Artificial Neural Networks for Evaluation of Constraint Violation in Structural Optimization by Genetic Algorithms

Pruettha Nanakorn and Yupaporn Thanyakriengkrai
Sirindhorn International Institute of Technology, Thammasat University
PO Box 22, Thammasat-Rangsit Post Office
Pathumthani 12121, Thailand
Phone (66) 2986-9009 Ext. 1906
Fax (66) 2986-9009 Ext. 1900
E-Mail: nanakorn@siit.tu.ac.th

Abstract

This study employs artificial neural networks (ANNs) to reduce the amount of time in structural design optimization processes by genetic algorithms (GAs). In structural design optimization, structural analysis is always necessary since it allows structural constraints to be monitored and satisfied during the optimization process. Especially, when a GA is used as an optimizer, due to the nature of GAs, many structural analyses must be performed. As a result, this part of the process inevitably becomes computationally expensive. To avoid this drawback, an ANN is applied to replace the structural analysis process in a GA for structural design optimization. ANNs are recommended because, in structural optimization processes, complete structural responses are not actually necessary. In fact, any parameters that provide the degree of constraint violation will be sufficient. As a result, the network can always be kept simple and small. Consequently, reduced computational time can be expected. In this study, an efficient design of the network is proposed. The efficiency of this proposed network is shown by solving optimization problems of ten-bar truss and one-bay eight-story frame structures. Results obtained from the proposed scheme and the conventional scheme are compared and discussed.

1. Introduction

Recently, genetic algorithms (GAs) have been applied to solve various structural optimization problems. This is because of their advantages of imposing fewer mathematical requirements for solving problems and of being very effective at performing global search. Moreover, GAs are suitable for problems with discrete variables and most structural optimization problems have to deal with discrete design variables. Although GAs have several appropriate characteristics for structural design optimization, one of their best characteristics unfortunately requires long computational time when GAs are used to perform structural optimization. This characteristic is that a GA performs its search from many points in search space at the same time. This will essentially reduce the chance of being trapped in any local optimum points. Nevertheless, in structural design optimization, it also means that analysis of many structures, each of which is in fact one search point, has to be performed in each generation. In addition, to obtain good results, many generations are generally necessary. Due to this reason, the whole GA process will take long computational time. To avoid large computational time due to many structural analyses in structural optimization processes, some researchers employ artificial neural networks (ANNs) to replace these structural analyses [1-3]. The reason why ANNs are used is that computations of ANNs are, in general, lighter than those of structural analysis processes. Rogers [1] proposed guidelines for designing and training a neural network that simulates a structural analysis program and consequently reduces the amount of time taken by an optimization process to converge to an optimum design. These guidelines include the selection of training pairs and determination of the number of nodes on the hidden layer of a three-layer neural network. In his work, nodes

RECEIVED 18 OCTOBER, 2001
ACCEPTED 18 MARCH, 2002

on the input layer represent design variables of the problem being solved while the outputs of network are the objective function as well as constrained variables, such as stresses at several locations on the structure. By using ANNs, an improved strategy for GAs in structural optimization that reduces expensive computation arising from constraint evaluations was also proposed by Jingui et al. [2]. In their work, a three-layer neural network, in which the inputs of the network are design variables and the outputs are constrained variables, is used. Marcelin [3] proposed the use of backpropagation neural networks in creating function approximations for use in design optimization based on GAs. His strategy also consists of substituting, for finite element calculations in the optimization process, an approximate response of a neural network. In his work, the inputs and outputs of the network are also design variables and constrained variables, respectively.

It can be seen from the structures of these existing networks that they are suitable only for small structural analysis problems. In the current practice, the network inputs are usually assigned to be design variables while the outputs are constraints. In structural design, the numbers of design variables and constraints are, in most cases, linearly proportional to the size of the structure being designed. Consequently, if these networks are applied to large problems, they themselves become large systems and will require long computational time. It is therefore very important that the employed network must be designed in such a way that its size does not increase in the same degree as the size of the finite element analysis. In addition, if a large and complicated network is used, not only does computational time of the network become large due to its size but also training of the network will become difficult, if not impossible.

In this study, a GA is used in structural design optimization and, to reduce the computational time of the GA, an ANN is also employed to partially replace the structural analysis process in the GA. The ANN used in this study is designed in such a way that better training and better network prediction can be expected. The design of this network includes forms of inputs and outputs of the network. Special cares are given to the design of the network inputs and outputs to make certain that the size of the obtained network does not depend too much on the size of structures. The proposed ANN will be trained by using results obtained from the finite element analysis. After the training is completed, the network is then used to provide constraint evaluations to the GA instead of the finite element analysis. The network is updated by retraining from time to time to keep satisfactory accuracy. Results obtained from the proposed technique and the conventional GA are compared and discussed. The computational time used by both methods are also compared. All calculations in this study are performed on a Pentium III 733 MHz computer.

2. A GA-Based Structural Optimization Process with an ANN Constraint Evaluator

A GA starts its search from many points in search space at the same time. These starting search points are usually selected at random and known as the initial population. Through the consideration of fitness values of these search points, which are given based on their merit, and the randomized information exchange among the points, a new set of search points with higher merit is created. The process is categorized into three different operators, i.e., reproduction, crossover, and mutation operators [4]. By using these three operators, the process is repeated until a satisfactory result is obtained. One cycle of this repeating process is known as a generation.

When GAs are used in structural design optimization, fitness of each design solution is given based on the value of the objective function and the degree of constraint violation. Generally, an optimization problem using GAs can be expressed as

$$\begin{aligned} &\text{Maximize} \quad F(\mathbf{x}) = F[f(\mathbf{x})] \\ &\qquad\qquad \mathbf{x} = (x_1, x_2, \ldots, x_n) \in R^n \end{aligned} \tag{1}$$

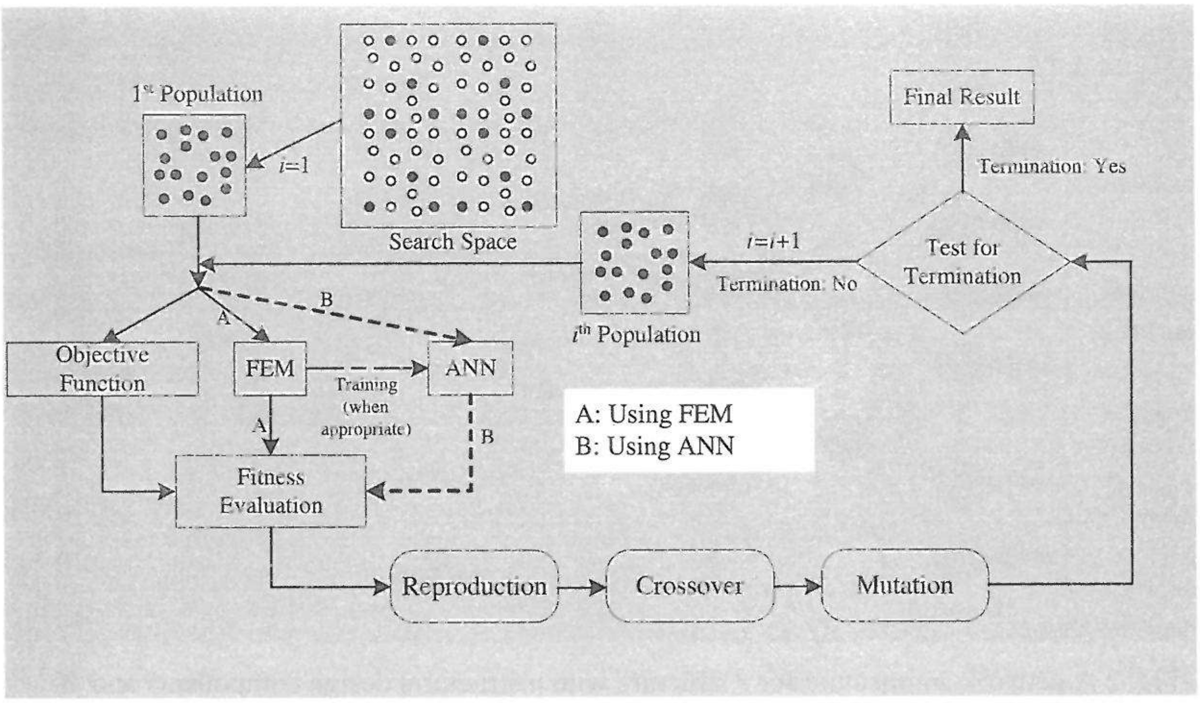


Fig. 1 A GA with an ANN constraint evaluator

under constraints defined as

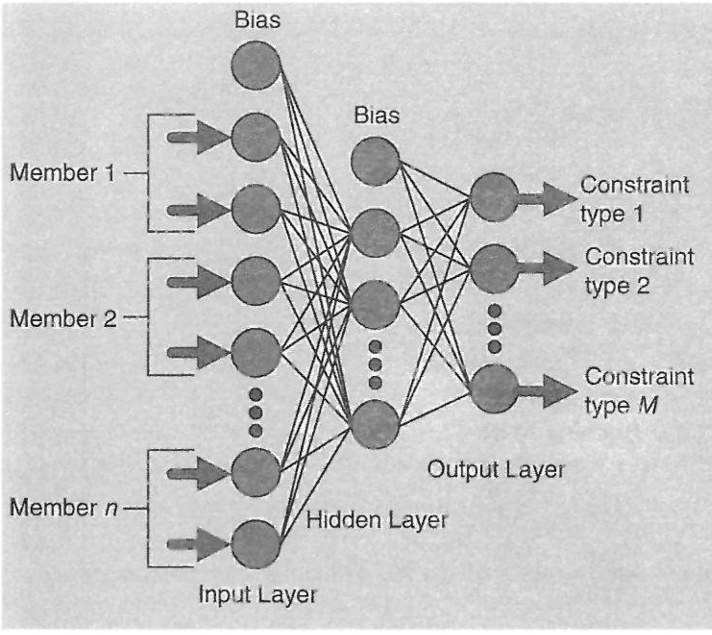
$$g_i(\mathbf{x}) \geq 0, \qquad i = 1,\ldots,K. \tag{2}$$

For structural design optimization, **x** is an n-dimensional vector called the design vector, representing n design variables of the structure being optimized, and $f(\mathbf{x})$ is the objective function. Here, $g_i(\mathbf{x})$'s represent inequality constraints that the design must satisfy, such as stress and displacement limits, and K represents the number of these constraints. Moreover, $F[f(\mathbf{x})]$ is the fitness function that is defined as a figure of merit.

It is not possible to directly utilize GAs to solve the above problem due to the presence of constraints. In GAs, constraints are usually handled by using the concept of penalty functions, which penalize inadmissible solutions, i.e.,

$$\begin{aligned} F^a(\mathbf{x}) &= F(\mathbf{x}) && \text{if } \mathbf{x} \in \tilde{\mathbf{F}} \\ F^a(\mathbf{x}) &= F(\mathbf{x}) - P(\mathbf{x}) && \text{otherwise} \end{aligned} \tag{3}$$

where $\tilde{\mathbf{F}}$ denotes the admissible search space. Here, $P(\mathbf{x})$ is a penalty function whose value is greater than zero. In addition, $F^a(\mathbf{x})$ represents an augmented fitness function after the penalty. Several forms of penalty functions have been proposed in the literature [4-7]. Nevertheless, the basic ideas are the same; i.e., the value of the penalty function depends on the degree of constraint violation. To be able to evaluate $P(\mathbf{x})$, it is therefore necessary to perform structural analyses of all design solutions in the GA population. Since an evolution process needs quite a number of generations to obtain converged results, many structural analyses will be required and they will consume a lot of computational time especially when huge structures are considered. To circumvent this problem, an ANN will be trained to provide the degree of constraint violation instead of the structural analysis. Using ANNs can be beneficial because, in structural optimization processes, complete structural responses are not necessary. Actually, any parameters that provide the degree of constraint violation will be sufficient. As a result, a simple ANN with simple outputs



Sections for each member	Binary coding	Bipolar coding
Type 1	00	(-1)(-1)
Type 2	01	(-1)(1)
Type 3	10	(1)(-1)
Type 4	11	(1)(1)

Fig. 2 A network architecture for a structure with n structural design components and M types of constraint

can be used. Consequently, the time necessary for the simple ANN to provide the necessary prediction will be small, and the total computational time can finally be reduced. In addition, when the size of the structure under consideration increases, the size of the matrix equation used in the structural analysis will increase in a faster fashion than the size of the ANN does. This is because although the structure is larger, the outputs from the ANN can still be kept simple and small since they are not representing complete structural responses. Consequently, when larger problems are solved, the proposed method, which uses an ANN, will even perform better.

Fig. 1 shows the scheme used in the proposed optimization process. The system begins with an ordinary GA. In several early generations, the finite element analysis is used to obtain structural responses and the ANN will not be considered at all. This is because, in these early generations, the GA population changes drastically and it is too difficult to train the network. At an appropriate generation, the training is performed by using results obtained from the finite element analysis. In the subsequent generations, this trained network is used to predict the degree of constraint violation instead of the finite element analysis. After some generations, the network is updated by retraining, and the newly updated network is then used in the subsequent generations. This process is repeated until the end of the calculation. Updating the network at every appropriate interval of generations is necessary because the population keeps changing generation by generation. Consequently, the network trained by older generations may not be able to represent newer generations well.

3. Design of the Artificial Neural Network for Evaluation of Constraint Violation

The most important thing to be considered when an ANN is used to replace the structural analysis in the evaluation of constraint violation is the design of the network itself. With improper design, the performance of the network may drop significantly. In this paper, a simple network of three-layers (an input layer, a hidden layer and an output layer) with a bipolar sigmoid function as an activation function is employed. The learning algorithm is the backpropagation algorithm. In this study, only sizing optimization is considered. Therefore, the inputs of the network are sectional types of the members comprising the structure being fed into the network. Under

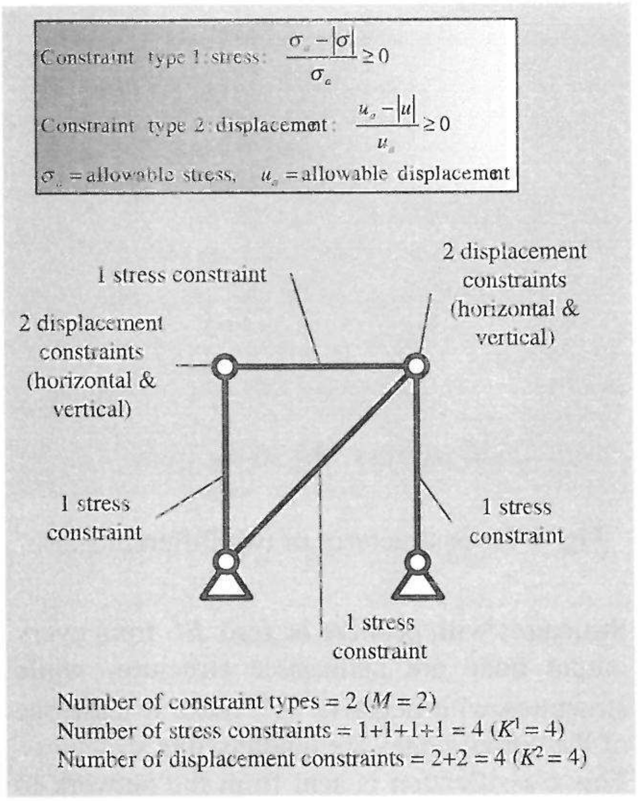

Constraint type 1: stress: $\frac{\sigma_a - |\sigma|}{\sigma_a} \geq 0$
Constraint type 2: displacement: $\frac{u_a - |u|}{u_a} \geq 0$
σ_a = allowable stress, u_a = allowable displacement
1 stress constraint
2 displacement constraints (horizontal & vertical)
2 displacement constraints (horizontal & vertical)
1 stress constraint
1 stress constraint
1 stress constraint
Number of constraint types = 2 (M = 2)
Number of stress constraints = 1+1+1+1 = 4 (K^1 = 4)
Number of displacement constraints = 2+2 = 4 (K^2 = 4)

Fig. 3 Number of constraint types and number of constraints

normal circumstances, it is tempting to directly use sectional properties of the members, such as areas and moments of inertia, as the inputs of the network. Nevertheless, since only sectional types are already sufficient, binary codes of sectional types used in the GA can be used as the inputs. Using the GA codes, not sectional properties, is obviously preferable because the inputs will be discrete and the network can learn much better when its inputs are discrete than when they are continuous. However, it is found that, for this type of problem, the network will perform even better if the inputs are bipolar. As a result, binary codes used in the GA will be changed into bipolar codes before they are used as the inputs of the network. This can be done by simply changing 0 into –1. Fig. 2 shows an example network for a structure with *n* structural members. Each member is assumed to have four types of section to select from; hence, two-bit strings are used for each member. Consequently, two input nodes are required for each member.

As mentioned earlier, it is enough for the network to predict only the degree of constraint violation and the outputs of the network do not have to be complete structural responses. Nevertheless, since the nature of different types of constraint, including the degree of violation, can be different, it is advisable to separate the prediction for each type of constraint, meaning that there should be one output node for each type of constraint. For example, if there are displacement and stress constraints to be considered, two output nodes, one for all the displacement constraints and the other for all the stress constraints, will be used. The network in Fig. 2 shows an example of the network for a structure with *M* types of constraint.

Consider the constraints of an arbitrary type *J*. These constraints can generally be expressed as

$$g_i^J \geq 0 \qquad i = 1, \ldots, K^J \tag{4}$$

where K^J is the number of the constraints of this constraint type *J*. As an example, Fig. 3 shows a truss structure with stress and displacement constraints. The number of constraint types and the number of constraints of each constraint type are shown in the figure.

Next, define the degree of constraint violation for the type-*J* constraints as

$$E^J = \min_{i=1}^{K^J}\left(g_i^J\right) \tag{5}$$

for admissible structures under the type-*J* constraints

$$E^J = \sum_{i=1}^{K^J}\left[\min\left(0, g_i^J\right)\right] \tag{6}$$

for inadmissible structures under the type-*J* constraints

Admissible structures under the type-*J* constraints are structures that do not violate this type of constraint. Note that these admissible structures will have positive or zero E^J which actually means there is no type-*J* constraint violation. On the other hand, inadmissible structures will have negative E^J.

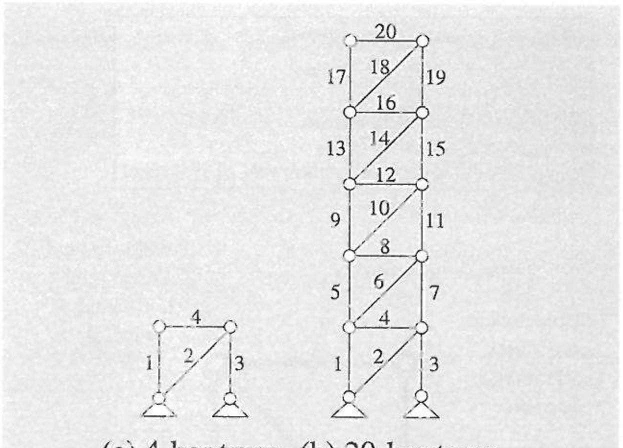
The parameter E^J is designed in such a way that its magnitude shows how much the structure is far away from the boundary between the admissible and inadmissible structures. For admissible structures, E^J is defined as the minimum value of g_i^J's. If a structure is admissible under the type-J constraint, all of its g_i^J's must be positive or at least zero. If only one of them becomes negative, the structure becomes inadmissible with respect to the type-J constraint. What it takes to change an admissible structure into an inadmissible structure is to reduce its minimum g_i^J to a negative value. Hence, the minimum value of g_i^J's is used as E^J for admissible structures. For inadmissible structures, E^J is defined as the summation of all negative g_i^J's. Note that, for an inadmissible structure, at least one of its g_i^J's is negative and there can be some positive or zero g_i^J's. If all negative g_i^J's of the structure are increased to zero, then the structure becomes admissible. Therefore, the summation of all negative g_i^J's can be naturally used as E^J for inadmissible structures.

The degree of constraint violation E^J in (5) and (6) cannot be used directly as the output of the network without normalization. In this study, the normalization is done in such a way that the maximum and minimum values of the outputs during training are 1 and –1, respectively. Therefore, we have

$$\begin{aligned} \tilde{E}^J &= E^J / \left|E^J_{\min}\right| \qquad & E^J < 0 \\ \tilde{E}^J &= E^J / E^J_{\max} \qquad & E^J \geq 0 \end{aligned} \tag{7}$$

where $\tilde{E}^J$ represents the normalized degree of constraint violation. Here, $E^J_{\max}$ and $E^J_{\min}$ denote the maximum and minimum E^J's of all training patterns.

In the optimization by the GA, structures will be categorized into two categories, i.e., admissible and inadmissible structures.



(a) 4-bar truss (b) 20-bar truss

Fig. 4 Truss structures of two different sizes

Structures with positive or zero E^J from every output node are admissible structures while structures with negative E^J from at least one of the output nodes are inadmissible structures. This classification is sent from the network to the GA so that each inadmissible structure can be properly penalized in the calculation of the fitness. In this study, a very simple form of the penalty function is used, i.e.,

$$P(\mathbf{x}) = \lambda \tag{8}$$

where λ is a user-defined constant.

To this point, the numbers of nodes in the input and output layers have been assigned. The number of nodes in the hidden layer is thereafter calculated by using a criterion that, during training, the number of unknown weights of the network should not exceed the number of available equations. In the training of neural networks, the number of equations is equal to the number of training patterns times the number of output nodes. Since the numbers of the input and output nodes of the proposed network are already fixed, the number of the unknown weights depends only on the number of the hidden nodes. Since the number of training patterns is known, by employing the aforementioned criterion, the number of the hidden nodes can easily be determined.

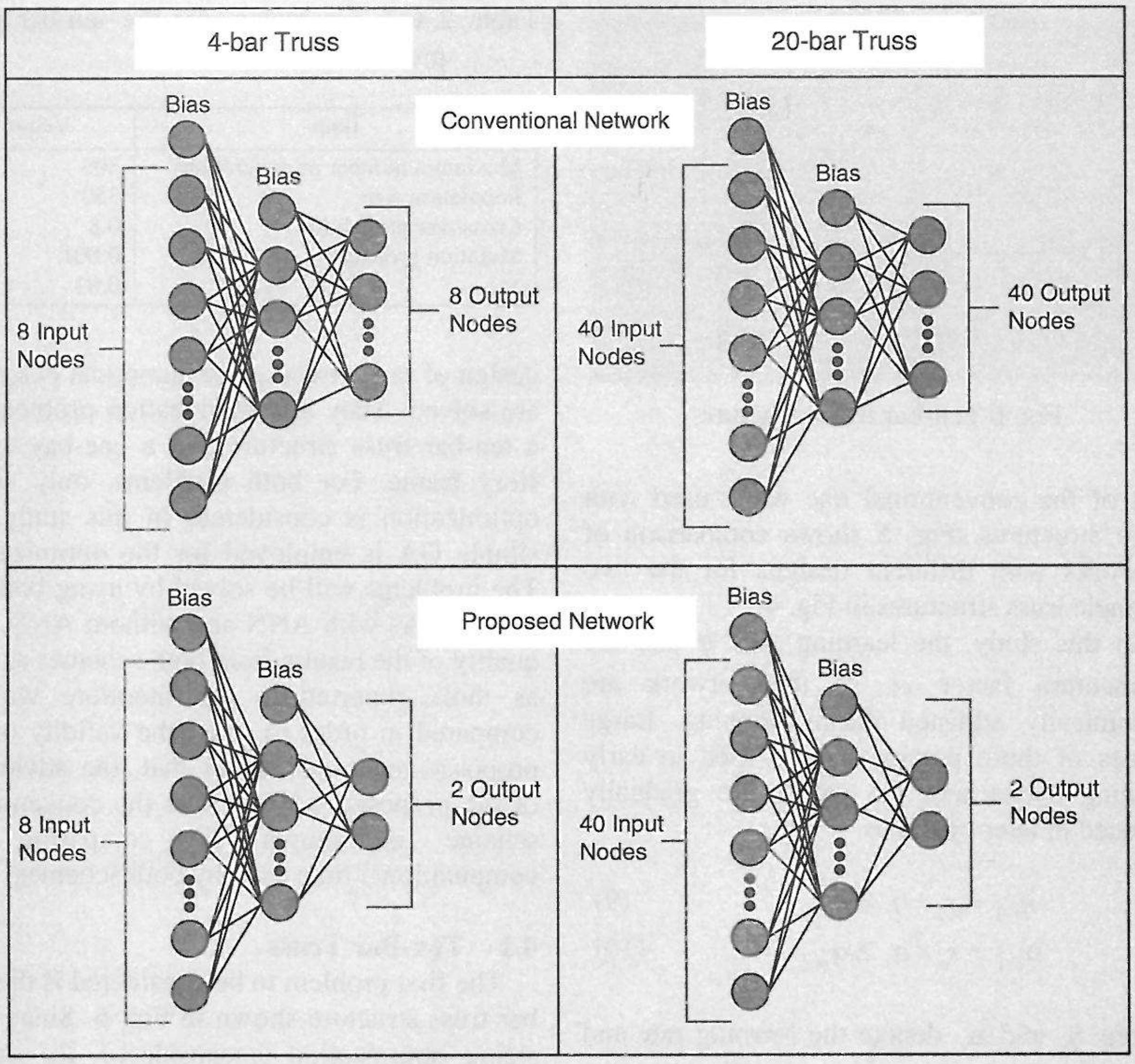


Fig. 5 Networks for the 4-bar and 20-bar truss structures

By keeping the outputs of the network just sufficient and simple, the size of the proposed network can be kept quite small compared with those networks that provide complete structural responses. This difference becomes even more apparent when the size of the structure being considered becomes larger. To illustrate this fact, consider two truss structures of different sizes shown in Fig. 4. Assume that there are four choices of section to be selected for each member of the trusses. Therefore, for bipolar coding of each member, two bits are necessary. As a result, the network for the 4-bar truss requires eight input nodes while the one for the 20-bar truss requires 40 nodes. Assume further that there are two types of constraint, i.e., displacement and stress constraints. For each member, there is one stress constraint and, for each node, there are two displacement constraints, i.e., vertical and horizontal displacements. If the complete responses are used as the outputs of the network, the network for the 4-bar truss will require eight output nodes while the one for the 20-bar truss will require 40 output nodes. However, if the proposed design is used, the number of the output nodes is kept equal to the number of the constraint types regardless of the total number of constraints. As a result, the size of the proposed network will be much smaller than

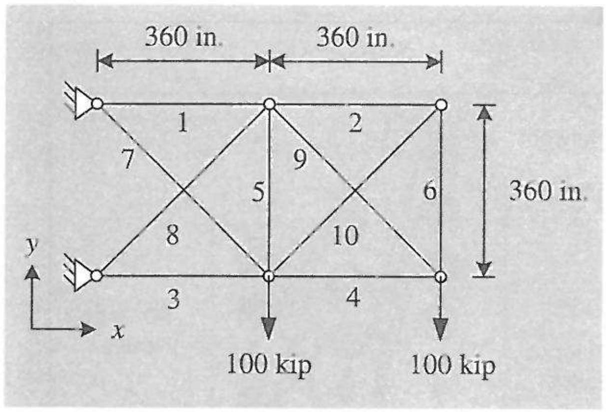


Fig. 6 Ten-bar truss structure

that of the conventional one when used with large structures. Fig. 5 shows comparison of networks with different designs for the two example truss structures in Fig. 4.

In this study, the learning rate η and the momentum factor α of the network are dynamically adjusted during training. Large values of these parameters are used in early training cycles and the values are gradually reduced in later cycles as

$$\eta_{t+1} = r_\eta \times \eta_t \geq \eta_{\min} \quad (9)$$

$$\alpha_{t+1} = r_\alpha \times \alpha_t \geq \alpha_{\min} \quad (10)$$

where η_t and α_t denote the learning rate and the momentum factor of the learning cycle t. Moreover, r_η and r_α are positive factors that are both less than one. In addition, $\eta_{\min}$ and $\alpha_{\min}$ represent the minimum boundaries of the learning rate and the momentum factor, respectively.

4. Results

To illustrate the efficiency of the proposed design of the network, two numerical examples are solved. They are optimization problems of a ten-bar truss structure and a one-bay eight-story frame. For both problems, only sizing optimization is considered. In this study, the simple GA is employed for the optimization. The problems will be solved by using both the simple GAs with ANN and without ANN. The quality of the results from both schemes as well as those reported in the literature will be compared in order to show the validity of the proposed technique. After that, the advantage of the proposed scheme over the conventional scheme is shown by comparing the computational time used by both schemes.

Table 1 Design parameters for the ten-bar truss problem

Item	Value
Modulus of elasticity	10^7 psi
Weight density	0.1 lb/in^3
Allowable tensile stress	25,000 psi
Allowable compressive stress	25,000 psi
Maximum x, y-displacements	2 in.

Table 2 GA parameters for the ten-bar truss problem

Item	Value
Maximum number of generations	300
Population size	350
Crossover probability	0.8
Mutation probability	0.001
λ	0.01

4.1 Ten-Bar Truss

The first problem to be considered is the ten-bar truss structure shown in Fig. 6. Since only sizing optimization is considered, the design variables are ten sectional areas of the ten

Table 3 ANN parameters for the ten-bar truss problem

Item	Value
Number of input nodes	50
Number of hidden nodes	13
Number of output nodes	2
η	0.1
α	0.3
r_n	0.95
r_α	0.95
$\eta_{\min}$	0.01
$\alpha_{\min}$	0.01
Number of learning cycles	300
Generations at which the network is trained	20, 30, 40, 50, 60, 70, 80, 100, 120, 160, 200, 240

Table 4 Comparison of the results for the ten-bar truss problem

Member	Size of member (in^2)					
	Simple GA with ANN (Proposed)	Simple GA without ANN	GA (Nanakorn and Meesomklin [8])	GA (Rajeev and Krishnamoorthy [9])	GA (Galante [10])	GA (Camp et al. [11])
1	30.0	33.5	33.5	33.5	33.5	30.0
2	1.99	1.99	1.62	1.62	1.62	1.62
3	26.5	22.9	22.9	22.0	22.0	26.5
4	13.9	16.9	15.5	15.5	14.2	13.5
5	2.13	1.62	1.62	1.62	1.62	1.62
6	2.38	1.99	1.62	1.62	1.62	1.62
7	11.5	11.5	7.22	14.2	7.97	7.22
8	22.9	18.8	22.9	19.9	22.9	22.9
9	19.9	22.9	22.0	19.9	22.0	22.0
10	1.80	1.80	1.62	2.62	1.62	1.62
Total weight (lb)	5624.6	5640.5	5499.3	5613.8	5458.3	5556.9

members of the truss. The cross-sectional areas of members 1, 3, 4, 7, 8 and 9 are taken from the following 32 discrete values, i.e., 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, and 33.5 in^2. For the rest of the members, the cross-sectional areas are taken from the following 32 discrete values, i.e., 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, and 14.2 in^2. A five-bit string is required for each design variable because there are, for each member, $2^5=32$ choices of sections to be selected. There are two types of constraint in this problem, i.e., the stress and displacement constraints. The design parameters used in the problem are shown in Table 1 and the GA parameters are shown in Table 2. In this problem, the fitness function $F(\mathbf{x})$ is defined as

$$F(\mathbf{x}) = \frac{1}{1+Weight(\mathbf{x})} \tag{11}$$

where weight in pound is used in the equation.

Since there are ten structural members to be optimized and each member requires a five-bit string, the number of the input nodes is equal to 50. In addition, since there are two types of constraint, two output nodes are used. The network parameters are shown in Table 3. For this problem, the network is first trained at the 20th generation and it is updated by retraining at the generations listed in Table 3.

The results of the simple GA with ANN are compared with those of the simple GA without ANN and of the literure as shown in Table 4. It can be seen that the results from the proposed method and the simple GA without ANN are comparable. This means that using the ANN to replace the finite element analysis in the GA is acceptable regarding the quality of the GA results. Note that the results of these two cases are the best results of 40 different runs. Although the result of the proposed scheme is

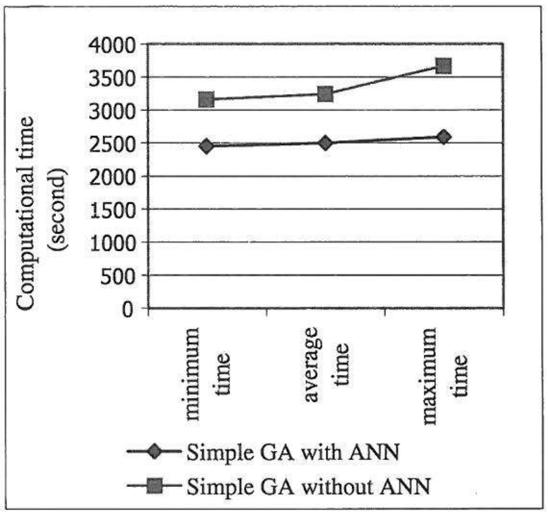


Fig. 7 Computational time for the ten-bar truss problem

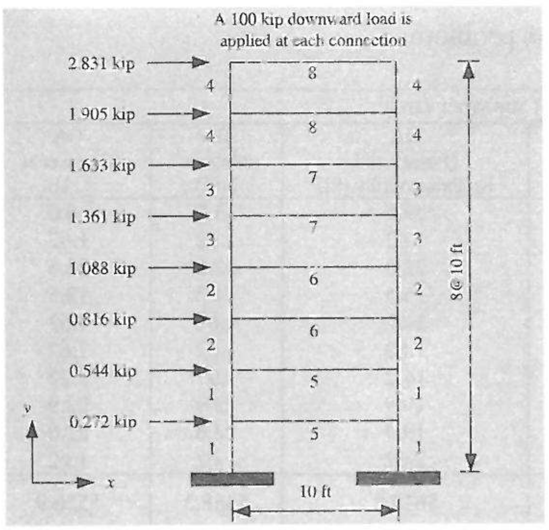


Fig. 8 One-bay eight-story frame

not as good as those reported in the literature, it must be noted that in this study only the simple GA is employed. The results from the literature shown in Table 4 are obtained from more sophisticated GAs. Lastly, the computation time of the simple GA with ANN and the simple GA without ANN is shown in Fig. 7. The maximum, minimum and average time is obtained from the computational time of the 40 runs. It can be clearly seen that the proposed scheme employs much less time than the conventional scheme. On average, more than 20% reduction of computational time is observed.

4.2 One-Bay Eight-Story Frame

The next problem considered here is the one-bay eight-story frame structure shown in Fig. 8. The 24 members of the structure are categorized into eight groups (see Fig. 8). In this problem, 256 sections are selected from a list of 268 W-sections from the American Institute of Steel Construction Allowable Stress Design (AISC-ASD) [12] by discarding the 12 biggest sections from the list. Thus, an eight-bit string is required for each design variable. There is only one displacement constraint in the problem that is the maximum *x*-displacement at the top of the structure. The design parameters used in the problem are shown in Table 5 while the GA parameters are shown in Table 6. The fitness function used in this problem is the same as the one used in the previous problem (see Eq. 11) except that the unit used in the equation for this problem is kip instead of pound.

Table 5 Design parameters for the one-bay eight-story frame problem

Item	Value
Modulus of elasticity	$29\text{x}10^3$ ksi
Weight density	$2.83\text{x}10^{-4}$ kip/in^3
Maximum *x*-displacement at the top of the structure	2 in.

Table 6 GA parameters for the one-bay eight-story frame problem

Item	Value
Maximum number of generations	200
Population size	70
Crossover probability	0.85
Mutation probability	0.05
λ	10

In this problem, there are 24 members to be optimized. Nevertheless, they are categorized into only eight groups. Each group needs an eight-bit string to represent. Consequently, the number of the input nodes is equal to 64. As there is only one type of constraint, one output node is used. The network parameters are shown in Table 7. Similar to the previous problem, the network is first trained at the 20th

Table 7 ANN parameters for the one-bay eight-story frame problem

Item	Value
Number of input nodes	64
Number of hidden nodes	1
Number of output nodes	1
η	0.1
α	0.3
r_η	0.95
r_α	0.95
$\eta_{\min}$	0.01
$\alpha_{\min}$	0.01
Number of learning cycles	300
Generations at which the network is trained	20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100, 110, 120, 140, 160, 180

Table 8 Comparison of the results for the one-bay eight-story frame problem

Group Number	Simple GA with ANN (Proposed)	Simple GA without ANN	GA (Nanakorn and Meesomklin [8])	GA (Camp et al. [11])	Optimality criteria (Camp et al. [11])
1	W 18 x 60	W 21 x 50	W 12 x 45	W 18 x 46	W 14 x 34
2	W 16 x 26	W 12 x 26	W 14 x 34	W 16 x 31	W 10 x 39
3	W 18 x 35	W 21 x 44	W 12 x 35	W 16 x 26	W 10 x 33
4	W 12 x 16	W 10 x 19	W 10 x 19	W 12 x 16	W 8 x 18
5	W 21 x 50	W 18 x 46	W 18 x 35	W 18 x 35	W 21 x 68
6	W 12 x 26	W 21 x 44	W 18 x 40	W 18 x 35	W 24 x 55
7	W 18 x 35	W 12 x 22	W 16 x 36	W 18 x 35	W 21 x 50
8	W 18 x 40	W 14 x 53	W 16 x 26	W 16 x 26	W 12 x 40
Total weight (kip)	8.49	8.86	7.47	7.38	9.22

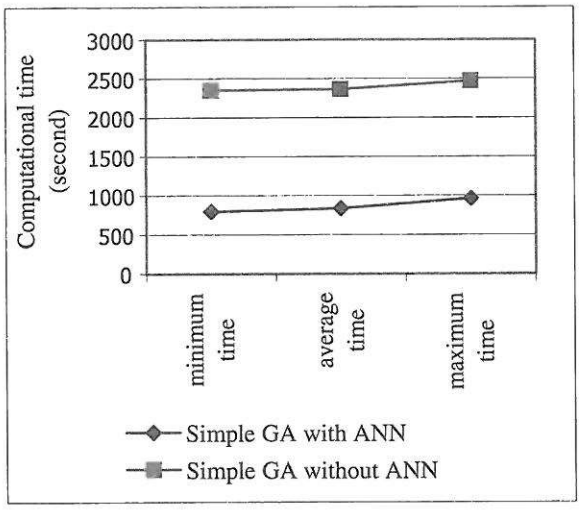


Fig. 9 Computational time for the one-bay eight-story frame problem

generation and it is updated at the generations listed in Table 7.

Similar to the previous problem, the results of the simple GA with ANN are compared with those of the simple GA without ANN as well as those of the literature. The comparison is shown in Table 8. Again, the results from the proposed method and the simple GA without ANN are comparable. Note that the results of these two cases are also the best results of 40 different runs. It is observed that the result of the proposed scheme may not be as good as some of the results from the literature. However, those results from the literature are obtained from much more sophisticated GAs.

Finally, the computational time of the simple GA with ANN and the simple GA without ANN is compared in Fig. 9. It can be clearly seen that the proposed scheme employs impressively much less time than the conventional scheme. More than 60% reduction of computational time is observed on average. It must be noted that, compared with the previous problem, much higher computational time reduction is achieved in this problem. This is because the size of the structure in the current problem is much larger than that of the previous problem. As a result, the analysis time required by the finite element analysis increases significantly. However, since the outputs of the network are kept simple and do not depend on the size of the structure, the size of the neural network itself increases to a much lesser degree than the finite element analysis does. As a result, the computational time reduction in this problem naturally becomes much more apparent than that in the previous example. Therefore, it is evident that the proposed method will be even more favorable when used in larger problems.

5. Conclusions

In this study, an artificial neural network is applied to replace the structural analysis process in the simple genetic algorithm for structural design optimization. The objective of the proposed scheme is to reduce the time required by structural analyses in the optimization process. To achieve this goal, the network used is designed to provide only the degree of constraint violation, instead of the complete structural responses, to the optimizer.

As a result, the form of the network is simple, and small, which subsequently leads to small computational time. The proposed network is used with the simple GA to solve optimization problems of truss and frame structures. The results from the proposed scheme and the simple GA without ANN are comparable. This means that using ANNs, instead of the finite element analysis, to evaluate the degree of constraint violation in GAs is acceptable regarding the quality of GA results. As for the computational time, it can be seen from the results that the proposed scheme employs much less time than the conventional GA without ANN. Moreover, it can be observed that the time reduction is higher if the proposed scheme is used in larger structural problems since the size of the network itself increases to a much lesser degree than the finite element analysis does when the size of the structure increases. As a result, higher benefit is expected from the proposed scheme when it is used with larger problems.

References

[1] J.L. Rogers, "Simulating Structural Analysis with Neural Network", Journal of Computing in Civil Engineering, Vol. 8, No. 2, 1994, pp. 252-265.

[2] L. Jingui, D. Yunliang, W. Bin and X. Shide, "An Improved Strategy for GAs in Structural Optimization", Computers & Structures, Vol. 61, No. 6, 1996, pp. 1185-1191.

[3] J.L. Marcelin, "Evolutionary Optimisation of Mechanical Structures: Towards an Integrated Optimisation", Engineering with Computers, Vol. 15, 1999, pp. 326-333.

[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[5] K. Deb, Optimization for Engineering Design: Algorithms and Examples, Prentice-Hall, 1995.

[6] J.L. Marcelin, P. Trompette and R. Dornberger, "Optimization of Composite Beam Structures using a Genetic Algorithm", Structural Optimization, Vol. 9, 1995, pp. 236-244.

[7] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1996.

[8] P. Nanakorn and K. Meesomklin, "An Adaptive Penalty Function in Genetic Algorithms for Structural Design Optimization", Computers and Structures, Vol. 79, No. 29-30, 2001, pp. 2527-2539.

[9] 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.

[10] M. Galante, "Genetic Algorithms as an Approach to Optimize Real-World Trusses", International Journal for Numerical Methods in Engineering, Vol. 39, 1996, pp. 361-382.

[11] C. Camp, S. Pezeshk and G. Cao, "Optimized Design of Two-Dimensional Structures Using a Genetic Algorithm", Journal of Structural Engineering, Vol. 124, No. 5, 1998, pp. 551-559.

[12] T. Burns, Structural Steel Design–LRFD, Delmar Publishers, 1995.