結構最佳化遺傳演算法懲罰因子調整之研究

RESEARCH ON ADJUSTING PENALTY FACTORS IN GENETIC ALGORITHMS FOR STRUCTURAL OPTIMIZATION

鍾 添 東* 溫 中 榮†

Tien-Tung Chung *Chung-Rong Wen*

*副教授 †博士班研究生

*†國立台灣大學機械工程學系

*Associated Professor †Ph.D. Student

*†Department of Mechanical Engineering, National Taiwan University

摘 要

本文研究結構最佳化遺傳演算法中每個世代懲罰參數之調整方法。提出一種新型式的外懲罰函數，並定義兩個控制參數；一個參數爲非合理區個體數目與族群大小之比例，另一個參數爲族群違反限制條件之程度；這兩個參數用於動態控制每一個世代之懲罰項大小。經由所提出之懲罰策略，使得搜尋過程中族群多樣性與選擇壓力可維持適當的平衡，以避免早熟之收斂結果。基於所提出之新改良策略，發展出一套結合遺傳演算法與 ANSYS 有限元素分析軟體的整合型程式；並利用所發展程式，深入研究一些結構設計問題之最佳設計過程；從搜尋結果之比較可知，本文所提出之懲罰策略是可靠的，可以快速求出滿意之收斂結果。

關鍵詞： 遺傳演算法、結構最佳化、懲罰技術。

Abstract

This paper studies the methods for adjusting penalty factors of violated constraints at each generation in the genetic algorithms for structural optimization. A new form of exterior penalty function is proposed, and two control parameters are defined. One parameter concerns with the ratio of the number of infeasible individuals to the population size, and the other concerns with the degree of constraint violation in the population. These two parameters are used to dynamically control the magnitude of the penalty term at each generation. With the proposed penalty strategies, the genetic search process will maintain suitable balance between the population diversity and the selective pressure, and prevent premature convergence. Based on the new improved strategy, an integrated program is also developed by combing genetic algorithms and commercial finite element software ANSYS. With the developed program, optimum design processes of some structural design problems are investigated. From the search result, it shows that the proposed penalty strategy is reliable and results in fast and satisfactory convergent solutions.

Keywords: genetic algorithms, structural optimization, penalty techniques.

1. INTRODUCTION

Structural optimum design has been fairly comprehensive in applications after tens of years of development and becomes the essential tool for structural design. When describing optimization problems, the general mathematical form is described as following:

$$\begin{aligned} &\text{Find } \vec{x},\ \vec{x}=(x_1,\cdots,x_n)\in \mathcal{F}\subseteq \mathcal{S} \\ &\text{such } F(\vec{x})\to \min \\ &\text{subject to } g_i(\vec{x})\le 0 \quad i=1,\cdots,n_C \end{aligned} \tag{1}$$

where $\vec{x}$ is design vector, $F(\vec{x})$ is objective function on the search space $\mathcal{S}$, $\mathcal{F}$ is the feasible region defined by the set of constraints $g_i(\vec{x})\le 0$, n_C is the number of constraints.

Many methods have been developed and used for the optimum design of structures. Most of these methods use the nonlinear mathematical programming techniques to find the optimum solutions. Recently, genetic algorithm has been receiving an increasing attention as a novel optimization technique for structural optimization problems. Genetic algorithm was first

developed by John Holland [1]. Goldberg published a book, explaining the theory and application examples of genetic algorithm in details [2]. It uses the basic Darwinian mechanism of "survival of the fittest" and efficiently exploits useful information contained in a population of solutions to generate new solutions with better performance. Through the Goldberg's book, this robust optimization method was thus widely known to the academy. He also developed a program, SGA (Simple Genetic Algorithms), to provide a platform to undertake genetic calculation and analysis with computers. Furthermore, unlike traditional gradient-based method, genetic algorithm requires no computation of sensitivities. It is easy to tackle mix-variable design, and has the better chance to achieve global optimum.

In the past several years, many structural optimization problems have been successfully solved by genetic algorithms. The problems of the optimal design for several truss structures were solved successfully [3~6]. Deb used the genetic algorithms to accomplish the optimal design of welded joints [7]. Chen and Tsao applied genetic algorithms to the optimal design of machine elements [8].

Generally structural design is required to satisfy a number of inequality constraints, for examples, the requirements of stress, deflection and dimensional relationships. Several techniques have been proposed to handle constraints with genetic algorithms. The existing techniques can be roughly classified as: (1) rejecting strategy, (2) repairing strategy, (3) modifying genetic operators strategy, and (4) penalizing strategy [9]. The first three strategies have the advantages that they never generate infeasible solutions, but have the disadvantage that they consider no points outside feasible regions. Because infeasible solutions may take a relatively big portions of the population for highly constrained problem, the optimum solution may be difficult to be found only within feasible regions by applying genetic search. Therefore, the penalizing strategy is adopted to consider the infeasible region in the genetic search. There are comprehensive surveys on constraint handling techniques available in the literatures [9~11].

Joines and Houck proposed a technique in which dynamic penalties (non-stationary function method) are used [12]. The penalty term is not constant and changes with the generation number. The selective pressure on infeasible solutions is dynamically increased throughout the whole generations. At the end stage of the evolution process, this penalty becomes a very large value and may produce premature convergence. Once the population is trapped in a feasible (or infeasible) local optimum, it may stagnate there forever. Coit, Smith and Tate proposed a method of adapting penalties that uses a penalty function which takes a feedback from the search process [13~15]. The magnitude of the penalty is dynamically modified according to the fitness of the best solution found so far, and is proportional to the difference between the best feasible solution ever seen and the best overall solution ever seen. However, this method relies only on the best individuals in the past populations to adjust the penalty, and this might not reflect the realistic situation in the current generation.

The central problem of applying genetic algorithm to constrained structural optimization is how to handle constraints. Besides, the major obstacle for GA applications in structural optimization is the very large number of function evaluations. The cost of finite element analysis for large structures is very high. How to choose the appropriate penalty on the infeasible solution to avoid premature convergence and accelerate the convergence is the main problem. This paper uses genetic algorithms to discuss the effects of penalty techniques on the structural optimization. It also compares with the dynamic penalty function method and the adaptive penalty function method through practical engineering structural optimization problems, and proposes an improved strategy.

This paper is organized as follows: Section 2 briefly introduces genetic algorithms combining with the FEA software ANSYS for structural optimization. Section 3 briefly surveys two constraint handling methods based on penalty functions, the dynamic penalty function method and the adaptive penalty function method. Then, improved strategies on handling constraints are proposed, and the effects of penalty degree on optimization convergence are discussed. Section 4 is devoted to a detailed study of the performance of the improved strategies on three selected structural test cases. Finally, some concluding remarks are contained in section 5.

2. GENETIC ALGORITHMS FOR STRUCTURAL OPTIMIZATION

Genetic algorithm is a stochastic searching method that is different from traditional searching methods. Traditional methods such as nonlinear programming start with a prototype design while genetic algorithms start with a randomly generated design group, called "population". Each individual within the population is called "chromosome", representing the solution to an optimization problem. Chromosome consists of many genes that are generally represented in binary bits. Flowchart of traditional genetic algorithms is shown in

Fig. 1. Selection, crossover, and mutation are the three principal operators in genetic algorithms.

Selection is a process that individual strings are selected as parents to reproduce offspring according to their fitness. The rule is that the best gets more copies and the worst dies off. The roulette wheel selection scheme, i.e., proportional selection scheme, is popular in the literature and is adopted in this study. This scheme is implemented as a linear search through a roulette wheel with each slice weighted in proportion to scaled fitness value [2].

Crossover operates on two individuals at a time and generates offspring by combining both individuals' features. Two individuals in the population will exchange portions of their binary representations. A simple way is to choose a random cut-point and generate offspring by exchanging the segments to the right of this point.

The mutation operator arbitrarily alters one gene value according to a predetermined probability. Despite being regarded as a secondary operator, mutation actually makes an important contribution to the search effectiveness. Selection and crossover generate new individuals, but they do not introduce any new genetic features into the population. The mutation operator introduces diversity and reflects features that are not present in the current population, and therefore can prevent premature convergence.

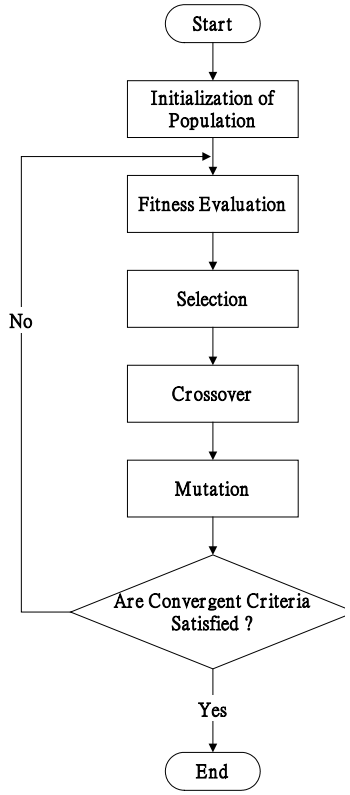

Fig. 1 Flowchart of traditional genetic algorithms

This paper uses commercial FEA software ANSYS in combination with genetic algorithms written in C to construct an optimization system for optimum design of structures. The system is divided into two parts: structural analysis and optimization search. A new design will be generated after one cycle of optimization procedure and transmitted to ANSYS for structural analysis in order to get the objective function value and compare constraints. The results are then transferred to the optimization search procedure for the next generation. The iterative processes are repeated until convergent criteria are satisfied. Figure 2 is the framework of the structural optimization program.

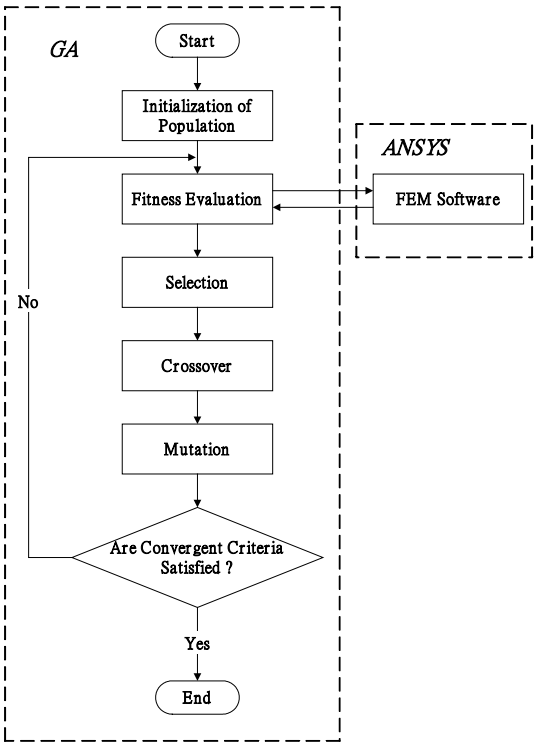

Fig. 2 Framework of structural optimization program using genetic algorithms

3. PENALTY FUNCTION METHOD

The general exterior penalty function method for solving solution of Eq. (1) has the following form [10]:

$$\tilde{F}(\vec{x}) = F(\vec{x}) + P(\vec{x}) \tag{2}$$

$$P(\vec{x}) = \sum_{i=1}^{n_C} r_i < g_i(\vec{x}) >^2$$
$$< g_i(\vec{x}) >= \begin{cases} 0 & \text{if } g_i(\vec{x}) < 0 \\ g_i(\vec{x}) & \text{if } g_i(\vec{x}) \geq 0 \end{cases} \tag{3}$$

where $\widetilde{F}(\vec{x})$ is the penalized fitness function, $P(\vec{x})$ is the penalty function, r_i is the penalty factor for the i-th constraint.

For applying the exterior penalty function method, the focus is on how to design the penalty term $P(\vec{x})$ so that it can effectively guide the searching of genetic algorithms towards the promising area of solution space. This section explains two penalty function methods briefly (dynamic penalty function and adaptive penalty function). Then, we propose an improved strategy for the penalty term when using genetic algorithms in structural optimization.

3.1 Dynamic Penalty Function

The dynamic penalty function was proposed by Joines and Houck [12]. Its fitness function has the following form:

$$\widetilde{F}(\vec{x}) = F(\vec{x}) + (C \times k)^{\alpha} \sum_{i=1}^{n_C} < g_i(\vec{x}) >^{\beta} \tag{4}$$

where C, α and β are constant, k is the generation number. From the item of $(C \times k)^{\alpha}$, we know that the penalty term of infeasible individuals is increased with the generation number, and it becomes very large at the end stage of the evolution.

3.2 Adaptive Penalty Function

Adaptive penalty function was proposed by Smith and Tate [15]. The penalty term is dynamically modified according to the fitness of the best solution found so far. The form is as follows:

$$\begin{aligned} \widetilde{F}(\vec{x}) &= F(\vec{x}) + \left(\frac{n_c}{2}\right)^{r_s} (\Delta F) \\ \Delta F &= F_{\text{feas}} - F_{\text{all}} \end{aligned} \tag{5}$$

where F_{all} is the value of the optimal objective function regardless of the constraints, F_{feas} is the value of optimal objective function satisfying constraints, ΔF is the difference between F_{feas} and F_{all}, and r_s is the severity parameter.

Coit and Smith proposed the concept of Near-Feasible Threshold (NFT) [13], with the idea that when the searched solution falls into the infeasible region, but nears the optimal solution as the point b shown in Fig. 3, the information of this solution should not be discarded. This near-feasible threshold is a narrow band outside the feasible region as shown in Fig. 3.

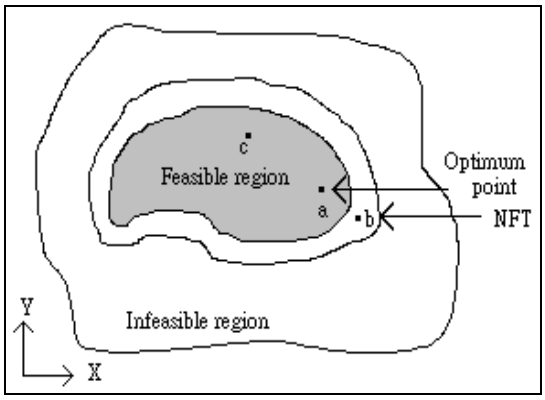

Fig. 3 Near-feasible threshold

The fitness function developed from this near-feasible threshold is as follows:

$$\widetilde{F}(\vec{x}) = F(\vec{x}) + \Delta F < \frac{g_i(\vec{x})}{\varepsilon_i} >^{r_s} \tag{6}$$

where ε_i is the near-feasible threshold, which can be 1% ~ 5% of usual constraint limits.

3.3 Improved Strategies for Penalty Function Method

In general, different values of r_i in Eq. (3) shall be used for different optimization problems to avoid insufficient or extraordinary penalty. For the sake of studying effect of the penalty degree on convergence, r_i is redefined to change with generation. The penalty function is usually simplified as following:

$$P(\vec{x}) = r_k \sum_{i=1}^{n_C} < g_i(\vec{x}) >^2 \tag{7}$$

where r_k is the penalty factor of the k-th generation.

Penalty function method has effect on the selective pressure. Penalty techniques used for constraint handling in structural optimization usually can not decide the appropriate strength of penalty due to unknown characteristics of the problem. If the penalty is too large, the selective pressure for infeasible individuals will be too strong and convergence will be premature. If the penalty is too small, selective pressure will be too light and convergence within specified number of generations can not be achieved.

This paper studies the effect of penalty function on the selective pressure from the following two points of view:

1. Whether each individual in the population is liable to violate constraints?
2. What's the degree of each individual's violation to the constraints?

Based on the above two points, this paper proposes two indices to dynamically adjust suitable penalty strength for each generation so as to accelerate convergence and avoid prematurity. The proposed two indices are c_1 and c_2, expressed as following:

$$c_1 = \frac{n_I}{n_p} \tag{8}$$

$$c_2 = \frac{1}{n_c}\sum_{i=1}^{n_c} \frac{< g_i(\vec{x}_j) >^2}{b_i^{\max}} \tag{9}$$

$$b_i^{\max} = \begin{cases} \max\{< g_i(\vec{x}_j) >^2; j = 1, \cdots, n_p\}, \\ \qquad\qquad \text{for } n_I > 0 \\ \varepsilon, \qquad\qquad \text{for } n_I = 0 \end{cases} \tag{10}$$

where c_1 and c_2 are two indices for measuring penalty factor, n_I is number of infeasible individuals in the population, n_p is the population size, $b_i^{\max}$ is the maximum of violation for constraint i among current population, and ε is a small positive number used to avoid the zero-division problem.

With these two indices, the value of the penalty factor r_k of Eq. (7) are defined as:

$$r_k = h_1(c_1)\, h_2(c_2)\, r_{k-1}, \quad k = 1, \cdots, n_G \tag{11}$$

$$h_1(c_1) = 1 - w + 2wc_1 \tag{12}$$

$$h_2(c_2) = 1 - w + 2wc_2 \tag{13}$$

where h_1 and h_2 are functions of c_1 and c_2, respectively. n_G is the total number of generation, w can be any value from 0.1 to 0.5.

In Eq. (8), c_1 represents the ratio of infeasible individuals to the whole population. From the first point described above and Eq. (8), when the population contains less infeasible individuals at some stage of evolutionary process, penalty factor r_k will be decreased to prevent the loss of information in the infeasible region. On the other hand, if there are more infeasible individuals than feasible individuals in the population, r_k should be increased.

Hence, in order to keep the appropriate ratio between feasible and infeasible individuals in the population, the variation of $h_1(c_1)$ is varied with c_1 (increased or decreased). For example, if we want to keep 50% of infeasible individuals in the population, then the value of c_1 is 0.5, and $h_1(c_1)$ = 1.0. It means that the penalty factor r_k is appropriate and is not increased or decreased. For higher c_1, the $h_1(c_1)$ should be increased, and vice versa. So, $h_1(c_1)$ can be defined as a monotonically increasing function with c_1. Besides, for the reason of avoiding unexpected numerical difficulties, the range of $h_1(c_1)$ is limited in the range from $1 - w$ to $1 + w$ for $0 \le c_1 \le 1$. For example, the construction of $h_1(c_1)$ can be a linearly function expressed as Eq. (12).

From the second point described above and Eq. (9), c_2 measures the severity of constraints violation in the current population. If c_2 is large, it represents violation degree against constraints in the current population is high, then the penalty factor r_k in Eq. (11) is increased in order to force the search towards the feasible region, such that the search may not wander deeply into the infeasible region. The expression of $h_2(c_2)$ is similar to $h_1(c_1)$, and expressed in Eq. (13).

In general, it is better to use a modest penalty in the initial stages to insure adequate sampling of the search space and then gradually increase the penalty to force the optimization process to converge to a feasible solution. The initial penalty factor r_0 is determined using the first population, to try to balance between objective function and constraint violation, and can be expressed as [16]:

$$r_0 = \frac{\sum_{j=1}^{n_P} F(\vec{x}_j)}{\sum_{i=1}^{n_C}\sum_{j=1}^{n_P} < g_i(\vec{x}_j) >^2} \times 1000 \quad \text{if } n_I \neq 0 \tag{14}$$

This improved strategy adjusts the penalties dynamically at each generation in order to achieve a balance between the preservation of information and the selective pressure for infeasibility. According to this approach, the magnitude of the penalty term is actually controlled by two important parameters (c_1 and c_2). With these two control parameters, the genetic algorithm has enough information not only about how many constraints were violated, but also about the degree in which these constraints were violated.

This paper applies the above mentioned improved strategy in practical structural optimization, and compares with the results of the dynamic penalty function method and the adaptive penalty function method.

4. APPLICATION EXAMPLES

Three examples are illustrated to demonstrate the effects of the new penalty strategies described in the preceding section. GA-D represents the genetic algorithm using the dynamic penalty function method and GA-A the adaptive penalty function method. For all examples, the GA parameters are shown in Table 1.

Table 1 GA parameters used in all examples

Population size (Np)	Crossover Rate (Pc)	Mutation Rate (Pm)	Maximum Generation
30	0.8	0.01	100

4.1 Example 1: Torsional Rod Design

The first example is the design of a two-element thin-walled cantilever torsional rod subjected to sinusoidal excitation. The structure is shown in Fig. 4 and has been studied by Hajela [17]. The design problem is to minimize the total structural weight subjected to given stress constraints produced during the dynamic torsional displacement. The design variables are the wall thicknesses of the two elements t_1 and t_2.

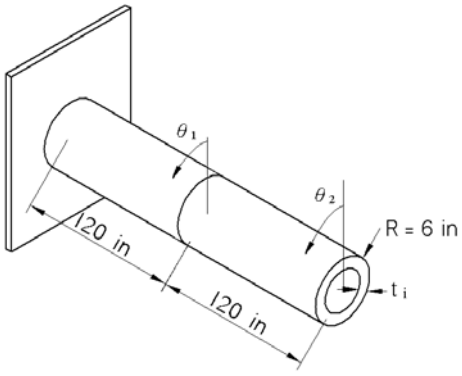

Fig. 4 Two-element torsional rod problem

The mathematical statement of this problem is as follows:

$$\begin{aligned}&\text{Minimize } F(t_1,t_2)=1440\,\pi\rho(t_1+t_2)\\&\text{subject to}\\&\quad g_i=(S_i/S_{\max})^2-1\le 0,\quad i=1,2\\&\text{where}\\&\quad S_1/S_{\max}=3T_n\,t_2(1-\lambda_e)/\mu\\&\quad S_2/S_{\max}=T_n[3\lambda_e\,t_2+t_1(1-2\lambda_e)]/\mu\\&\quad \mu=t_2[t_1(1-2\lambda_e)^2-t_2(6\lambda_e-3\lambda_e^2)]\end{aligned}\qquad(15)$$

ρ is the density, S_i is the stress of i-th element, $S_{\max}$ is the allowable stress, λ_e is a nondimensional measure of the forcing frequency, and T_n is a function of the amplitude of the torsional load. The lower and upper bounds of each design variable are 0.04 and 3in, respectively. At λ_e = 1/6 and T_n = 0.085, the design space is shown in Fig. 5. The exact global minimum point is at t_1 = 0.04 and t_2 = 0.25. The variation of c_1, c_2 and r_k/r_0 is shown in Fig. 6, and the value of r_k/r_0 oscillates between the range from 1.0 to 2.8. The design histories are shown in Fig. 7, and the optimum solutions obtained by different penalty function methods are shown in Table 2.

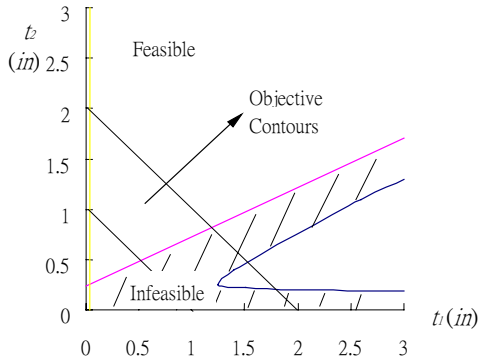

Fig. 5 The design space of the torsional rod problem

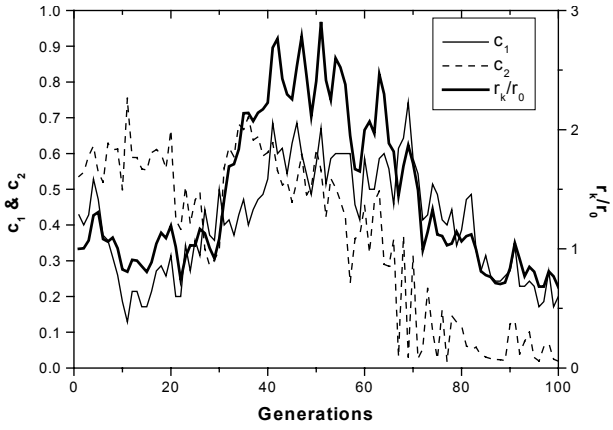

Fig. 6 The variation of c_1, c_2 and r_k/r_0 for the torsional rod problem

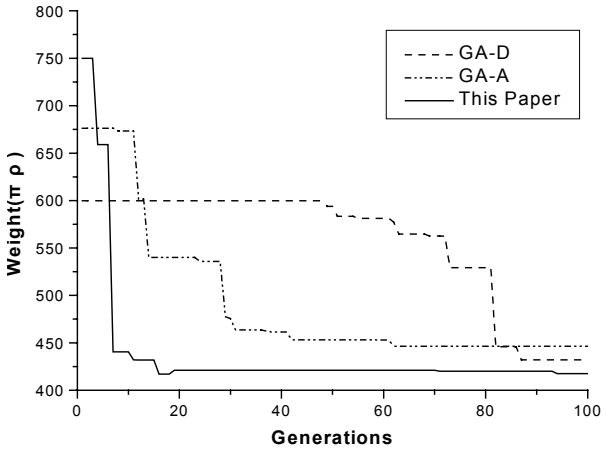

Fig. 7 The design histories of the torsional rod problem

Table 2 Optimum solutions of the torsional rod problem

	This paper	GA-D	GA-A	Exact sol. (Hajela 1990)
t_1	0.04	0.0458	0.04	0.04
t_2	0.251	0.2604	0.2712	0.25
Weight ($\pi\rho$)	419.04	441.561	448.723	417.6

4.2 Example 2: 10-Bar Planar Truss

The second example is the classic 10-bar planar truss shown in Fig. 8 [18]. The cross sectional areas of each bar element are sized to obtain a minimum weight structure subjected to prescribed stress and displacement constraints. There are 10 independent design variables. The displacement constraints are limited to 2in in the *x*- and *y*-directions at each node. The stress constraint requires the stress in each bar element to be less than 25kpsi. The static loads of 100kips are applied at nodes 2 and 3 in the downward direction. The Young's modulus of the material is 10^4ksi, and the weight density is 0.1 lb/in^3. The discrete values for cross-sectional areas are chosen from the AISC Manual, and the values are: S = {1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.5, 13.5, 13.9, 14.2, 15.5, 16.0, 16.9, 18.8, 19.9, 22.0, 22.9, 26.5, 30.0, 33.5} (in^2) [19].

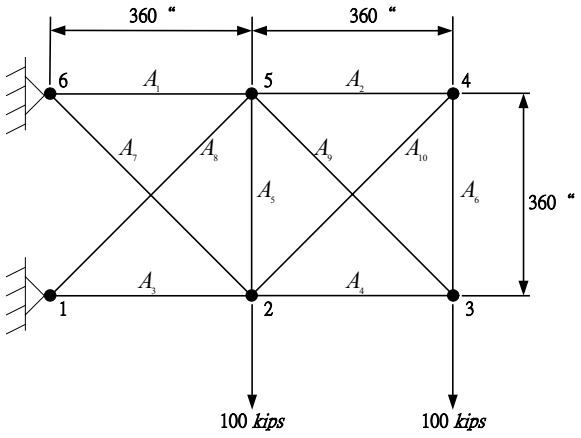

Fig. 8 10-bar planar truss problem

The mathematical statement of this problem can be written as follows:

$$\text{Find } \vec{A}$$

$$\text{which minimize } W(\vec{A}) = \sum_{i=1}^{10} \rho A_i L_i$$

$$\text{subject to } -25\,\text{kips} \le \sigma_i \le 25\,\text{kips} \qquad i = 1, 2, \cdots, 10$$

$$-2\,\text{in} \le u_j, v_j \le 2\,\text{in} \qquad j = 1, 2, \cdots, 6 \tag{16}$$

where A_i is the cross-sectional area of the i-th bar member, L_i is the length of the i-th bar member, σ_i denotes the stress of the i-th bar member and u_j, v_j are the displacement of the j-th node in the x- and y-direction respectively.

The variation of c_1, c_2 and r_k/r_0 is shown in Fig. 9, and the value of r_k/r_0 oscillates between the range from 1.0 to 20.9. The design histories are shown in Fig. 10. Table 3 records the optimum solutions obtained by different penalty function methods, and compares with other methods provided by Rajeev [18]. The solutions of CONMIN and OPTDYN were obtained from traditional optimization method by assuming continuous design spaces, instead of discrete spaces. As shown in Table 3, this study obtains better solution than the other three GA based methods (GA-D, GA-A, and [18]).

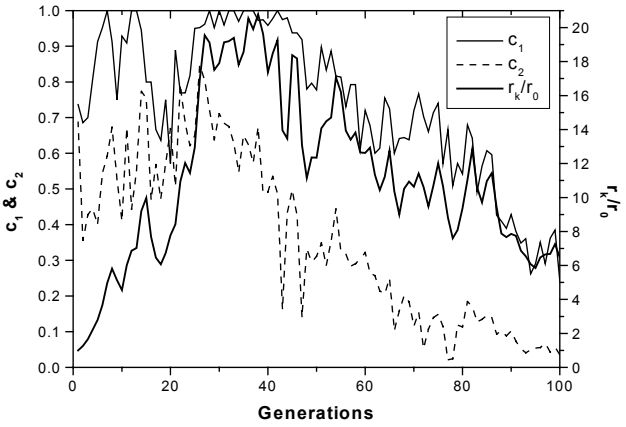

Fig. 9 The variation of c_1, c_2 and r_k/r_0 for the 10-bar planar truss problem

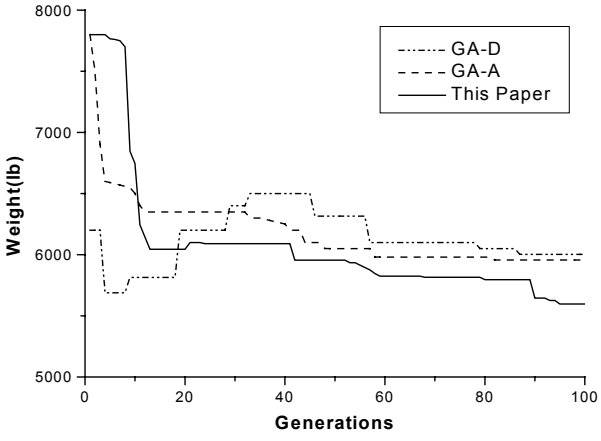

Fig. 10 Design histories for the 10-bar planar truss problem

Table 3 Optimum solutions of 10-bar planar truss problem

	This paper	GA-D	GA-A	Rajeev	OPTDYN	CONMIN
A_1	33.5	30.0	30.0	33.50	25.70	25.20
A_2	1.62	2.38	2.88	1.62	0.10	1.89
A_3	26.5	22.0	26.5	22.00	25.11	24.87
A_4	13.5	26.5	11.5	15.50	19.39	15.83
A_5	1.62	1.62	1.62	1.62	0.10	0.10
A_6	1.80	1.62	2.38	1.62	0.10	1.75
A_7	13.9	11.5	15.5	14.20	15.40	16.76
A_8	18.8	18.8	19.9	19.90	20.32	19.73
A_9	19.9	26.5	26.5	19.90	20.74	20.98
A_{10}	1.80	1.62	2.13	2.62	1.14	2.51
Weight(lb)	5597.04	6002.58	5955.55	5613.84	5472.00	5563.00

4.3 Example 3: 25-Member Space Truss

The third example is a 3-D 25-member transmission tower space truss shown in Fig. 11. The objective is to minimize the weight of the structure. The displacement constraints require the static displacement at each node to be less than 0.35in in x-, y- and z-directions, and each member stress is limited to 40kpsi. The design variables are the cross-sectional areas of the members. To maintain the symmetry of the optimized structure, the design variable linking shown in Table 4 is employed. Loading conditions for this space truss are given in Table 5, and nodal coordinates are given in Table 6. The Young's modulus of the material is 10^4ksi, and the weight density is 0.1 lb/in^3. Discrete values for each cross-sectional area are taken from the available set S = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4} (in^2) [19].

The variation of c_1, c_2 and r_k/r_0 is shown in Fig. 12, and the value of r_k/r_0 oscillates between the range from 1.0 to 22.6. The design histories are shown in Fig. 13, and Table 7 shows the optimum GA solutions using various penalty function methods. From Table 7, it shows that the best solution is obtained in this study.

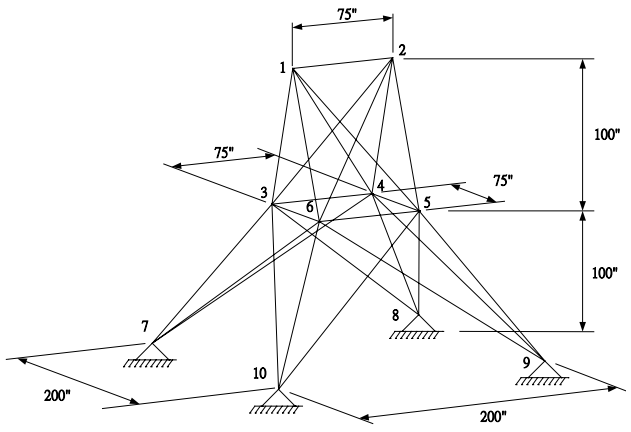

Fig. 11 25-member space truss

Table 4 Design variable linking for 25-member space truss

Design variables	Members
A1	1-2
A2	1-4, 2-3, 1-5, 2-6
A3	2-5, 2-4, 1-3, 1-6
A4	3-6, 4-5
A5	3-4, 5-6
A6	3-10, 6-7, 4-9, 5-8
A7	3-8, 4-7, 6-9, 5-10
A8	3-7, 4-8, 5-9, 6-10

Table 5 Loading conditions for 25-member space truss

Node	F_x (lbs)	F_y (lbs)	F_z (lbs)
1	1000	−10000	−10000
2	0	−10000	−10000
3	500	0	0
6	600	0	0

Table 6 Nodal coordinates of 25-member space truss

Node	X	Y	Y
1	– 37.5	0.0	200.0
2	37.5	0.0	200.0
3	– 37.5	37.5	100.0
4	37.5	37.5	100.0
5	37.5	– 37.5	100.0
6	– 37.5	– 37.5	100.0
7	– 100.0	100.0	0.0
8	100.0	100.0	0.0
9	100.0	– 100.0	0.0
10	– 100.0	– 100.0	0.0

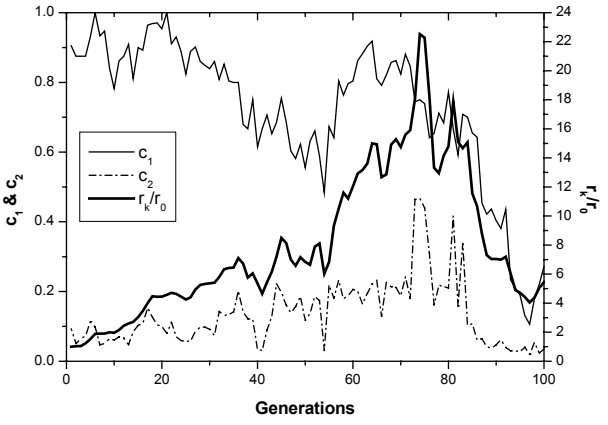

Fig. 12 The variation of c_1, c_2 and r_k/r_0 for 25-member space truss

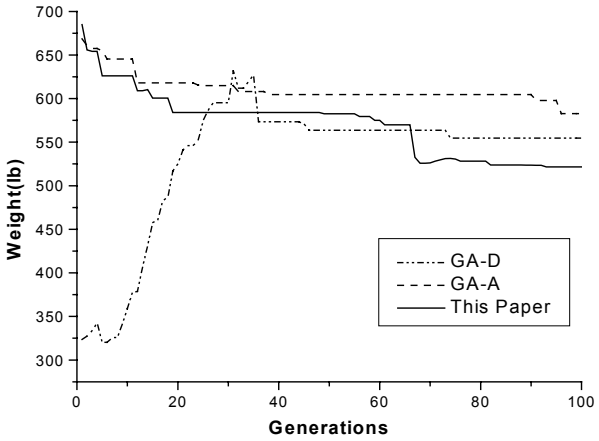

Fig. 13 Convergence histories for the 25-member space truss

Table 7 Optimum solutions of 25-member space truss and other methods in the literature

	This paper	GA-D	GA-A	Rajeev	Zhu
A1	0.10	0.10	0.2	0.10	0.10
A2	0.70	2.10	1.9	1.80	1.90
A3	3.20	1.70	2.6	2.30	2.60
A4	0.50	0.30	0.2	0.20	0.10
A5	0.40	0.30	0.3	0.10	0.10
A6	1.00	1.10	0.7	0.80	0.80
A7	1.40	1.40	2.4	1.80	2.10
A8	3.00	3.40	2.6	3.00	2.60
Weight(lb)	521.548	554.65	582.673	546.01	562.93

5. CONCLUSIONS

The major difficulty of handling constraints using penalty function methods in genetic algorithms is to set appropriate values for the penalty term at each generation and prevent premature convergence. This paper studies the effect of penalty factor in genetic algorithms on structural optimization and proposes an improved strategy for adjusting the penalty term. The improved strategy can automatically adjust the penalty factor at each generation based on two control parameters, one is the ratio of infeasible individuals to the population size, the other is the degree of the constraint violation of the population. With the proposed method, detailed design processes of three examples are studied. The search results are compared to other researchers, and the comparison shows that the proposed method gets more faster and better results.

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial System*, University of Michigan Press, Ann Arbor, Mich, 1975.

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

[3] W. M. Jenkins, "Towards structural optimization via the genetic algorithms," *Computers & Structures*, Vol. 40, No. 5, 1991, pp. 1321–1327.

[4] P. Hajela and C.-Y. Lin, "Genetic search strategies in multicriterion optimal design," *Structural Optimization*, Vol. 4, 1992, pp. 99–107.

[5] S.-J. Wu and P.-T. Chow, "Steady-state genetic algorithms for discrete optimization of trusses," *Computers & Structures*, Vol. 56, No. 6, 1995, pp. 979–991.

[6] T.-Y. Chen and C.-J. Chen, "Improvements of simple genetic algorithm in structural design," *International Journal for Numerical Methods in Engineering*, Vol. 40, 1997, pp. 1323–1334.

[7] K. Deb, "Optimal design of a welded beam structure via genetic algorithms," *AIAA Journal*, Vol. 29, No. 11, 1991, pp. 2013–2015.

[8] J. L. Chen, and Y.-C. Tsao, "Optimal design of machine elements using genetic algorithms," *Journal of the Chinese Society of Mechanical Engineers*, Vol. 14, No. 2, 1993, pp. 193–199.

[9] M. Gen and R. Cheng, "A survey of penalty techniques in genetic algorithms," *Proceedings of the IEEE Conference on Evolutionary Computation*, 1996, pp. 804–809.

[10] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," *Proceedings of the 4th Annual Conference on Evolutionary Programming*, MIT Press, Cambridge, MA, 1995, pp. 135–155.

[11] C. A. Coello, "Towards a survey of constraint handling techniques used with evolutionary algorithms," *Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada*, 1999.

[12] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs," *Proceedings of the First IEEE International Conference on Evolutionary Computation*, IEEE Press, 1994, pp. 579–584.

[13] D. W. Coit and A. E. Smith, "Penalty guided genetic search for reliability design optimization," *International Journal of Computers and Industrial Engineering*, Vol. 30, No. 4, 1996, pp.895–904.

[14] D. W. Coit, A. E. Smith and D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INOFRMS Journal on Computing*, Vol. 8, No. 2, 1996, pp. 173–182.

[15] A. E. Smith and D. M. Tate, "Genetic optimization using a penalty function," *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 499–505.

[16] S. B. Hamida and M. Schoenauer, "An adaptive algorithm for constrained optimization problems," *Proceedings of Parallel Problem Solving from Nature VI*, LNCS 1917, Springer Verlag, 2000, pp. 529–538.

[17] P. Hajela, "Genetic search—an approach to the nonconvex optimization problem," *AIAA Journal*, Vol. 28, No. 7, 1990, pp. 1205–1210.

[18] S. Rajeev and C. S. Krishnamoorthy, "Discrete optimization of structures using genetic algorithms," *Journal of Structural Engineering*, Vol. 118, No. 5, 1992, pp. 1233–1250.

[19] J. S. Arora, *Introduction to Optimum Design*, McGraw-Hill Book Company, 1989.

Tien-Tung Chung (鍾添東) 國立清華大學動力機械學系學士 (1976)、國立台灣大學機械工程學研究所碩士 (1980)、國立台灣大學機械工程學研究所博士 (1986)。曾任國立台灣大學機械系助教、講師，現任國立台灣大學機械系副教授。主要研究領域爲電腦輔助設計、結構分析、最佳化設計等。

Chung-Rong Wen (溫中榮) 國立中山大學機械工程學碩士 (1990)，現爲國立台灣大學機械工程學博士班研究生，1990 年 8 月起任教於建國技術學院機械工程學系。

收稿日期 91 年 5 月 22 日、修訂日期 91 年 6 月 20 日、接受日期 91 年 6 月 20 日
Manuscript received May 22, 2002, revised June 20, 2002, accepted June 20, 2002