AUGMENTED LAGRANGIAN GENETIC ALGORITHM FOR STRUCTURAL OPTIMIZATION

By Hojjat Adeli,[1] Member, ASCE, and Nai-Tsang Cheng[2]

ABSTRACT: This paper presents a robust hybrid genetic algorithm for optimization of space structures using the augmented Lagrangian method. An attractive characteristic of genetic algorithm is that there is no line search and the problem of computation of derivatives of the objective function and constraints is avoided. This feature of genetic algorithms is maintained in the hybrid genetic algorithm presented in this paper. Compared with the penalty function-based genetic algorithm, only a few additional simple function evaluations are needed in the new algorithm. Furthermore, the trial and error approach for the starting penalty function coefficient and the process of arbitrary adjustments are avoided. There is no need to perform extensive numerical experiments to find a suitable value for the penalty function coefficient for each type or class of optimization problem. The algorithm is general and can be applied to a broad class of optimization problems.

INTRODUCTION

Genetic algorithms can be used directly to solve unconstrained optimization problems only. For solving the constrained optimization problems, either genetic algorithm (GA) operations have to be modified or the problem has to be transformed to an unconstrained problem before the genetic algorithm can be employed. In this work, we focus on the second approach.

A number of papers have been published recently for solving optimization problems by combining a simple penalty function with a single penalty function coefficient and genetic algorithms. The quadratic penalty function is the most common function used in the penalty function method (Goldberg and Samtani 1986; Hajela 1990; and Adeli and Cheng 1993). The objective function associated with the penalty function is penalized whenever some of the constraints are violated. Theoretically, the penalty decreases when the value of the penalty function coefficient is increased and convergence is achieved by increasing the penalty function coefficient to infinity. However, a large value for the penalty function coefficient causes ill conditioning in the optimization process and results in numerical instability or slow convergence. Furthermore, since the value of the penalty function coefficient is unknown, it requires numerous numerical experiments and/or experience in order to choose a suitable value and updating process for the penalty function coefficient.

To circumvent the problem of the unpredictability of the penalty function coefficient, we use the augmented Lagrangian method in this work. The augmented Lagrangian (Lagrange multiplier) method was introduced by Powell (1969) for solving the equality constrained optimization problems and extended by Fletcher (1975) for solving the general constrained optimization problems. The basic idea is to integrate the penalty function method

[1]Prof., Dept. of Civ. Engrg., Ohio State Univ., 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210.

[2]Grad. Student, Dept. of Civ. Engrg., Ohio State Univ., 470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH.

Note. Discussion open until June 1, 1994. To extend the closing date one month, a written request must be filed with the ASCE Manager of Journals. The manuscript for this paper was submitted for review and possible publication on May 28, 1992. This paper is part of the ***Journal of Aerospace Engineering***, Vol. 7, No. 1, January, 1994. ©ASCE, ISSN 0893-1321/94/0001-0104/$1.00 + $.15 per page. Paper No. 4136.

with the primal-dual method, which is based on sequential minimization of the Lagrangian function. Instead of only a single penalty function coefficient, in the augmented Lagrangian method, two parameters associated with each constraint are used. Thus, two parameter sets are used instead of a single parameter. In this method, there is no need for the Lagrange multipliers to go to infinity to ensure convergence. Powell (1969) and Fletcher (1975) also proposed an outer loop to adjust these coefficients at some specific points automatically according to previous information about constraints. Furthermore, derivatives of objective function and constraints are not necessary. Powell (1969) showed that this algorithm has linear convergence (that is, convergence rate is constant).

GENETIC ALGORITHM

Genetic algorithms, based on the Darwinian survival-of-the-fittest theory, is an efficient and broadly applicable global search algorithm (Goldberg 1989). The basic mechanism of natural evolution is used in this algorithm to yield the best or better characters among the old population and create superior offsprings. In contrast to conventional search techniques, genetic algorithm starts from a group of points coded as finite length alphabet strings (chromosomes) instead of one real parameter set (called starting point in mathematical programming algorithms). Furthermore, genetic algorithm is not a hill-climbing algorithm so that the derivative information and step size calculation are not necessary. Therefore, the so called local hill-climbing problem, that is the entrapment of the solution in a local minimum is avoided (Adeli and Cheng 1993).

Genetic algorithms consists of three basic operations: reproduction, crossover, and mutation. Reproduction is a process to decide which strings will survive in the next generation and how many copies of them should be produced in the mating pool according to their corresponding fitnesses. Suppose we have a population of size n. Then, the probability of the ith individual string to be selected into the mating pool is

$$ps_i = \frac{f_i}{f_{sumi}} \quad \ldots\ldots (1)$$

where f_i = the fitness of the ith individual string and f_{sumi} is the summation of the fitnesses. Theoretically, the number of copies for the ith individual string is given by

$$\text{num}_i = n \times ps_i = \frac{f_i}{f_{\text{ave}}} \quad \ldots\ldots (2)$$

where f_{ave} and f_{sumi}/n is the average fitness of the current iteration. Thus, the better strings (those with a smaller value of fitness in minimization problems) receive more copies for mating so that their desirable characters and string patterns (schemata) (Holland 1975) may be passed onto their offsprings. In this process, each string is decoded into a set of real numbers in order to calculate its corresponding fitness. A binary string can be decoded by the following formula:

$$C = C_{\min} + \frac{B}{2^L}(C_{\max} - C_{\min}) \quad \ldots\ldots (3)$$

where C = the value the string represents, $C_{\min}$ and $C_{\max}$ = the lower and

upper bounds for C, L = the length of the binary string, and B = the integer value of the binary string.

Crossover and mutation are both random operations. No prior information is used in these operations. To perform crossover, two newly reproduced strings (parents) have to be selected randomly from the mating pool. Then, two new strings (children) are created by exchanging some portions randomly between the selected parents. Many different crossover strategies have been used in the literature. In this work, we employ the two-point and uniform crossover strategies. Mutation is simply a process to change some bits (genes) in all the strings according to the mutation rate. To reflect the fact that mutation occurs infrequently in the real world, it is usually performed with a small probability.

AUGMENTED LAGRANGIAN METHOD

In optimization problems often an objective (cost) function, a function of design variables, is minimized. Most design optimization problems must also satisfy certain equality and inequality constraints. An optimum design problem with n design variables and m constraints (j inequality and m-j equality constraints) can be defined as: minimize $f(\mathbf{X})$

$$\text{subject to } g_i(\mathbf{X}) \leq 0 \qquad i = 1, \ldots, j \qquad \ldots\ldots (4)$$

$$\text{and } h_i(\mathbf{X}) = 0 \qquad i = j + 1, \ldots, m \qquad \ldots\ldots (5)$$

where $\mathbf{X} = (X_1, \ldots, X_n)$ = the vector of design variables.

Powell (1969) combined the penalty function method with the primal-dual method and reported a new penalty function in the following form for finding a local minimum of the equality constraint problem:

$$P[h(\mathbf{X}), \boldsymbol{\gamma}, \boldsymbol{\mu}] = \frac{1}{2} \sum_{i=1}^{m} \gamma_i [h_i(\mathbf{X}) + \mu_i]^2 \qquad \ldots\ldots (6)$$

where $\gamma_i > 0$ and μ_i = real parameters associated with the ith equality constraint. The constrained problem is converted into an unconstrained problem by minimizing the following function:

$$\phi(\mathbf{X}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = f(\mathbf{X}) + P[h(\mathbf{X}), \boldsymbol{\gamma}, \boldsymbol{\mu}] \qquad \ldots\ldots (7)$$

This is the so-called Lagrange multiplier or the augmented Lagrangian method and $r_i\mu_i = \lambda_i$ is called the Lagrange multiplier of the constraint $h_i(\mathbf{X}) = 0$.

Fletcher (1975) modified Powell's Lagrange multiplier method for solving the inequality constrained optimization problem by introducing the penalty function

$$P[g(\mathbf{X}), \boldsymbol{\gamma}, \boldsymbol{\mu}] = \frac{1}{2} \sum_{i=1}^{m} \gamma_i \{[g_i(\mathbf{X}) + \mu_i]^+\}^2 \qquad \ldots\ldots (8)$$

In this case, the constrained problem is converted into an unconstrained problem by minimizing the following function:

$$\phi(\mathbf{X}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = f(\mathbf{X}) + \frac{1}{2} \sum_{i=1}^{m} \gamma_i \{[g_i(\mathbf{X}) + \mu_i]^+\}^2 \qquad \ldots\ldots (9)$$

where

$$[g_i(\mathbf{X}) + \mu_i]^+ = \max[0, g_i(\mathbf{X}) + \mu_i] \qquad \ldots\ldots (10)$$

The well-known quadratic penalty function method can be derived from (9) and (10) by setting $\mu_i = 0$ ($i = 1, \ldots, m$) and the optimization process is completed by increasing the penalty function coefficients γ_i ($i = 1, \ldots, m$) to infinity. However, large values of penalty function coefficients cause ill-conditioning in the solution of the optimization problem and result in slow convergence or numerical instability. Powell (1969), on the other hand, introduced an outer loop to update the Lagrange multipliers $\lambda_i = \gamma_i \mu_i$ automatically according to the information obtained in previous iterations. There is no need for penalty function coefficients or Lagrange multipliers to go to infinity in order to ensure convergence. Furthermore, derivatives of objective function and constraints are not necessary in the coefficient-updating process.

FORMULATION OF STRUCTURAL OPTIMIZATION PROBLEM

The augmented Lagrangian genetic algorithm developed in this work is general. But, in this work we apply it to axial force space structures only. The structural optimization of axial force (truss) structures with N elements and M degrees of freedom can be defined as follows: find the vector of the design variables, cross-sectional area $\mathbf{A} = (A_1, \ldots, A_N)$, such that

$$W(\mathbf{A}) = \sum_{i}^{N} \rho_i L_i A_i = \mathbf{L}^T \boldsymbol{\rho} \mathbf{A} \quad \ldots\ldots (11)$$

is minimized, subject to the following stress, displacement, and fabricational constraints:

$$\boldsymbol{\sigma}^L \leq \boldsymbol{\sigma} \leq \boldsymbol{\sigma}^U \quad \ldots\ldots (12)$$

$$\mathbf{d}^L \leq \mathbf{d} \leq \mathbf{d}^U \quad \ldots\ldots (13)$$

$$\mathbf{A}^L \leq \mathbf{A} \leq \mathbf{A}^U \quad \ldots\ldots (14)$$

where the function $W(\mathbf{A})$ = the total weight of the structure. The term $\boldsymbol{\rho}$ = a diagonal matrix of weights per unit volume and $\mathbf{L}$ = the vector of element lengths. The terms $\mathbf{A}^U$, $\mathbf{A}^L$, $\boldsymbol{\sigma}^U$, $\boldsymbol{\sigma}^L$, $\mathbf{d}^U$, $\mathbf{d}^L$ represent the upper and lower bounds of the vectors of cross-sectional areas $\mathbf{A}$ (with N elements), element stresses $\boldsymbol{\sigma}$ (with N elements), and nodal displacements $\mathbf{d}$ (with M elements, equal to the number of degrees of freedom of the structure). The nodal displacements and element stresses are found by a finite element structural analysis for every string (set of design variables) in every iteration using the following equations:

$$\mathbf{K}\mathbf{d} = \mathbf{P} \quad \ldots\ldots (15)$$

$$\boldsymbol{\sigma} = \mathbf{T}\mathbf{d} \quad \ldots\ldots (16)$$

where $\mathbf{K}$ = the stiffness matrix of the structure, $\mathbf{P}$ = the external nodal force vector, and $\mathbf{T}$ = a transformation matrix relating nodal displacements to element stresses.

Since genetic algorithm can be used directly for unconstrained optimization problems only, we transform the constrained problem to an unconstrained problem using the aforementioned augmented Lagrangian method. We define the following penalty function for optimization of structures:

$$P(\boldsymbol{\sigma}, \mathbf{d}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = \frac{1}{2}\left\{\sum_{1}^{N} \gamma_i \left[\left(\frac{|\sigma_i|}{|\sigma_i^a|} - 1 + \mu_i\right)^+\right]^2 + \sum_{1}^{M} \gamma_{i+N} \left[\left(\frac{|d_i|}{|d_i^a|} - 1 + \mu_{i+N}\right)^+\right]^2\right\} \quad \text{..... (17)}$$

In this case the corresponding unconstrained optimization problem becomes:

$$\text{minimize } \phi(\mathbf{A}, \boldsymbol{\gamma}, \boldsymbol{\mu}) = \frac{1}{L_f}\sum_{1}^{N} \rho_i L_i A_i + \frac{1}{2}\left\{\sum_{1}^{N} \gamma_i \left[\left(\frac{|\sigma_i|}{|\sigma_i^a|} - 1 + \mu_i\right)^+\right]^2 + \sum_{1}^{M} \gamma_{i+N} \left[\left(\frac{|d_i|}{|d_i^a|} - 1 + \mu_{i+N}\right)^+\right]^2\right\} \quad \text{..... (18)}$$

where L_f = a factor for normalizing the objective function, σ_i = the stress in member i, d_i = the displacement in the direction of degree of freedom i, and

$$\sigma_i^a = \sigma_i^L \quad \text{when } \sigma_i < 0 \quad \text{..... (19a)}$$

$$\sigma_i^a = \sigma_i^U \quad \text{when } \sigma_i \geq 0 \quad \text{..... (19b)}$$

$$d_i^a = d_i^L \quad \text{when } d_i < 0 \quad \text{..... (20a)}$$

$$d_i^a = d_i^U \quad \text{when } d_i \geq 0 \quad \text{..... (20b)}$$

$$\left(\frac{|\sigma_i|}{|\sigma_i^a|} - 1 + \mu_i\right)^+ = \max\left(\frac{|\sigma_i|}{|\sigma_i^a|} - 1 + \mu_i, 0\right) \quad \text{..... (21)}$$

$$\left(\frac{|d_i|}{|d_i^a|} - 1 + \mu_{i+N}\right)^+ = \max\left(\frac{|d_i|}{|d_i^a|} - 1 + \mu_{i+N}, 0\right) \quad \text{..... (22)}$$

The normalizing factor L_f is used to make the terms in the objective and penalty functions dimensionally consistent. The magnitude of this factor is chosen to make the two terms in the right-hand side of (18) numerically close to each other, so that one term does not dominate the other. In the GA terminology, (18) is called the fitness function which is used in the reproduction phase in order to guide the genetic search.

AUGMENTED LAGRANGIAN GENETIC ALGORITHM FOR OPTIMIZATION OF STRUCTURES

We present a hybrid structural optimization algorithm by integrating genetic algorithm with augmented Lagrangian method in a nested loop. The outer loop is used to update the Lagrange multipliers according to the augmented Lagrangian method, similar to the first algorithm in Belegundu and Arora (1984). The inner loop performs the genetic algorithm to minimize the penalized objective function associated with the Lagrange multipliers in the outer loop. The hybrid algorithm is presented in seven steps. The inner loop is represented by step 3.

1. Step 1. Set $I = 1$ and $K = \infty$, initialize the values of vectors $\boldsymbol{\gamma}$ and

$\boldsymbol{\mu}$, and choose the values of parameters $\alpha > 1$, $\beta > 1$, $\varepsilon > 0$ and the normalizing factor L_f, where I = a counter for parameter set $\boldsymbol{\mu}$, and ε = the stopping criterion for the outer loop (desired accuracy).

2. Step 2. Generate the chromosome or string (design) population, $\mathbf{A}_j^{(1)}$ ($j = 1, \ldots, n$), for the first iteration randomly where n = an even number representing the size of string population.

3. Step 3. Set the counter of inner loop, J, equal to zero and perform the genetic search to minimize $\phi(\mathbf{A}, \boldsymbol{\gamma}, \boldsymbol{\mu}^{[I]})$ subject to the lower and upper bounds of $\mathbf{A}$.

a. Set $J = J + 1$.

b. Decode each chromosome or string (design) using (3) and find the element stresses and nodal displacements for each string using finite element structural analysis.

c. Calculate the fitness of each string using (18), which combines the objective function with the penalty function. Since we have a minimization problem, rescale the fitness for each string using the following formulas:

$$\phi[\mathbf{A}, \mathbf{r}, \boldsymbol{\mu}^{(I)}] = D_{\max} - \phi[\mathbf{A}, \mathbf{r}, \boldsymbol{\mu}^{(I)}] \quad \text{when } \phi[\mathbf{A}, \mathbf{r}, \boldsymbol{\mu}^{(I)}] < D_{\max} \quad \ldots\ldots (23a)$$

$$\phi[\mathbf{A}, \mathbf{r}, \boldsymbol{\mu}^{(I)}] = 0 \quad \text{when } \phi[\mathbf{A}, \mathbf{r}, \boldsymbol{\mu}^{(I)}] \geq D_{\max} \quad \ldots\ldots (23b)$$

so that the strings with fitnesses greater than or equal to the value of $D_{\max}$ are discarded with no chance to enter the mating pool. Thus, the smaller fitness string receives a higher probability of survival. In this work, we set $D_{\max}$ equal to the average fitness for the population.

d. Reproduce strings (designs) into the mating pool according to the rescaled fitness just calculated. Each rescaled fitness corresponding to a string is divided by the summation of the rescaled fitnesses and

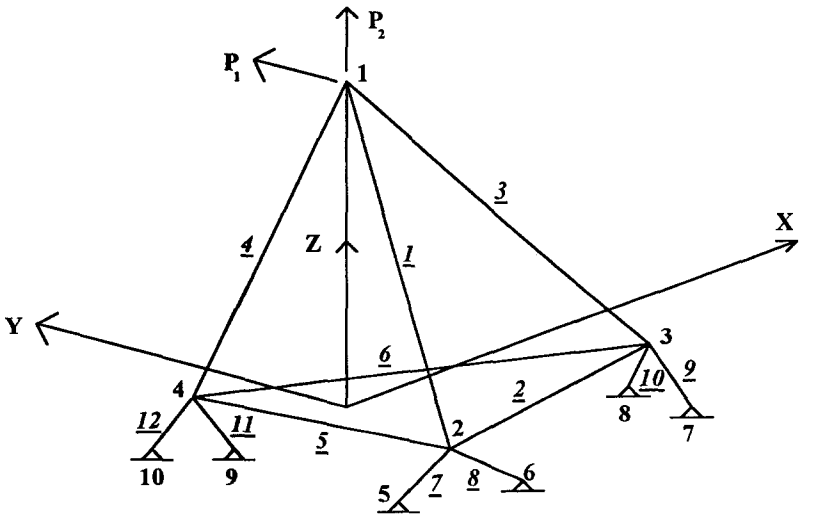

FIG. 1. 12-Bar Truss

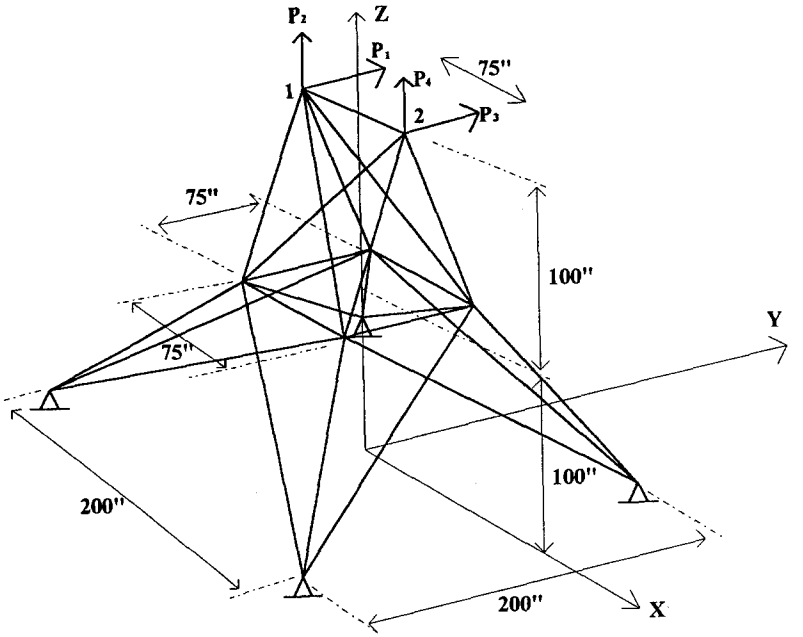

FIG. 2. 25-Bar Truss (1 in. = 0.0254 m)

consequently scaled to a value between 0.0 and 100%. Thus, better strings occupy bigger portions on the range and consequently receive more copies during the reproduction phase. Then, n numbers between 0.0 and 100% are chosen randomly and compared with the aforementioned range in order to select n preferred strings (designs) and include them into the mating pool.

e. Match the strings (designs) in the mating pool randomly, two at a time, and apply crossover and mutation operations to create new offsprings (new designs). If the uniform crossover is used, a mask (Adeli and Cheng 1993) should be created randomly at the beginning of each iteration.

f. Replace old strings by the offsprings and go to the first part of step 3 until the stopping criterion (for inner loop) is met or $J = J_{\max}$. The population of new strings is represented by $\mathbf{A}^{(J)} = (\mathbf{A}_1^{(J)}, \ldots, \mathbf{A}_n^{(J)})$, and the string (design) with smallest fitness in this population is represented by $\mathbf{A}^{(J)*}$.

4. Step 4. Evaluate the values of constraints, $g_i(\mathbf{A}_j^{(J)})$ $(i = 1, \ldots, N + M)$ for every string and calculate the average value for each constraint i as follows:

$$g_{i_{\text{ave}}}^{(J)} = \frac{\sum_{j=1}^{n} g_i[A_j^{(J)}]}{n} \quad \ldots\ldots (24)$$

Set

$$K^* = \max_i |\max[g_{i_{\text{ave}}}^{(J)}, -\mu_i^{(J)}]| \quad \ldots\ldots (25)$$

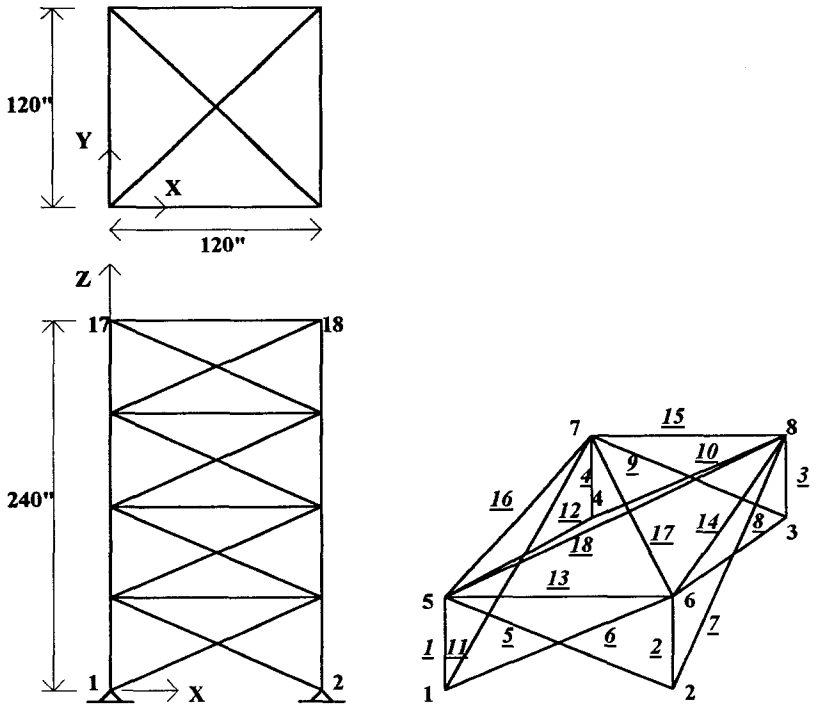

FIG. 3. 72-Bar Truss (1 in. = 0.0254 m)

and

$$\mathbf{I} = \left\{ i : |\max[g_{i_{\text{ave}}}^{(I)}, -\mu_i^{(I)}]| > \frac{K}{\alpha} \right\} \quad \ldots\ldots (26)$$

If $K^* \leq \varepsilon$ (stopping criterion for outer loop), then terminate the run and $\mathbf{A}^{(I)*}$ is the solution. Otherwise go to the next step. Note that there is no guarantee that the solution found here is feasible, and the constraints must be examined when the solution is selected for design among the population (Adeli and Cheng 1993).

5. Step 5. If $K^* \geq K$, update γ_i and $\mu_i^{(I)}$ using the following equations for all $i \in \mathbf{I}$, increase the counter I by one, and go to step 3

$$\gamma_i = \beta\gamma_i \quad \ldots\ldots (27)$$

$$\mu_i^{(I)} = \frac{\mu_i^{(I)}}{\beta} \quad \ldots\ldots (28)$$

Otherwise go to the next step.

6. Step 6. Set

$$\boldsymbol{\mu}_i^{(I+1)} = \boldsymbol{\mu}_i^{(I)} + \max[g_{i_{\text{ave}}}^{(I)}, -\mu_i^{(I)}] \qquad (i = 1, \ldots, N + M) \quad \ldots\ldots (29)$$

If $K^* \leq K/\alpha$, set $K = K^*$ and $I = I + 1$, and go to step 3, otherwise go to the next step.

7. Step 7. Update $\gamma_i = \beta\gamma_i$ and $\mu_i^{(I+1)} = \mu_i^{(I+1)}/\beta$ for each $i \in \mathbf{I}$, and $K = K^*$. Then increase the counter I by 1 and go to step 3.

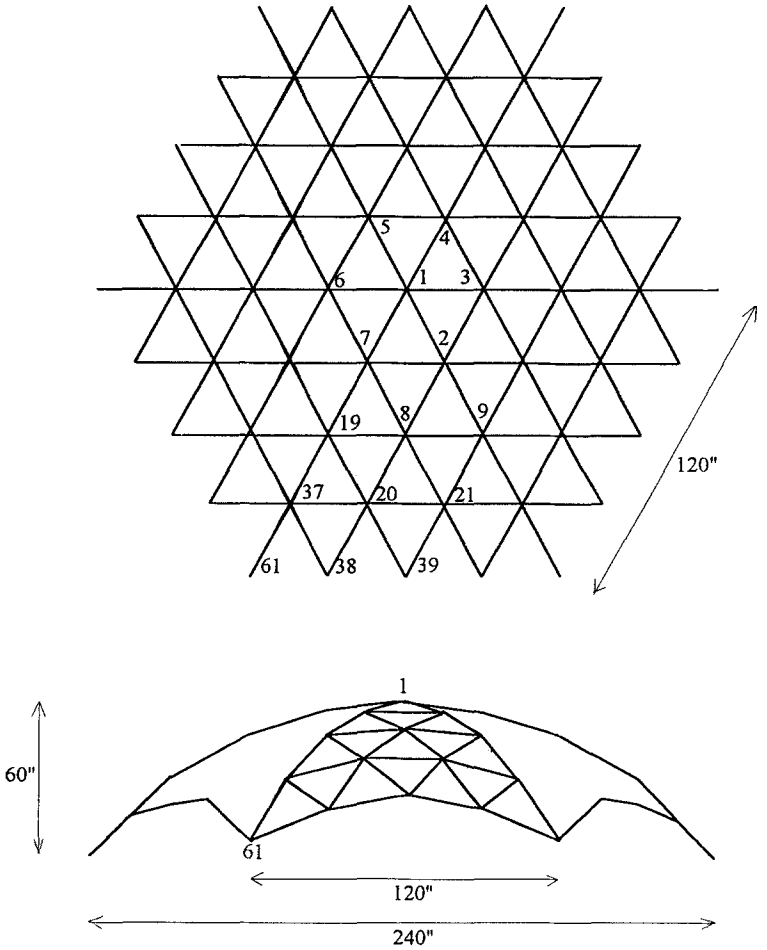

FIG. 4. Geodesic Dome Space Truss (1 in. = 0.0254 m)

We have implemented this augmented Lagrangian genetic algorithm in C language. In order to study the performance of the augmented Lagrangian genetic algorithm, the values of following parameters are changed: *nump*: size of population; *numb*: number of bits for each parameter; and *cross*: 1 for one-point crossover, 2 for two-point crossover, and greater than 2 for uniform crossover.

EXAMPLES

Using the aforementioned algorithm, the minimum weight design of four space structures (trusses), a 12-bar truss (Fig. 1), a 25-bar truss (Fig. 2), a 72-bar truss (Fig. 3), and a Geodesic dome space truss (Fig. 4) are presented in this section. The second problem has also been solved by Kirsch and Taye (1989). The other three problems have also been solved by Adeli and Kamal (1986, 1992), and others. These problems were deliberately chosen

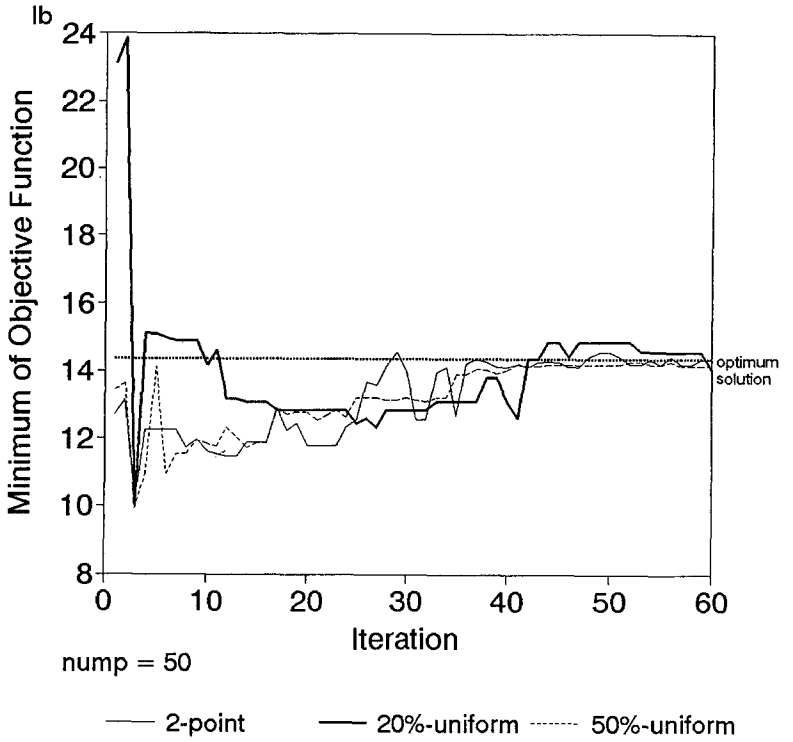

FIG. 5. Convergence History of Minimum Value of Objective Function (Weight of Structure) for Example 1 (1 lb = 4.45 N)

from the literature in order to verify the results obtained from the algorithm presented in this paper. Ten-bit strings are used to represent design variables (cross-sectional areas). The rates (probabilities) of crossover and mutation are set to 0.8 and 0.005, respectively, as recommended by De Jong (1975). We found slow convergence rate using one-point crossover; thus, we present the results for two-point and uniform crossovers only.

As recommended by Belegundu and Arora (1984), the initial values of $\mu_i^{(1)}$ ($i = 1, \ldots, N + M$) are set equal to zero, the initial values of γ_i ($i = 1, \ldots, N + M$) are set equal to 3, and parameters α and β are set equal to 1.5 and 10.0, respectively. The maximum numbers of iterations for inner and outer loop are set equal to eight and 10, respectively. Thus, the total number of iterations in all examples is limited to 80. In all examples, values of 2779.48 kg/m^3 (0.1 lb./in.^{3}) and 69 GPa (10,000 ksi) are used for unit weight (ρ) and the modulus of elasticity (E).

For each example, three cases were run using the same size of population and three different crossover operations: two-point, 50%-uniform, and 20%-uniform crossover. The optimum solutions obtained by the augmented Lagrangian genetic algorithm are comparable with those of the previously reported solutions by Adeli and Kamal (1986, 1992) using the general geometric programming technique and concurrent optimality criteria approach, and Kirsch and Taye (1989) using the design planes method.

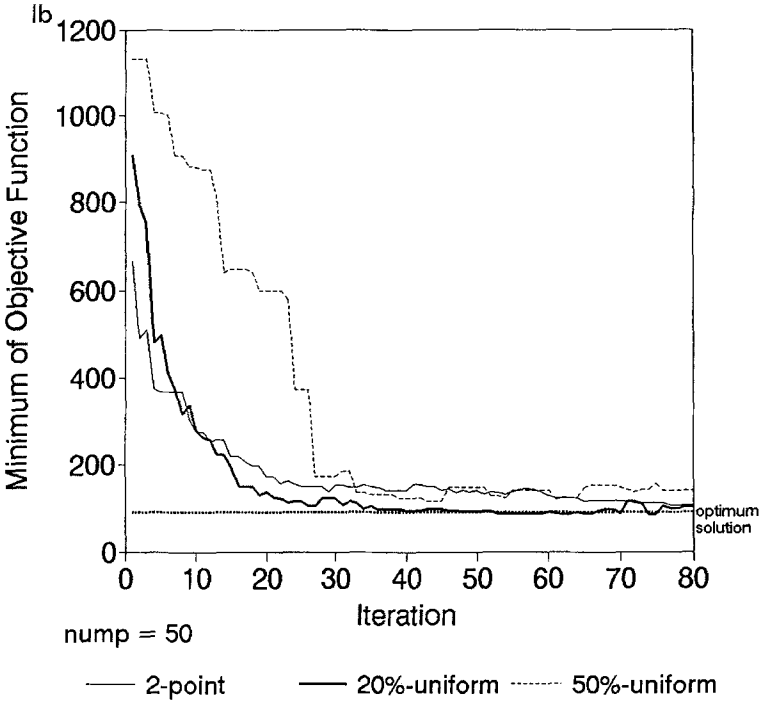

FIG. 6. Convergence History of Minimum Value of Objective Function (Weight of Structure) for Example 2 (1 lb = 4.45 N)

Example 1. 12-Bar Space Truss

Three design variables are used to represent the cross-sectional areas of three element groups in this structure (Fig. 1) [refer to Adeli and Kamal (1986) for dimensions of this structure and the details of the element groups]. The magnitudes of the external forces applied at node 1 in the y- and z-direction are $P_1 = -266.88$ kN (-60 kips) and $P_2 = 66.72$ kN (15 kips). The lower and upper bounds of cross-sectional areas are chosen as 0.000065 m^2 (0.1 sq in.) and 0.00258 m^2 (4.0 sq in.). The element stresses are limited to 140 MPa (20.0 ksi) in tension and 105 MPa (15.0 ksi) in compression. The displacement constraints are 0.0102 m (0.4 in.) along y-axis and 0.0064 m (0.25 in.) along z-axis at node 1. A value of 100.0 is used for the normalizing factor L_f. The size of population is fixed at 50. The convergence history of the minimum value of the objective function (weight of the structure) is presented in Fig. 5.

Example 2. 25-Bar Space Truss

The dimensions and loads are indicated in Fig. 2. Seven design variables are used to represent the cross-sectional areas of seven element groups in this structure [refer to Kirsch and Taye (1986) for the details of the element groups]. The magnitudes of the external forces applied at nodes 1 and 2 in the y- and z-direction are $P_1 = 88.96$ kN (20 kips), $P_2 = -22.24$ kN (-5 kips), $P_3 = -88.96$ kN (-20 kips), and $P_4 = -22.24$ kN (-5 kips), respectively. The lower and upper bounds of cross-sectional areas are se-

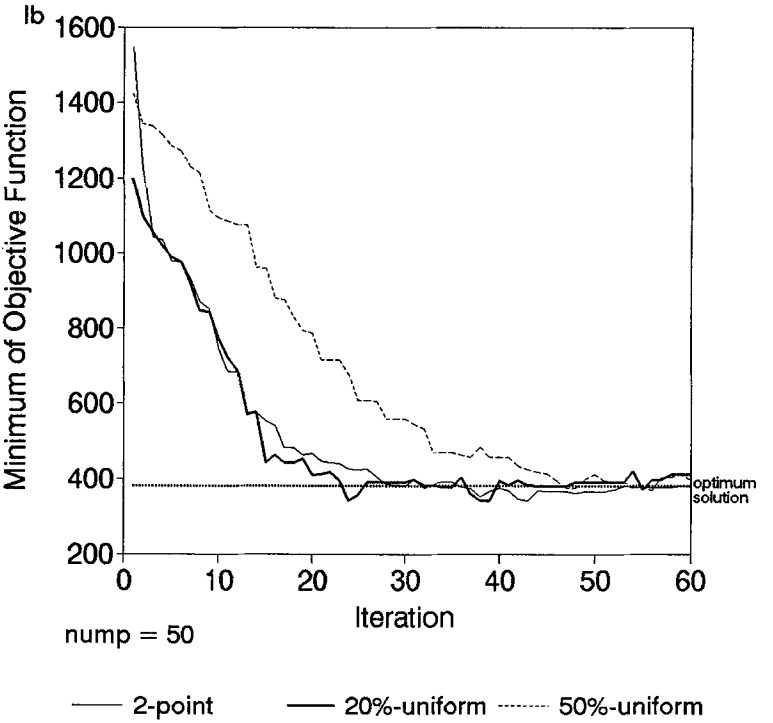

FIG. 7. Convergence History of Minimum Value of Objective Function (Weight of Structure) for Example 3 (1 lb = 4.45 N)

lected as 0.000065 m^2 (0.1 sq in.) and 0.00710 m^2 (11.0 sq in.). The element stresses are limited to 275 MPa (40.0 ksi) in both tension and compression. No displacement constraint is used in this example. A value of 400.0 is used for the normalizing factor L_f. The size of population is fixed at 50. The convergence history of the minimum value of the objective function is presented in Fig. 6. This figure shows the two-point and 20%-uniform crossover yield better results than the 50%-uniform crossover.

Example 3. 72-Bar Space Truss

The dimensions are indicated in Fig. 3. Sixteen design variables are used to represent the cross-sectional areas of 16 element groups in this structure [refer to Adeli and Kamal (1986) for details of the element groups]. The magnitudes of the external forces applied at node 17 in the x-, y-, and z-directions are $P_1 = 22.24$ kN (5 kips), $P_2 = 22.24$ kN (5 kips), and $P_3 = -22.24$ kN (-5 kips), respectively. The lower and upper bounds of cross-sectional areas in this example are 0.000065 m^2 (0.1 sq in.) and 0.0032 m^2 (5.0 sq in.). The element stresses are limited to 170 MPa (25.0 ksi) in both tension and compression. The displacement constraints are given as 0.00635 m (0.25 in.) in the directions of x- and y-axes at nodes 17, 18, 19, and 20 (at the top level of the truss). A value of 1,000.0 is used for the normalizing factor L_f. The size of the population is fixed at 50. The convergence history of the minimum value of the objective function is presented

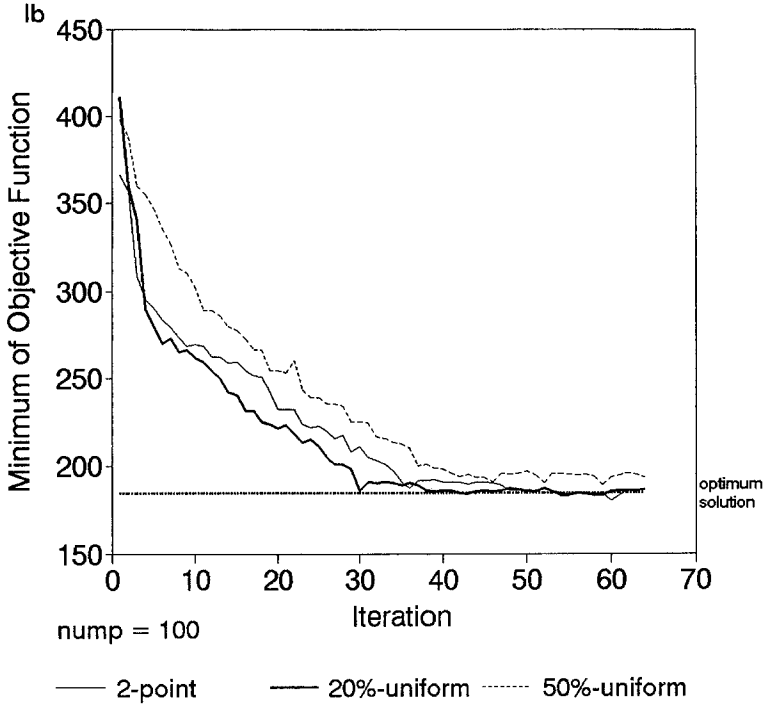

FIG. 8. Convergence History of Minimum Value of Objective Function (Weight of Structure) for Example 4 (1 lb = 4.45 N)

in Fig. 7. This figure shows the two-point crossover yields better results than the other two methods.

Example 4. Geodesic Dome Space Truss

The geodesic dome space truss consists of 132 elements and 61 nodes. The dimensions are indicated in Fig. 4. Because the structure is symmetric about the x- and y-axis, the number of design variables is reduced to 36, the loading on the structure consists of a downward vertical load of 4.448 kN (1 kip) at each node. The lower and upper bounds of cross-sectional areas in this example are 0.000065 m^2 (0.1 sq in.) and 0.0032 m^2 (5.0 sq in.). The element stresses are limited to 170 MPa (25.0 ksi) in both tension and compression. The displacement constraints are given as 0.00254 m (0.1 in.) in the x-, y-, and z-directions for all nodes. A value of 500.0 is used for the normalizing factor L_f. The size of the population is fixed at 100. The convergence history of the minimum value of the objective function is presented in Fig. 8. This figure shows the 20%-uniform and two-point crossover yield better results than the 50%-uniform crossover.

FINAL COMMENTS AND CONCLUSION

The selection and management of the value of the penalty function coefficient has been a problem in penalty function-based optimization algorithms. Theoretically, the values of the penalty function coefficient should

be increased from a small value to infinity to ensure convergence. In practice, we choose only a sequence of increasing values for penalty function coefficient and stop the optimization process whenever the required accuracy is met. It requires numerous numerical experiments and experience in order to choose suitable values for the penalty function coefficient. If a small value is used as the starting value for the penalty function coefficient, the solution usually drops rapidly to the infeasible region. This is due to the fact that the weight of the objective function is much greater than that of the penalty function and thus resulting in a negligible penalty and an infeasible solution (design). On the other hand, a large starting value for the penalty function coefficient causes ill-conditioning in the optimization solution, slow convergence, or numerical oscillation.

The augmented Lagrangian (Lagrange multiplier) method has been shown to be superior to ordinary penalty function method (Powell 1969; Fletcher 1975; Belegundu and Arora 1984). In this method, there is no need for penalty function coefficients or Lagrange multipliers to go to infinity to achieve the optimum solution. In addition, in contrast to the manual control of penalty function coefficient, an outer loop is used in augmented Lagrangian method to adjust the coefficients automatically at some specific points using the information about constraints.

An attractive characteristic of genetic algorithms is that there is no line search and the problem of computation of derivatives of the objective function and constraints is avoided. This feature of genetic algorithms is maintained in the hybrid genetic algorithm presented in this paper. Compared with the penalty function-based genetic algorithm, only a few additional simple function evaluations are needed in the hybrid genetic algorithm presented in this paper. Furthermore, instead of single penalty function coefficient a set of Lagrange multipliers is used in this algorithm. In a sense each constraint is assigned its own penalty function coefficient, and the coefficients are adjusted individually. Thus, the imbalance of penalties from some specific constraints are avoided resulting in a rapid and stable convergence history.

In the augmented Lagrangian genetic algorithm presented in this paper, the wild-guess or trial-and-error approach for the starting penalty function coefficient and the process of arbitrary adjustments are avoided. There is no need to perform an extensive numerical experimentation for finding a suitable value for the penalty function coefficient for each type or class of optimization problem. Thus, the hybrid genetic algorithm presented in this paper is general and can be applied to a broad class of optimization problems.

APPENDIX. REFERENCES

Adeli, H., and Cheng, N.-T. (1993). "Integrated genetic algorithm for optimization of space structures." *J. Aerosp. Engrg.*, ASCE, 6(4), 315–328.

Adeli, H., and Kamal, O. (1986). "Efficient optimization of space trusses." *Comput. Struct.*, 24(3), 501–511.

Adeli, H., and Kamal, O. (1992). "Concurrent optimization of large structures: part II—applications." *J. Aerosp. Engrg.*, ASCE, 5(1), 91–110.

Belegundu, A. D., and Arora, J. S. (1984). "A computational study of transformation methods for optimal design." *AIAA J.*, 22(4), 535–542.

De Jong, K. A. (1975). "An analysis of the behavior of a class of genetic adaptive systems," PhD thesis, University of Michigan, Ann Arbor, Mich.

Fletcher, R. (1975). "An ideal penalty function for constrained optimization." *J. Inst. Mathematics and Its Applications*, 15(3), 319–342.

Gill, P. E., and Murray, W., eds. (1974). *Numerical methods for constrained optimizations*. Academic Press, New York, N.Y.
Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company, Inc., Reading, Mass.
Goldberg, D. E., and Samtani, M. P. (1986). "Engineering optimization via genetic algorithm." *Proc. Ninth Conf. on Electronic Computation*, 471–482.
Hajela, P. (1990). "Genetic search—an approach to the nonconvex optimization problem." *AIAA J.*, 28(7), 1205–1210.
Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, Mich.
Kirsch, U., and Taye, S. (1989). "Structural optimization in design planes." *Comput. Struct.*, 31(6), 913–920.
Powell, M. J. D. (1969). "A method for nonlinear constraints in minimization problems." *Optimization*, R. Fletcher, ed., Academic Press, London, England.
Syswerda, G. (1989). "Uniform crossover in genetic algorithms." *Proc. Third Int. Conf. on Genetic Algorithms*, George Mason University, June 4–7, Morgan Kaufmann Publishers, Inc., 2–9.