infeasible zone, and f_i' [Eq. (9b), decimal part of f_i''] defines a point's distance to the Pareto optimal set.

Figure 4 serves as an illustration of four points to Eq. (12). Point Q(0.5, 0.7) is in the feasible zone (zone 1). Infeasible points P(0.3, 0.3) and R(0.6, 0.7) are in the same zone (zone 2), and infeasible point S(0.15, 0.52) is in a higher zone (zone 3). According to rule 1, point Q is the best one no matter where it is in the feasible zone. Though points P and R have the same penalty function values ($R_P = R_R = 2$), point P is better than point R because its function values $\{f_1', \ldots, f_n'\}_P$ are closer to the Pareto region. S is the worst of the three infeasible points, being farthest from the feasible zone. Applying Eq. (12), the new values of the four points are Q(0.5, 0.7), P(2.3, 2.3), R(2.6, 2.7), and S(3.15, 3.52). In this minimization example, Q dominates point P, P dominates R, and R dominates S. Clearly, this fuzzy logic penalty function method can correctly express rank order of a given point in a Pareto GA.

Using the revised penalty function method, the original multiobjective constrained optimization problem Eq. (3) becomes

$$\begin{gathered} \text{minimum}\{f_1'', \ldots, f_m''\} \\ x_i^{(1)} \leq x_i \leq x_i^{(u)} \qquad i = 1, \ldots, L \end{gathered} \tag{13}$$

Pareto Optimal Set and Pareto-Set Filter

This Pareto GA tries to locate a Pareto optimal set in an objective space. Each point in the set should be equally important and treated as an optimum goal. However, reproducing a new generation cannot guarantee that the best traits of the parents will be inherited by their offspring. It is possible that some of these traits may never appear in future phases of evolution due to limited population size. In the evolutionary process, many points appear once or twice and then disappear forever. Some of them may be the sought for optimum goals: the Pareto optimal points. To avoid missing Pareto optimal points, a new concept called Pareto-set filter is introduced. A Pareto-set filter pools nondominated points ranked 1 at each generation and drops dominated points.

At each generation, the points designated rank 1 are put into a filter. When new points are added to the Pareto-set filter, all points in the filter are subjected to a nondominated check (filtering process) and dominated points are discharged. Thus, only nondominated points are stored in the filter. Filter size can be set as equal to population size or any reasonable value. When the number of points in a filter surpasses a given size, the points with minimum distance relative to other points are removed to maintain an even distribution of points.

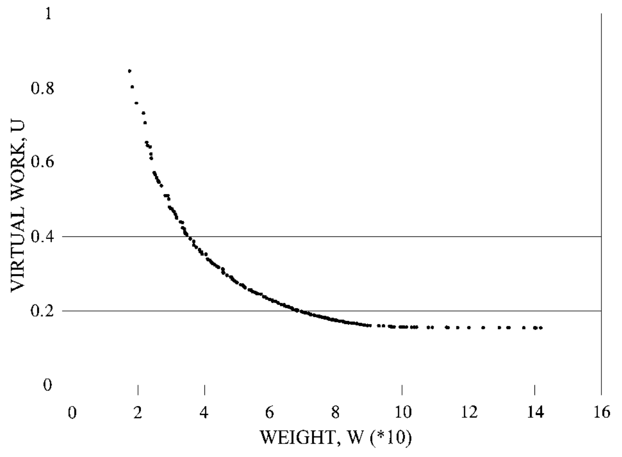

Fig. 5 Points of rank 1 (generation 500).

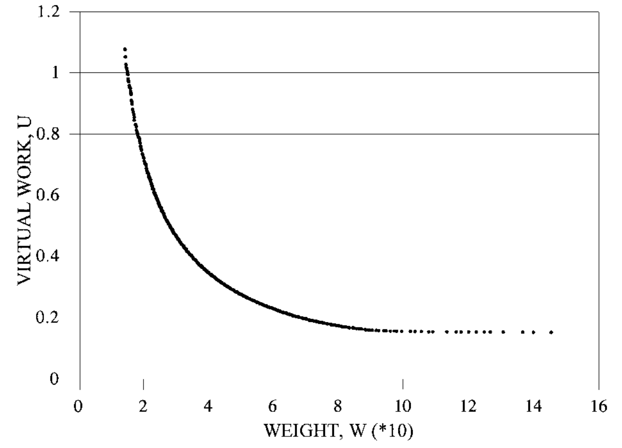

Fig. 6 Points in Pareto-set filter (generation 500).

Whereas the solution of a GA without a Pareto-set filter comes only from the last generation p_n [Eq. (14)], the solution of this Pareto GA is picked from the entire evolution strategy P [Eq. (15)]:

$$p_n = \{f_{i,n};\ i = 1, \ldots, L | f_{i,n} \in F\} \tag{14}$$

$$P = \{f_{i,j};\ i = 1, \ldots, m;\ j = 1, \ldots, n | f_{i,j} \in F\} \tag{15}$$

where L is the number of points assigned rank 1 in generation n, m is the filter size, n is the generation number, $f_{i,j}$ is a nondominated solution in generation j, and F is the feasible zone of an optimization problem.

Collection P explicitly includes collection p_n and dominates the latter. Figures 5 and 6 show results of points in a filter and points from rank 1 at generation 500. These results come from a two-objective optimization problem in a 25-bar space truss (example 1). Note that the filter brings much improvement in the optimum result set.

This new technique avoids missing any Pareto set points; all Pareto set points appearing in the evolutionary process are recorded. The final filter gives a set of points closer to or actually in the Pareto set zone. Furthermore, the effects of genetic drift are reduced, and the convergence of the approach is robust.

Niche Technique for the Pareto GA

Biological evolution maintains a variety of species. Because of the finite size of an artificial population and the stochastic errors associated with GA operators, a phenomenon known as genetic drift occurs in GA evolutionary processes. Genetic drift makes a converging population become nearly identical and cluster at certain optimum regions.[15] Niche techniques force individuals to share available resources and maintain appropriate diversity. As noted, the goal of a Pareto GA is to locate the Pareto optimal set of an MOP. Thus, an effective niche technique is the key to the success of GAs for MOPs.

A niche technique presented herein is derived from the concept that an offspring replaces its parent if the offspring's fitness exceeds that of the inferior parent. After a pair of new offspring is produced, the parents are replaced only when the rank of one offspring in the parent population is no worse than the best rank of their parents. Otherwise, the parents go to the next generation. Applying this limitation, new offspring always hold gene characteristics superior or equivalent to their parents. Children are only produced around their parent positions or at new position not dominated by the old ones. Note two other characteristics of this niche method. It prevents the formation of a lethal, and its reproduction procedure is a steady-state one. Equation (16) outlines the niche technique,

$$\text{parent1} + \text{parent2} \rightarrow \text{child1 and child2} \tag{16a}$$

$$\text{parent\_rank} = \min(\text{parent1\_rank, parent2\_rank}) \tag{16b}$$

$$\text{child\_rank} = \min(\text{child1\_rank, child2\_rank}) \tag{16c}$$

$$\text{test} = \text{child\_rank} \leq \text{parent\_rank} \tag{16d}$$

$$\text{new\_child1} = \text{ if (test) child1 else parent1} \tag{16e}$$

$$\text{new\_child2} = \text{ if (test) child2 else parent2} \tag{16f}$$

where two children's ranks (child1_rank and child 2_rank) are determined in the domain of their parent population in which they are treated as members of their paternal population. Alternatively, if the min in Eq. (16b) is changed to max, the control condition is relaxed. However, numerical experiments show the original choice (min) is better than the alternative one (max).

The proposed method has been tested in numerical experiments of MOPs. Comparison of results (with/without niche technique) in a 25-bar space truss (Example 1) is shown in Figs. 7 and 8.

The niche technique helps prevent genetic drift and maintains a uniformly distributed population along the Pareto optimal set.

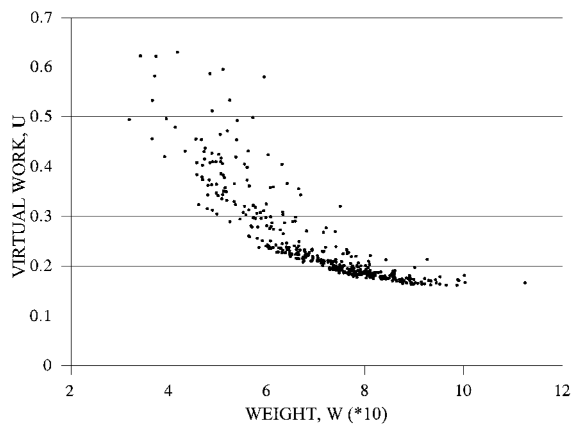

Fig. 7 Generation 800 without niche.

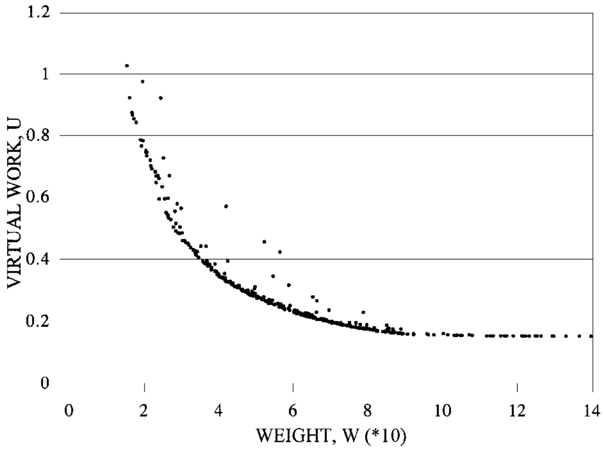

Fig. 8 Generation 800 with niche.

Additional computational time is negligible with regard to the total time spent in optimization via GAs.

Applications

Pareto GA techniques just outlined were applied to two examples. First is a 25-bar space truss. Some points of its Pareto optimal set can be determined by a traditional optimization method. Second is an integrated optimum design of structural and control systems for a four-bar pyramid truss. Here, a traditional optimization algorithm is usually trapped in a local solution. For both examples, GA parameters are chosen as follows. Uniform crossover[16] and stochastic remainder selection[7] are used for crossover and selection operating procedures. Crossover probability is 0.60, and mutation probability is 0.01.

Example 1: 25-Bar Space Truss

A 25-bar space truss,[17] shown in Fig. 9, is doubly symmetric. This condition divides all truss members into eight groups. They are A_1, $A_2 = A_3 = A_4 = A_5$, $A_6 = A_7 = A_8 = A_9$, $A_{10} = A_{11}$, $A_{12} = A_{13}$, $A_{14} = A_{15} = A_{16} = A_{17}$, $A_{18} = A_{19} = A_{20} = A_{21}$, and $A_{22} = A_{23} = A_{24} = A_{25}$. Acting loads, $P_1 = 20$, $P_2 = -5$, $P_3 = 20$ and $P_4 = -5$, are also shown in Fig. 9. The weight density and the elastic module of the structure are $\rho = 0.01$ and $E = 10,000$, respectively.

Optimum objective is to minimize structural weight W and the sum of the virtual work of all members U, i.e.,

$$W = \sum A_i L_i \rho_i, \qquad U = \sum \frac{\sigma_i \bar{\sigma}_i v_i}{E} \tag{17}$$

where A_i, L_i, v_i, and ρ_i are cross-sectional area, length, volume, and weight density of the ith bar and σ_i and $\bar{\sigma}_i$ are virtual work stresses for the real and dummy loadings, respectively. Dummy loading $P = 1$ acts in the same position and direction as loading P_1. Constraint conditions are

$$\begin{aligned} |\sigma_i| &\le 25\,\text{ksi}, & i &= 1, \ldots, 25 \\ 0.1\ \text{in.}^2 &\le A_i \le 5.0\,\text{in.}^2, & i &= 1, \ldots, 25 \end{aligned} \tag{18}$$

In the GA optimization procedure, population size is 400 and chromosome (string) length is 120. Optimal results are displayed in Figs. 5–8 and 10. In Fig. 10, optimum solutions (Min. W, Min. U, and Min. W, U) are on the Pareto-set filter. Two optimum solution (Min. W, $W = 11.570$; Min. U, $U = 0.152$) are found by using a traditional optimal method[18]; the multiobjective optimum solutions (Min. W and U, $W = 40.611$, $U = 0.339$) is solved by using the game theory algorithm.[19] Results indicate that the points in the Pareto-set filter in generation 800 form a subset of the Pareto optimal set.

Optimum results of a Pareto GA allow designers to select the best compromise solution from the solution set and to determine whether the solution is robust. In the selection process, a decisionmaker should not pick a solution from segments whose slopes are close or equal to 0 or $\pi/2$. Note the right end segment of the Pareto optimal set in Fig. 10. When virtual work decreases from 0.155 to 0.152 (-2%), structural weight increases from 100.0 to 150.0 ($+50\%$). Clearly, this segment is not a good tradeoff region. In this example, a rational compromise design is situated at the middle segment of the set.

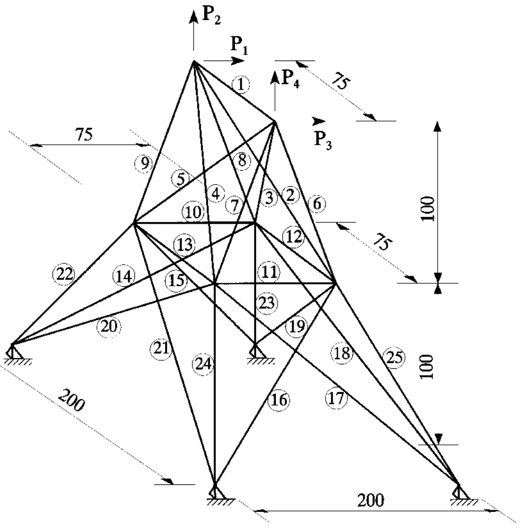

Fig. 9 Example 1: 25-bar space truss.

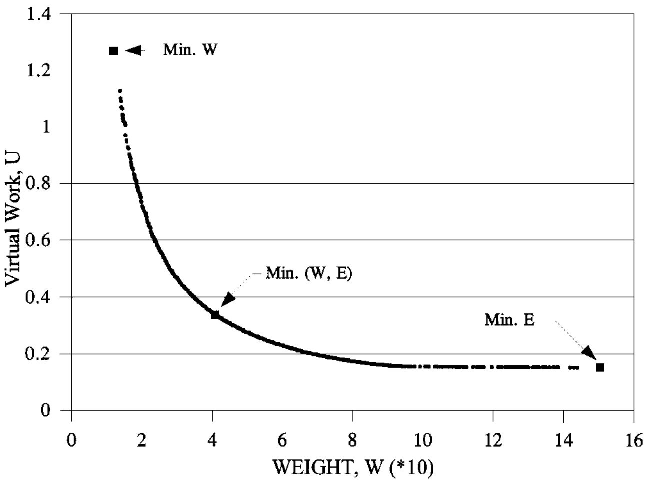

Fig. 10 Pareto-set filter at generation 800.

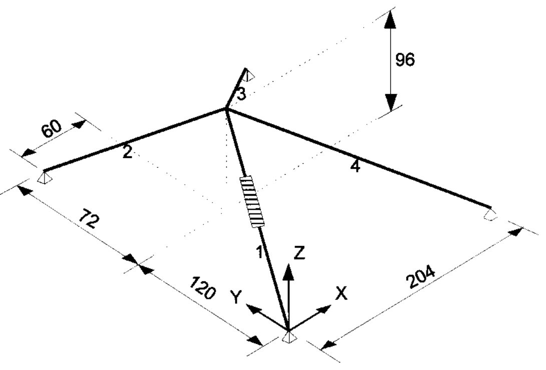

Fig. 11 Four-bar pyramid truss.

Example 2: Four-Bar Pyramid Truss

A four-bar pyramid truss[20] is shown in Fig. 11. The weight density and the elastic module are $\rho = 0.0001$ and $E = 1$, respectively. An active control system is located on bar 1. Passive damping ratio ξ of the structure is assumed to be zero. A nonstructural mass of two units is attached at the top node.

In this optimization example, minimizing structural weight W and control effort[21,22] V is the optimum goal,

$$W = \sum A_i L_i \rho_i, \qquad V = \text{tr}[P] \tag{19}$$

The integrated structural and control design procedure for this system appears in the Appendix.

Optimum constraint conditions are put on frequencies ω_i of the closed-loop system $[A_{\text{cl}}]$ [Eq. A2] and cross-sectional areas of the bars:

$$\begin{aligned} &0.7 \le \omega_1 \le 0.8, \qquad 0.9 \le \omega_2 \le 1.0, \qquad 1.1 \le \omega_3 \\ &10.0 \le A_i \le 1000, \qquad i = 1, \ldots, 4 \end{aligned} \tag{20}$$

In this example, population size is 400 and chromosome length is 60. Numerical results are shown in Figs. 12–15.

In Fig. 12, distribution of the initial population is distant from the possible optimal zone, and only four points among the 400 individuals are in the feasible zone. Comparing Fig. 13 with Fig. 14 shows that the points in the Pareto-set filter (Fig. 13) constitute a much

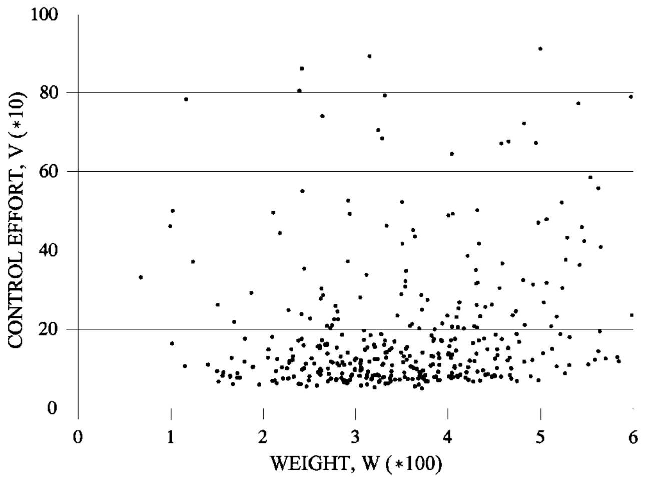

Fig. 12 Population distribution at generation 0.

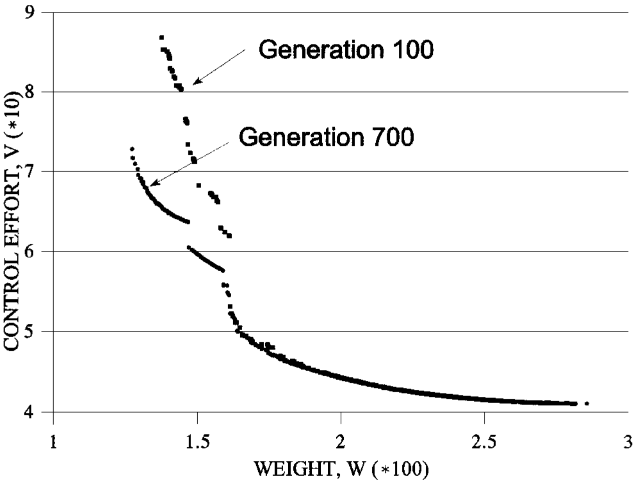

Fig. 15 Filters at generation 100 and 700.

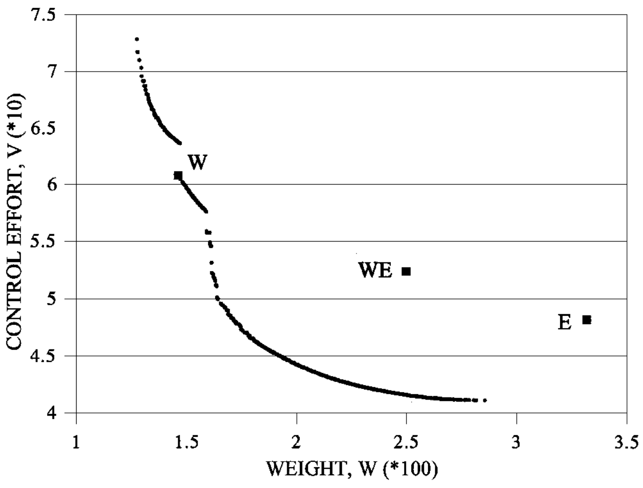

Fig. 13 Pareto-set filter at generation 700.

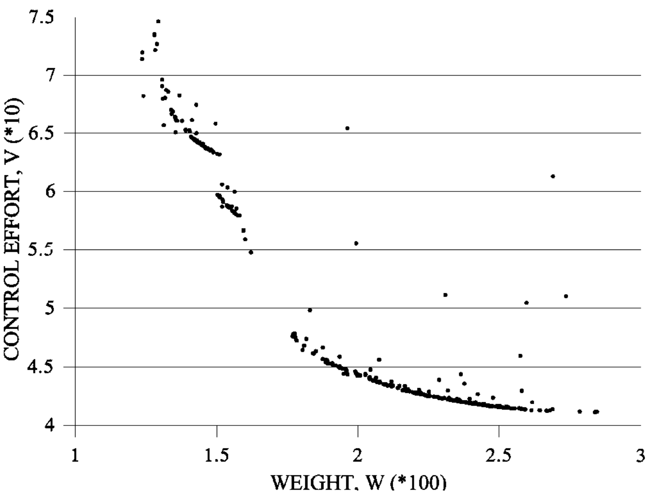

Fig. 14 Population distribution at generation 700.

better nondominated set than the point of rank 1 (Fig. 14) in generation 700. GA results in the filter give the relationship of weight vs input effort tradeoffs. Points W, V, and WV in Fig. 13 are optimum solutions of traditional optimization algorithms,[18,19] which minimize weight (W, $W = 145.83$), input effort (V, $V = 48.218$), and weight and input effort (W and V, $W = 249.83$, $V = 52.388$) subject to the given constraints. Obviously, the single-objective optimization results (W, V) are local ones and the multiobjective optimization solution (WV) is not in the Pareto optimal set. Point WV is not an optimum solution for this MOP because weight W and control effort V can be simultaneously improved without damage to either one. Figure 15 displays the point change in the filter from generation 100 to generation 700. Numerical results show that this Pareto GA is efficient and robust and displays global convergence.

Discussion and Conclusions

Details of the Pareto optimal solution and Pareto GA theory are presented, and the relationship between them is described. A Pareto genetic algorithm is developed for MOPs with/without constraints. Numerical experimental results, obtained from two engineering problems, show the ability of the Pareto GA to seek tradeoff surface regions and find global optimum.

Even more encouraging is the apparent robustness of the method. Note in example 2 that it is difficult to find feasible solutions (in the initial population, there are only four points in the feasible region). Yet results still show an even distribution subset of nondominated points in the feasible region after 700-generation evolution. Development of the Pareto GA can truly bring a rational decisionmaking process into engineering design and multiobjective optimization. Designers do not need to work in a black-box environment. They can make decisions based on objective tradeoffs determined by solutions of a Pareto GA. Results also allow for consideration of factors such as sensitivity and tolerance.

In this Pareto GA, a formulation to separate feasible and infeasible points and embed them with objective functions and to configure the GA rank-based evolution has been presented. The definition of fitness considers rank level of a point and population size of the rank.

A revised penalty function method is constructed on theories of fuzzy logic and Pareto optimal solutions. A penalty value is an expression of the nondominated character and infeasible position of a point. This revised method enables the Pareto GA to solve MOPs with constraints.

A newly developed niche technique operates in the objective function domain to take advantage of properties of the Pareto optimal set. It is easy to use and involves no new parameters. Computational cost of this technique is negligible. Its application to the evolutionary process maintains a population of diverse individuals.

The Pareto-set filter presented makes a Pareto GA more robust and less affected by genetic drift. Furthermore, results from the points in a filter are always superior to those from the points of rank 1.

This Pareto GA can handle complicated multiobjective response surfaces and achieve the global optimum more often than classical search methods. Although it takes longer to locate optimal regions, this Pareto GA seems able to do so more reliably. Its ability to give a designer insight into the problem and find the global optimum should encourage increased application in engineering and related fields.

Appendix: Integrated Structure and Control Design

Equations of motion for a structure with active controls are given by[22]

$$[M]\{\ddot{x}\} + [C]\{\dot{x}\} + [K]\{x\} = [D]\{f\} \qquad \text{(A1)}$$

For the four-bar pyramid truss,

$$[M] = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}, \qquad [C] = [0]$$

$$[K] = \begin{bmatrix} \sum_{i=1}^{4} c_{xi}^2 \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{xi} c_{yi} \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{xi} c_{zi} \frac{E_i A_i}{L_i} \\ \sum_{i=1}^{4} c_{yi} c_{xi} \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{yi}^2 \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{yi} c_{zi} \frac{E_i A_i}{L_i} \\ \sum_{i=1}^{4} c_{zi} c_{xi} \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{zi} c_{yi} \frac{E_i A_i}{L_i} & \sum_{i=1}^{4} c_{zi}^2 \frac{E_i A_i}{L_i} \end{bmatrix}$$

$$[D]^T = [0.3637, 0.7274, 0.5819]$$

where c_{xi}, c_{yi}, and c_{zi} are cosines of angles between the ith member and global axis X, Y, and Z, respectively (Fig. 11). E_i, A_i, and L_i are the elastic module, cross-sectional area and length of the ith member. For the given structure,

$$c_{x1} = 0.3637, \qquad c_{y1} = 0.7274$$
$$c_{z1} = 0.5819, \qquad L_1 = 164.973$$
$$c_{x2} = 0.4472, \qquad c_{y2} = -0.5367$$
$$c_{z2} = 0.7155, \qquad L_2 = 134.164$$
$$c_{x3} = -0.7682, \qquad c_{y3} = -0.3841$$
$$c_{z3} = 0.5121, \qquad L_3 = 187.446$$
$$c_{x4} = -0.6838, \qquad c_{y4} = 0.5698$$
$$c_{z4} = 0.4558, \qquad L_4 = 210.599$$

$[M]$, $[C]$, and $[K]$ are mass, damping, and stiffness matrices, respectively. $[D]$ is an applied load distribution matrix that relates to the control input force vector $\{f\}$.

A closed-loop system is given by

$$[A_{\mathrm{cl}}]\{u\} = \{\dot{u}\} \tag{A2}$$

where

$$[A_{\mathrm{cl}}] = [A] - [B][G] \tag{A3}$$

$$[G] = [R]^{-1}[B]^T[P] \tag{A4}$$

$[A]$ is a plant matrix, and $[B]$ is an input matrix:

$$[A] = \begin{bmatrix} [0] & [I] \\ [-\omega^2] & [-2\xi\omega] \end{bmatrix} \tag{A5}$$

$$[B] = \begin{bmatrix} [0] \\ [\Phi]^T[D] \end{bmatrix} \tag{A6}$$

where ξ (=0) is the damping factor, ω is the natural frequency of the structure, $[\Phi]$ is the modal matrix whose columns are eigenvectors, and $[\Phi]^T[M][\Phi] = [I]$, $[\Phi]^T[K][\Phi] = [\omega^2]$, and $[\Phi]^T[C][\Phi] = [2\xi\omega]$.

$[P]$ represents a symmetrical positive definite matrix called the Riccati matrix and is found by solving the following algebraic Riccati equation:

$$[A]^T[P] + [P][A] + [Q] - [P][B][R]^{-1}[B]^T[P] = [0] \tag{A7}$$

where $[Q]$ and $[R]$ are state weighting matrix and control weighting matrix, respectively. In this example, $[Q]$ and $[R]$ are taken as unit matrices.

Acknowledgments

This presentation is a portion of research results from projects partially supported by the National Science Foundation under Grants MSS-9214664 and CMS-9520804 as well as the Intelligent Systems Center at the University of Missouri–Rolla. The financial support is gratefully acknowledged.

References

[1]Cheng, F. Y., and Botkin, M. E., "Nonlinear Optimization Design of Dynamic Damped Frames," *J. Struct. Div.*, Vol. 102, 1976, pp. 609–628.

[2]Cheng, F. Y., and Juang, D. S., "Assessment of Various Code Provisions Based on Optimum Design of Steel Structures," *International Journal of Earthquake Engineering and Structural Dynamics*, Vol. 16, 1988, pp. 45–61.

[3]Rao, S. S., Venkayya, V. B., and Khot, N. S., "Game Theory Approach for Integrated Design of Structures and Controls," *AIAA Journal*, Vol. 26, No. 4, 1988, pp. 464–469.

[4]Cheng, F. Y. (ed.), *Recent Developments in Structural Optimization*, American Society of Civil Engineers, New York, 1986.

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

[6]Schaffer, J. D., "Multiple Objective Optimization with Vector Evaluated Genetic Algorithms," *Proceedings, 1st International Conference on Genetic Algorithms*, Lawrence Erlbaum, Pittsburgh, PA, 1985, pp. 93–100.

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

[8]Fonseca, C. M., and Fleming, P. J., "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization," *Proceedings, 5th International Conference on Genetic Algorithms*, Morgan Kauffman, 1993, San Mateo, CA, pp. 416–423.

[9]Horn, J., Nafpliotis, N., and Goldberg, D. E., "A Niched Pareto Genetic Algorithm for Multiobjective Optimization," *Proceedings, 1st IEEE Conference on Evolutionary Computation*, Vol. 1, Inst. of Electrical and Electronics Engineers, New York, 1994, pp. 82–87.

[10]Adeli, H., and Cheng, N. T., "Integrated Genetic Algorithm for Optimization of Space Structures," *Journal of Aerospace Engineering*, Vol. 6, No. 4, 1993, pp. 315–328.

[11]Powell, D., and Skolnick, M. M., "Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints," *Proceedings, 5th International Conference on Genetic Algorithms*, Morgan Kauffman, San Mateo, CA, 1993, pp. 424–431.

[12]Le Riche, R., and Haftka, R. T., "Optimization of Laminate Stacking Sequence for Buckling Load Maximization by Genetic Algorithm," *AIAA Journal*, Vol. 31, No. 5, 1993, pp. 951–956.

[13]Eschenauer, H., Koski, J., and Osyczka, A. (eds.), *Multicriteria Design Optimization: Procedures and Applications*, Springer–Verlag, Berlin, 1990.

[14]Zimmerman, H. J., *Fuzzy Set Theory and Its Applications*, 2nd ed., Kluwer Academic, Norwell, MA, 1990.

[15]Cavicchio, D. J., "Reproductive Adaptive Plans," *Proceedings, ACM 1972 Annual Conference*, Association for Computing Machinery, Boston, MA, 1972, pp. 1–11.

[16]Davis, L. (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[17]Venkayya, V. B., Khot, N. S., and Reddy, V. S., "Energy Distribution in an Optimal Structural Design," U.S. Air Force Flight Dynamics Lab., AFFDLTR-68-165, Wright–Patterson AFB, OH, 2, 1969.

[18]Vanderplaats, G. N., "DOT: Design Optimization Tools User Manual," 1994.

[19]Cheng, F. Y., and Li, D., "Multiobjective Optimization of Structures with/without Control," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 2, 1996, pp. 392–397.

[20]Morris, A. J., "The Optimization of Statically Indeterminate Structures by Means of Approximate Geometric Programming," 2nd Symposium on Structural Optimization, AGARD Conf., preprint-123, Milan, Italy, 1973.

[21]Gaudreault, M. L. D., Liebst, B. S., and Bagley, R. L., "Simultaneous Design of Active Vibration Control and Passive Viscous Damping," *Journal of Guidance, Control, and Dynamics*, Vol. 16, No. 4, 1993.

[22]Cheng, F. Y., "Application and Assessment of Structural Optimization and Active Control for Seismic Structures," *Recent Advances in Multidisciplinary Analysis and Optimization*, Proceedings, 3rd USAF/NASA Symposium, 1990, pp. 171–177.

A. D. Belegundu
Associate Editor