Genetic Algorithms for Optimization of 3D Truss Structures

Vedat Toğan and Ayşe Turhan Daloğlu

Abstract Various optimization techniques have been applied to find the optimum solutions of structural design problems in the last 50 or 60 years. Simple structural optimization problems with continuous design variables have been solved initially using mathematically diverse techniques. New approaches called meta-heuristic techniques have been emerging along with the progress of traditional methods. This chapter first introduces the mathematical formulations of optimization problems and then gives a summary and development process of the preliminary techniques such as genetic algorithm (GA) in obtaining the optimum solutions. The mathematical formulations of the structural optimization problems are associated with the design variables, loads, structural responses, and constraints. Strategies are proposed to improve the performance of the technique to reduce the number of search and the size of the problem. Finally, some examples related to 3D truss structures are presented.

1 Introduction

Optimization of the structures is one of the main research areas in civil and structural engineering. As a branch of applied and computational mathematics, optimization usually tries to find the best-fitted solution of the problem within a domain that contains acceptable values of design variables subject to some design restrictions or constraints. The optimum solution of the problem may be achieved by minimizing or maximizing a real objective function satisfying predefined restrictions at the same time. Such a solution is supposed to be the best one among a large feasible solution space that can satisfy all the constraints of the optimization problem. The function to be minimized or maximized is referred to as objective

V. Toğan · A.T. Daloğlu (✉)
Department of Civil Engineering, Karadeniz Technical University,
61080 Trabzon, Turkey
e-mail: aysed@ktu.edu.tr

X.-S. Yang et al. (eds.), *Metaheuristics and Optimization in Civil Engineering*,
Modeling and Optimization in Science and Technologies 7,
DOI 10.1007/978-3-319-26245-1_6

function, while the functions to represent restrictions are called as constraints. The design parameters of an optimization problem are called design variables. While the geometric properties of members can be considered as design variables in structural weight optimization, coordinates of nodal points can be treated as design variables for shape or topology design of structures. So the objective function, constraints, and variables can vary widely according to the type of the problem. For the minimum weight design of 3D truss structures, the objective function represents the weight of the truss, variables can be the areas of cross sections of the structural members, and the constraints may be the maximum allowable stresses and displacements of nodal joints.

2 Mathematical Formulations of Optimization Problems

An optimization problem subject to some constraints can be formulated as the following mathematical form:

$$\begin{aligned} &\min_{\mathbf{x}\in S} f(\mathbf{x}) \\ &S = \left\{ \mathbf{x} | h_j(\mathbf{x}) = 0, j = 1,\ldots,p;\;\; g_k(\mathbf{x}) \leq 0,\, k = 1,\ldots,m \right\} \end{aligned} \tag{1}$$

where $f(\mathbf{x})$, $h_j(\mathbf{x})$, and $g_k(\mathbf{x})$ are a C^1 function of $\mathbf{x} \in IR^n$. In addition, p and m are the total numbers of $h_j(\mathbf{x})$ and $g_k(\mathbf{x})$, respectively. S represents the feasible set for the optimization problem, $S \subset IR^n$.

However, in the engineering field, an optimization problem can also be generally defined as follows:

$$\begin{aligned} &\text{find} \quad \mathbf{x} = \{x_1, x_2, \ldots, x_n\} \\ &\text{min.} \quad W(\mathbf{x}) \\ &\text{s.t.} \quad g_k(\mathbf{x}) \leq 0, \quad k = 1 \text{ to } m \\ &\qquad x_{il} \leq x_i \leq x_{iu}, \quad i = 1 \text{ to } n \end{aligned} \tag{2}$$

In Eqs. (1) and (2), $\mathbf{x}$ is an n-dimensional vector representing the design variables of the optimization problem. Depending on the optimization type, the cross-sectional areas of the members, the nodal coordinates of the member connections, and the members itself are treated as the design variables. $f(\mathbf{x})$ and $W(\mathbf{x})$ are called objective functions or cost functions, which usually correspond to a real number to be used to evaluate how good a solution is. Since weight is usually adopted as the objective function of the optimization problem in structural engineering, $W(\mathbf{x})$ corresponds to the weight of the related structure. $h_j(\mathbf{x})$ and $g_k(\mathbf{x})$ are the equality and inequality constraints functions, respectively. In contrast, $h_j(\mathbf{x})$ in the engineering fields, the type of inequality constraint functions, $g_{\mathrm{k}}(\mathbf{x})$, are often encountered. And it generally requires a structural analysis to obtain the structural response such as displacements, forces, etc. Here, x_{il} and x_{iu} show the lower and upper boundaries of x_i. Strictly speaking, x_i might be continuous, discrete, and real but since it is preferred in practice to select the cross-sectional areas of members from the predefined list, x_i is taken as the discrete in structural engineering applications.

3 Genetic Algorithms

To find the value of **x** obeying the $h_j(\mathbf{x})$ and $g_k(\mathbf{x})$ and minimizing $W(\mathbf{x})$ requires an optimization method to be employed. Optimization methods, generally speaking, are classified as the gradient-based and gradient-free techniques. Of course, various categorizations can be encountered for the optimization methods available in the literature [1–3]. In fact, although they are referred with various classifications and names, there is not much difference in the main properties of the methods. For example, one of the most commonly used classifications is deterministic and stochastic. While the former uses derivatives of the objective function and constraints in the search of the optimum solution, the latter works with probabilistic transition rules instead of the gradient information of the objective function and constraints [4–12]. As seen from the example, deterministic and stochastic techniques are easily put into the classification of the gradient-based and the gradient-free optimization methods, respectively.

Genetic algorithm (GA) is probably one of the first optimization methods, which simulates the natural phenomena into a numerical algorithm. It mimics the procedure known as survival of the fittest. GA was firstly proposed by Holland [13]. Since new some valuable improvements in the GA such as adaptive operators, distinct coding schemes for the design variables, immigration, elitism, breeding, hybridization, etc., have been presented by the researcher [19–30].

After the study of Rajeev and Krishnamoorthy [14] which was a well-documented study for the application of the GA in the structural optimization problems, GA has gained more popularity in this area than that proposed for the first time. By studying Rajeev and Krishnamoorthy [14], it is realized at first glance that GA is very primitive compared with the level recently reached. For instance, the design variables were coded with binary scheme in the GA process proposed by Rajeev and Krishnamoorthy [14], while at the moment the design variables are treated as discrete and even that mixed [15–18]. Figure 1 demonstrates the binary scheme for the design variables of the structural optimization problems. Herein, it is assumed that the corresponding optimization problem has three design variables

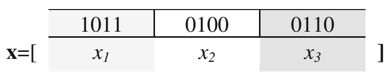
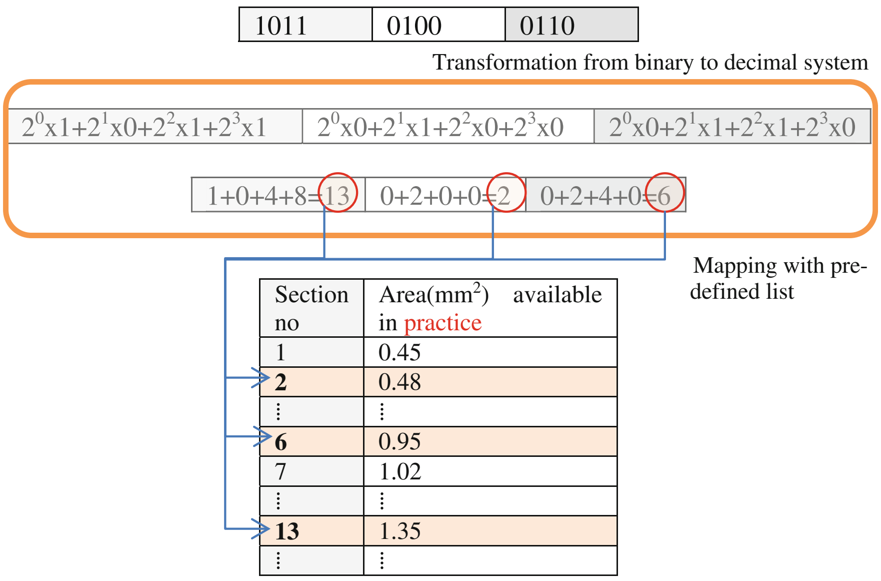


Fig. 1 Coding in binary system for design variables

which are the cross-section areas of the grouped members. And they are selected from 16 different sections available in practice.

When this scheme is chosen to find the value of the design variables in decimal system, a transformation is needed. Through this transformation, the sequence numbers of the sections related to the design variables are determined from a predefined list sections. The steps of the transformation and mapping can be summarized as the main steps shown in Fig. 2. After mapping with the predefined list, the cross-section area for the related design variable is directly used in the corresponding process, i.e., in calculating the structural responses via a structural analyzer.

Then the structure should be analyzed to determine its responses and then the requirements of the optimization problems defined by the constraints functions are evaluated. The next step is to calculate the value of $W(\mathbf{x})$. As mentioned before, it shows the goodness of the solution that consists of coupling the design variables (see Fig. 1)—the string is called as a solution. However, since the GA is an unconstrained optimization method like other gradient-free optimization techniques, $W(\mathbf{x})$ also includes the value of a function known as the penalty function which reflects the violation level of the constraints in normalized form for the solution. Equation (3) shows the objective function, incorporating the constraints violation as expressed in Rajeev and Krishnamoorthy [14].


1011
0100
0110
Transformation from binary to decimal system
2^0x1+2^1x0+2^2x1+2^3x1
2^0x0+2^1x1+2^2x0+2^3x0
2^0x0+2^1x1+2^2x1+2^3x0
1+0+4+8=13
0+2+0+0=2
0+2+4+0=6
Mapping with pre-defined list

Section no	Area(mm^2) available in practice
1	0.45
2	0.48
⋮	⋮
6	0.95
7	1.02
⋮	⋮
13	1.35
⋮	⋮

Fig. 2 Decoding and mapping steps for the design variables represented in the binary system

$$W(\mathbf{x}) = W'(\mathbf{x})\,(1 + KC) \\ \text{where } C = \sum_{k=1}^{m} c_k \tag{3}$$

where K is a parameter that is taken as 10, C is the violation coefficient computed in the following manner: if $g_k(\mathbf{x} > 0$, then $c_k = g_k(\mathbf{x})$; or if $g_k(\mathbf{x} \leq 0)$, then $c_k = 0$. Finally, $W'(\mathbf{x})$ represents the weight of the structure. Note that $g_k(.)$ is in the normalized form expressed as $\sigma/\sigma_a - 1 \leq 0$ for stress and $d/d_a - 1 \leq 0$ for displacement. σ_a and d_a in these expressions show the allowable values for stress and displacement for the problem considered.

3.1 Genetic Operators

The search procedures of GA were based on the mechanics of natural genetics and natural selection. With the help of the genetic operators adapted from nature, the concept of the survival of the fittest is simulated to form a robust search mechanism. Following the steps summarized above, the GA search procedure proposed by Rajeev and Krishnamoorthy [14] applies two genetic operators successively.

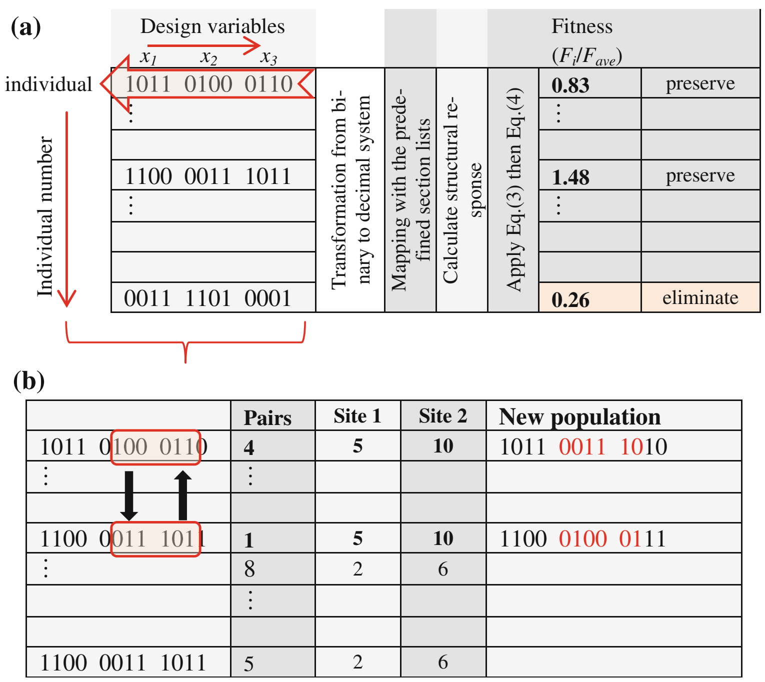
The first one is the reproduction operator which reflects the concept of the survival of the fittest in nature. It proceeds according to the individual fitness calculated as follows:

$$F_i = (W_{\max} + W_{\min}) - W_i(x) \tag{4}$$

where F_i is the fitness of the ith individual in the population, $W_{\max}$ and $W_{\min}$ are the maximum and minimum values of $W(\mathbf{x})$ computed using Eq. (3). Thus, the individuals with higher fitness values have a higher probability to survive, whereas the less fit ones get fewer chances of survival. And the worst fit individuals will be removed from the population.

Then, to exchange the solution segments between the individuals in the population, crossover operator is implemented. Double point crossover is applied to the pairs selected randomly as an example. Figure 3 demonstrates the application of the two genetic operators explained in Rajeev and Krishnamoorthy [14].

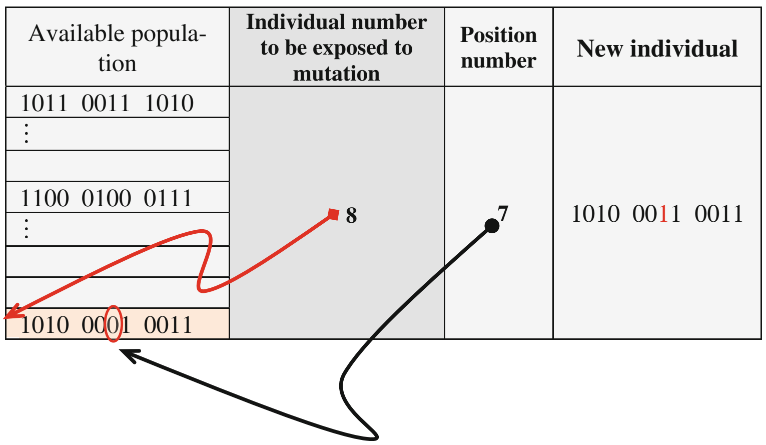
The genetic algorithm continues the process by following the above steps outlined in detail so as to modify the new population. To terminate the GA process, a criterion based on the similarity of the individuals in the population was imposed by Rajeev and Krishnamoorthy [14]. In addition, although mutation operator was not implemented in [14], the GA search process employed it to preserve the diversity among the population. Figure 4 illustrates the basic concept behind the mutation operator. It proceeds in three steps. First, an individual within the population is randomly selected. Then, a binary position to be changed is determined randomly.



Design variables x_1 x_2 x_3	Transformation from binary to decimal system	Mapping with the predefined section lists	Calculate structural response	Apply Eq.(3) then Eq.(4)	Fitness (F_i/F_{ave})	
1011 0100 0110					**0.83**	preserve
⋮					⋮	
1100 0011 1011					**1.48**	preserve
⋮					⋮	
0011 1101 0001					**0.26**	eliminate

	Pairs	**Site 1**	**Site 2**	**New population**
1011 0100 0110	**4**	**5**	**10**	1011 0011 1010
⋮	⋮			
1100 0011 1011	**1**	**5**	**10**	1100 0100 0111
⋮	8	2	6	
	⋮			
1100 0011 1011	5	2	6	

Fig. 3 Two genetic operators given in Rajeev and Krishnamoorthy [14]. **a** Reproduction operator. **b** Crossover operator



Available population	**Individual number to be exposed to mutation**	**Position number**	**New individual**
1011 0011 1010			
⋮			
1100 0100 0111	**8**	**7**	1010 0011 0011
⋮			
1010 0001 0011			

Fig. 4 Mutation operator

After that, the corresponding position value is switched to 1 if it is equal to 0 or to 0 if it is equal to 1.

All search procedures based on GA were called simple genetic algorithms in Rajeev and Krishnamoorthy [14] and since then, numerous modifications have been proposed by researchers in order to improve the search and/or computational performances of the GA in this area. For example, the coding scheme [19–21], the penalization [22–27], and new operators [28–30] are some of them. Other improvements based on adaptive concept have attracted more attention. The following section will review this concept.

4 Strategies Based on the Adaptive Concept

Some improvement or renewal in the GA operators or the GA algorithms have been made by researchers to increase the probability of finding the global solution and to enhance the performance of GA. Other improvements of GA are to relieve the user from the burden of determining sensitive parameter(s) existed in GA. The vast majorities of these efforts have been focused on the adaptive approaches in GA for both the penalty function, and the mutation and crossover. The key idea behind the adaptive approaches is to adjust itself automatically during the optimization procedure using genetic algorithms.

4.1 *Adaptive Penalty Scheme*

Although it is not a genetic operation, the penalty function is important in GA to demonstrate the extent of the violation of the constraints quantitatively. Penalty techniques can be classified as multiplicative or additive. A positive penalty factor is introduced in the multiplicative case to amplify the value of the fitness function of an infeasible individual in a minimization problem. This type of penalty has received less attention in the evolutionary computation community, compared with the additive type. A penalty functional is added to the objective function in the additive case to define the fitness value of an infeasible element [24].

An adaptive penalty scheme to be able to adjust itself automatically during the GA process is proposed by Toğan and Daloğlu [31] as follows:

$$\begin{aligned} f_{\text{penalty}} &= (C_{\max} + C(r))/(C_{\max} + C_{\text{ave}}) && C(r) \geq C_{\text{ave}} \\ f_{\text{penalty}} &= (C_{\text{ave}} + C(r))/(C_{\text{ave}} - C_{\min}) && C(r) < C_{\text{ave}} \\ f_{\text{penalty}} &= 0 && C(r) = 0 \qquad r = 1, \ldots, \text{nps} \end{aligned} \tag{5}$$

where f_{penalty} is the penalty function, $C(r)$ is the violation value of normalized constraints of the rth individual in the generation, and nps represents the population size. In addition, $C_{\max}$, $C_{\min}$, and C_{ave} are, respectively, the maximum, minimum and average violation values of generation. According to this formulation, Eq. (3) becomes

$$W(\mathbf{x}) = W'(\mathbf{x})\left(1 + f_{\text{penalty}}\right). \tag{6}$$

The penalty function is, therefore, kept free from any predefined or user-defined constants. Also the magnitudes of the violations are not characterized by a static rate for both near feasible and infeasible solutions during the design process. With the expressions in Eq. (5), the infeasible solutions will not be penalized with the same rate of penalty. The magnitude of the penalty tends to get heavier instead, as the level of the violation of the infeasible solution tends to get bigger. Moreover, the magnitude of penalty increases as the violation value gets closer to $C_{\max.}$ On the other hand, it decreases as the violation value gets closer to $C_{\min.}$ Thus, some infeasible individuals that are close to the feasible region in the search space will not disappear through the penalty scheme and they will find a chance to survive. This may sustain the capacity of finding the global solution for design problem using GA.

4.2 Adaptive Crossover and Mutation Schemes

Genetic operators are applied to mimic the natural evolution. Among these operators, crossover provides the genetic information exchange between the couples randomly, and mutation enables the development of new genetic material, and both play an effective role to reach the optimum or to get close to the optimum solution. It is arbitrary and up to the user to incorporate the rates of these operators in optimization process. The choices of mutation, p_m, and crossover, p_c, rates as well as generating positions to be shifted by mutation and mapping the individuals for crossover are also arbitrary.

Many refinements using adaptive controls provide significant improvements in performance for some situations [32]. Keeping all of these in mind, it can be concluded that the randomness on mapping for crossover, and specifying the gene position(s) for mutation may be removed. Mutation and crossover should be adapted for both the individual and the generation because it is possible to lose the best-fitted individual with this random process in mutation. Therefore, in contrast to traditional crossover and mutation operator based on randomization mechanisms, i.e., generating the pairs, and determining position of bit shifted by mutation of the solution, the mutation and the crossover operators can be adaptive and adjust themselves from generation to generation since the population is renewed from iteration to iteration. Adaptive means adjusting itself automatically depending on the fitness value of the individual and the other individuals in the generation.

The adaptive mutation and crossover operators suggested by Srivinas and Patnaik [33] are modified as follows and applied by Toğan and Daloğlu [31, 34–36], for the optimization of 2d and 3d truss structures:

$$p_m = \left\langle \begin{array}{ll} 0.5(W_{\max} - W(r))/(W_{\max} - W_{\text{ave}}) & W(r) \geq W_{\text{ave}} \\ (W_{\text{ave}} - W(r))/(W_{\text{ave}} - W_{\min}) & W(r) < W_{\text{ave}} \end{array} \right. \tag{7a}$$

$$p_c = \left\langle \begin{array}{ll} (W_{\max} - W^*)/(W_{\max} - W_{\text{ave}}) & W^* \geq W_{\text{ave}} \\ 1 & W^* < W_{\text{ave}} \end{array} \right. \tag{7b}$$

Here, *W(r)* is the fitness of the *r*th individual, W_{ave} is the average fitness value of the population, $W_{\max}$ and $W_{\min}$ are the maximum and minimum fitness values of an individual in the population, respectively, and W^* is the larger of the fitness values of the solutions to be crossed. p_m and p_c are mutation and crossover rates, respectively.

After the mutation rate, p_m, is determined using Eq. (7a), the numbers of design variables, m_{des}, disrupted by mutation are calculated by multiplying p_m with the string length of the solution. Then design variables in the individual are arranged according to the level of violation of normalized constraints, and they are renewed with m_{des} starting with the most violated one. Thus design variables in the individuals are classified and the good individuals are kept unchanged. Also the diversity of population is maintained since the design variables that violate the constraints are renewed.

Unlike Srivinas and Patnaik [33], and Bekiroğlu [37], W^* represents the lower value of the two fitness values of the solutions for crossover. The reason for that is because if the lower value of fitness is bigger than W_{ave}, the crossover will take place between the pairs having good fitness value, whereas when W^* represents the higher of the fitness values, there is a possibility for crossover to take place between the pairs having bad fitness values. Since the adaptive crossover is incorporated, information exchange between pairs can be done with various crossover points changing from 1 to string length of the individual (flexible point crossover). The numbers of design variables, c_{des}, exchanged by crossover between pairs are specified by multiplying p_c with string length for the solution. So if p_c is equal to 1, the individual of pairs will not be subjected to crossover operator.

5 Innovative Approaches in Genetic Algorithms

The enhancements developed by the researchers to increase the performance of GA are not limited to adaptive schemes. Besides, some refinements are proposed for making the search procedures of GA computationally effective. Member grouping and initial population strategies are some of them as described in the next subsections in detail.

5.1 Initial Population Strategy

To start the evolution process in GA, the initial population within the solution space is generated. All individuals constituting the initial population are selected from the solution space of the problem notwithstanding any condition. In other words, this step is random. However, even though the process of creating initial population seems ordinary, it critically affects the convergence, the performance and the ability of the GA. This case becomes more crucial for very complicated and large solution space which is mostly encountered in the practical application of GA in the area of structural engineering.

The idea starting the search of the solution space without a randomly generated set is the key rationale of creating the initial population automatically. So, adopting the list number of the maximum area of cross sections as the starting point for the design variables leads to more efficiency than randomly generated. And it is stored as the initial point for each group of tension members to create the initial population. For the groups of the compression members, two or three surplus of the list number of the member that has he maximum area of cross section in the group is taken from the list of sections. The value of cross-sectional area and radius of gyration of that section must be bigger than the values found previously. And the corresponding section list number gives the initial points for each group of compression members and is stored to create the initial population (see Toğan and Daloğlu [31, 34] for more details).

5.2 Member Grouping Strategies

In the structural optimization terminologies, the design variables refer to the variables affecting the value of the objective function, and they generally represent the areas of the cross section of the structural members. Since the GA completely independent from the characteristic of the problem, the design variables of the optimization problems should be coded in some encoding schemes such as binary, decimal, real, and so on. GA evolves those solutions by creating in terms of design variables and randomly selecting into the potential solutions space of the problem. A set of possible feasible or unfeasible solutions construct the population or generation. Each solution in the populations is known as the individual.

For a given problem, all of the cross-sectional areas of the structural members can be taken as design variables. In this case, however, the computation time gets very high and the results obtained from optimization process will probably be the local optima due to the expanded design space. Therefore in the GA applications, member grouping is generally applied for the members of the structural system in order to reduce the size of the problem. On the other hand, the member grouping adopted a priori might not lead to an accurate grouping and if the number of members of the structural systems becomes very large; i.e., for 3D roof trusses, transmission towers, this leads to very large string lengths, which delays convergence and precludes useful exchange of information [38].

Two new member grouping strategies are proposed by Toğan and Daloğlu [31, 34] to reduce the size of the search space of the design problem as much as possible, to increase the probability of catching the global optimum solution and enhance the performance of the GA. The key idea behind the member grouping strategies is to make a convenient member grouping that will end up with as few numbers of cross sections as possible in the final set, and reduce the size of the design space of the problem as much as possible. The efforts are also made to relieve the user from the burden of determining the member groups.

5.2.1 First Member Grouping Strategy

The first strategy (strategy 1) is based on the one proposed by previous researchers [31, 38, 39]. To implement this strategy, the same cross-section areas are assigned for all the structural members first as stated in [31, 38, 39]. Then the analysis of the structure is performed using these initial areas for each load cases. Following the static analysis, the entire range of internal forces is divided into several ranges both for tension and compression members. And members are grouped according to the internal forces in the members.

An additional group is added in Toğan and Daloğlu [34] to the system for zero force members or members with very low internal forces. Hence, all the members of the truss structure are grouped conveniently and accurately. Moreover, a complex solution space may be avoided under some conditions.

5.2.2 Second Member Grouping Strategy

For strategy 2, while the magnitude of the axial force is considered as the factor for grouping the tension members [31, 38, 39], slenderness ratio is considered as the main factor for the compression members to set the groups. Therefore, due to the importance of slenderness ratio, it may be more convenient to group the compression members according to their slenderness ratio in terms of radius of gyration of the cross section and the effective length of the member instead of grouping them depending on the magnitude of the axial force. This is the key idea behind strategy 2. Hence, as the tension members of the truss structure are grouped depending on the axial forces, the compression members are grouped according to their slenderness ratio. An extra member group for the zero force members or members with very low internal forces is also arranged (Toğan and Daloğlu [31, 34]).

6 Examples

In this section, in the light of the information given in the previous sections, two types of design examples are presented. In the first one, a 160-bar space truss is considered as an example to demonstrate the effectiveness and robustness of proposed adaptive approaches for GA and member grouping strategy over simple GA. Later, an investigation is performed to demonstrate the efficiency, accuracy, and reliability of the proposed initial population strategies by solving numerical examples taken from previous studies in the literature for comparison.

The population size is taken as 40 for all the examples and real-value coding is employed in the genetic algorithm.

At the beginning of the genetic process 40 % of the initial population is created by using the proposed initial population strategy automatically. Therefore the diversity of the population is preserved and the algorithm may be less likely to get stuck at local minima and may also avoid some early convergence. It is possible to create all the individuals of the initial population automatically. However, in this case, the initial population consists of the same individual only and the search performed in the solution space start in a certain region. On the other hand, as the adaptive schemes applied for both penalty functions and mutation and crossover operators are able to adjust itself automatically during the genetic process [31], they completely disrupt the initial population. So, creating all the individuals in the initial population automatically is not meaningful.

6.1 Example 1: 160-Bar Truss Tower

The 160-bar truss tower shown in Fig. 5 was optimized by Rajaev and Krishnamoorthy [14] and Galante [40] in advance. 32 cross-section types were used to optimize the tower and taken from the AISC Manual [41]. The members are classified into 16 groups. Details of the member groups were presented in Galante [40]. Rajaev and Krishnamoorthy [14] used in GA with one criterion (minimum weight) and without taking the buckling effect into account. As Galante [40] stated, it can be observed that the buckling effect plays an important role in truss optimization. So if it is not taken into account the truss obtained will not be suitable as load carrying structural system. Galante [40] optimized the transmission tower with the aim of the minimum weight and minimum number of cross-section types of bars taken from the market and also taking buckling and the slenderness limits recommended by AISC [41] into account.

The tower is optimized for the objective indicated by Galante [40] with the proposed algorithm. All parameters needed to start the optimization process are taken from the reference studies. The structure is optimized with the implemented improvements in GA and two member grouping strategies for the aim of that the final design forms three groups at one for tension and two for compression

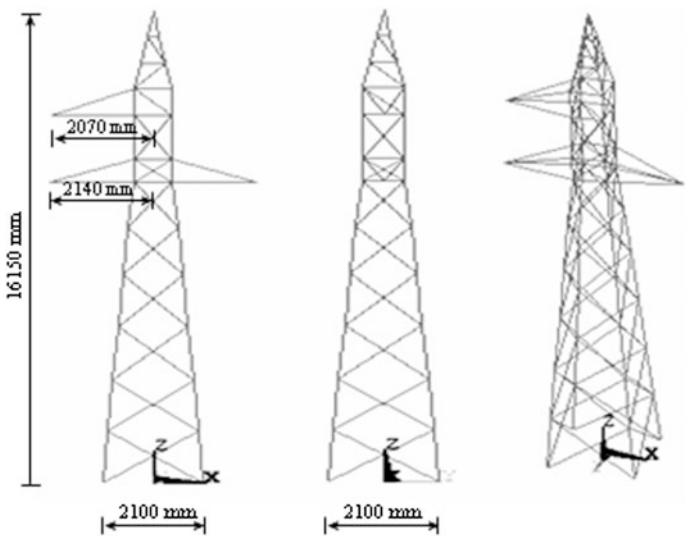


Fig. 5 160-bar trussed steel transmission tower

members. An extra member group for zero force members is not formed. This is because of the fact that Galante [40] reported that 51 bars have a higher compression stress than the limit recommended by AISC [41] to prevent buckling and 68 have higher slenderness ratio than the AISC advice. It is also observed from the optimization process of the tower that the arrangement of an extra member group for zero force members does not make any difference in the weight of the tower drastically and the two member grouping strategies give the same result. Therefore, the tower is optimized with three, four, and five member groups.

The final design obtained by the proposed strategies and the ones reported by Galante [40] are presented in Table 1. Galante [40] used both GA and the simple rebirth process in GA for the optimization of this example and mentioned that the GA with the rebirth process achieves a better truss. However, the optimum designs obtained with the proposed improvements in the algorithm ended up with a lighter truss than the design by Galante [40]. A question may arise: as the values of design variables presented in Table 1 for the three groups are the same, why are the optimum weight of truss different? The answer is hidden in Fig. 6a. It is shown in Fig. 6a that some members that belong to the first group skipped to the second group and a better solution from the previous one is reached. However, the most interesting result is obtained when the four member groups are adopted at two for tension and two for compression member for this example. Although this optimum design has more groups than the result obtained by Galante [40] and previously performed optimization cases, in this study, it seems still reasonable to achieve that the number of sections in the final set must be as few as possible to make fabrication and workmanship easier [31, 39, 40]. The reduction in the total weight of the

Table 1 Comparison of results for 160-bar transmission tower

Design variables (mm^2)	Galante [40] at 3 groups by the GA	Galante [40] at 3 groups by the GA with rebirth	This study			
			At 3 groups	At 3 groups	At 4 groups	At 5 groups
A_1		2812	767.74	767.74	767.74	767.74
A_2		3064	581.93	581.93	339.99	339.99
A_3		7096	1251.61	1251.61	581.93	581.93
A_4					1251.61	1251.61
A_5						150.96
Weight (kN)	15.33	14.61	14.269	12.651	10.544	10.449

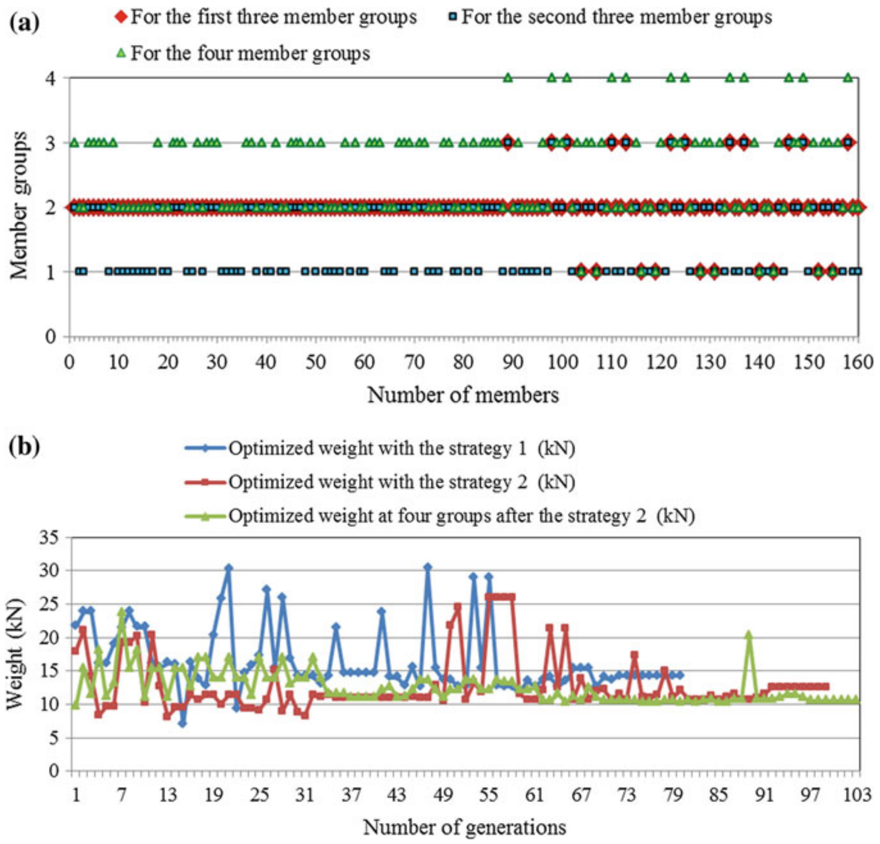


Fig. 6 **a** Member groups for 160-bar tower, **b** Variation of weight for the three cases with number of generations

tower is about 28 % compared to the designs summarized in Table 1. As mentioned above, when the number of groups is taken as five, the optimized weight is slightly changed. In other words, arranging an extra group for the zero force members did not affect the weight of the tower drastically. Hence, the design at four groups is more meaningful than the design at five groups for the design objective. Figure 6b shows the genetic histories of the value of the objective function taken as the weight of tower for three optimization cases.

6.2 Example 2: 640-Bar Space Deck Truss

A trussed space deck shown in Fig. 7 has 179 joints and 640 members. The truss was first studied by Jenkins [32] in order to assess the decimal GA on a more substantial structural problem. Jenkins [32] found the optimum height of truss in addition to optimum volume. The truss members were subjected to compressive stress limitation given in BS 5950 and tensile stress limitation, 275 N/mm^2. This trussed space deck was subjected to the one loading condition, which a single load of 300 kN was applied at the center of the upper plane (joint 90). A maximum displacement limitation of ±40 mm was imposed on every node in vertical direction. 21 discrete values of data for each design variable were taken from rectangular hollow steel sections with cross-sectional areas varying from 142 mm^2 in intervals to 4350 mm^2. Jenkins [32] collected the members of the structure in five distinct groups as follows: (1) upper deck longitudinal members, (2) upper deck transverse members, (3) lower deck longitudinal members, (4) lower deck transverse members, and (5) diagonal members connecting the upper and lower planes.

For this example, 23 pipe sections given in AISC [41] are adopted for each design variables. The allowable tensile stress is taken as 275 N/mm^2 and the modulus of elasticity is 210 kN/mm^2. The allowable compressive stress is calculated according to AISC [41] for the compression members and the maximum deflection imposed is 40 mm. The truss is subjected to one loading condition as specified in [32]. The optimization of the space truss is first carried out by using five groups imposed by Jenkins [32]. Then the optimal volume of the truss is obtained with four groups of members that were assumed as one for tension, two for compression members, and one for zero force members after preliminary analysis. Table 2 shows the optimum volume for each trial and the maximum deflection of the truss. The algorithm proposed in this study achieved a design with the best solution vector after approximately 29,000 searches for each trial. When four groups for the members are adopted after the preliminary analysis, the volume of the truss gets smaller than the result obtained by using five groups imposed by Jenkins [32]. Moreover, nearly 50 % reduction is obtained with both the member grouping and the new initial population strategies adopted in this study. This design obtained with the proposed algorithm confirms the intension of “both the weight of the structure and the number of cross section should be minimized to obtain an economical structure” as indicated in [31, 33–36, 38–40].

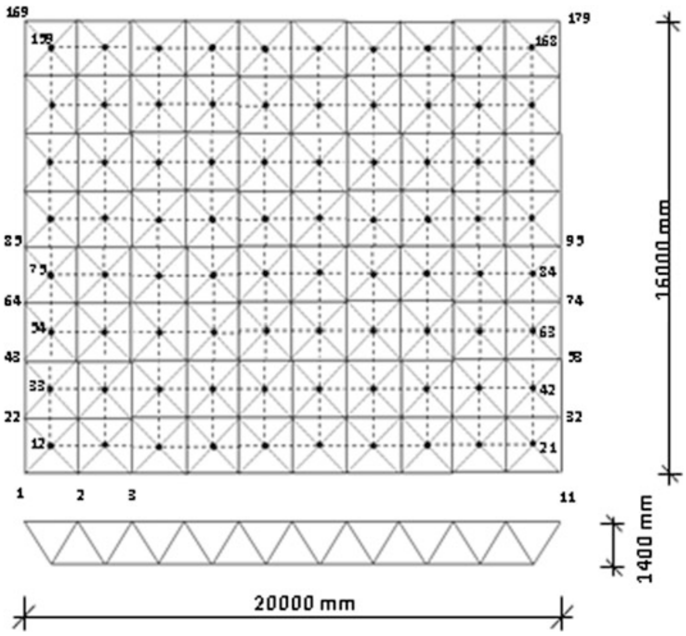


Fig. 7 640 trussed space deck truss

Table 2 Results for the 640-bar space deck truss

Variables	Optimal cross-section areas (mm^2)	
	Case 1	Case 2
A_1	1096.77	568.39
A_2	954.84	690.32
A_3	954.84	1096.77
A_4	690.32	161.29
A_5	1096.77	
Volume (cm^3)	1298026.52	659970.80
Max. def. (mm)	29.03	39.85
Note The coded values design variables for the automatically created individuals		
For case 1		
Coded values	Volume (cm^3)	Violation (stresses + displacements)
12 10 9 8 10	912520	10.503
For case 2		
Coded values	Volume (cm^3)	Violation (stresses + displacements)
9 9 12 1	602088.6	47.84

6.3 Example 3: 1512-Bar Shed Truss

In order to demonstrate the performance, efficiency, and practical capability of the algorithm with the proposed strategies, the design of 1512-bar shed truss is studied as a final design problem. A shed structure illustrated in Fig. 8 covers a square field with size of 40,000 × 40,000 mm^2 and has a height of 5359 mm. The structure is designed in a way that its sub-parts are reproduced along direction Z per 5000 mm and are connected by lateral elements on the lower arch level in that direction. In a sub-part each node on lower arch connects to four nodes on upper arches with diagonals. So, it consists of 409 nodes and 1512 truss elements. The structure is supported at two edges in the Z direction. The material density and modulus of elasticity are 7.85 × 10–8 kN/mm^3 and 210 kN/mm^2, respectively. It is subjected to a load scheme that is applied to all nodes of upper arches. 35,000 N is applied the nodes along Z direction and it increases 10,000 N per each node level in Y direction so that at the two nodes near symmetry axis the value is 55000 N (see Fig. 8). The allowable value of 250 N/mm^2 is employed for tensile stresses and the formulation of buckling obeying AISC [41] considered for compressive stresses. The algorithm is provided with 26 discrete values of data for each design variable. The structural properties are taken from the pipe sections as given in [41]. The maximum displacement limitation imposed is 50 mm.

As seen in Fig. 8, since the shed truss is a large structural system with more than 1000 members, it might be very difficult to optimize the area of individual members if the member groups become very large. Besides, very large string length causes the solution space of the problem to increase and becomes very complicated. Therefore, member grouping adopted to reduce the size of the problem and initial points to search the solution space are crucial to get an optimum design which is closer the global optimum.

The truss is designed by adopting the four groups at one for tension, two for compression, and one for zero members after the preliminary analysis. Table 3 gives the best solution vectors and the corresponding weight. An optimal structural

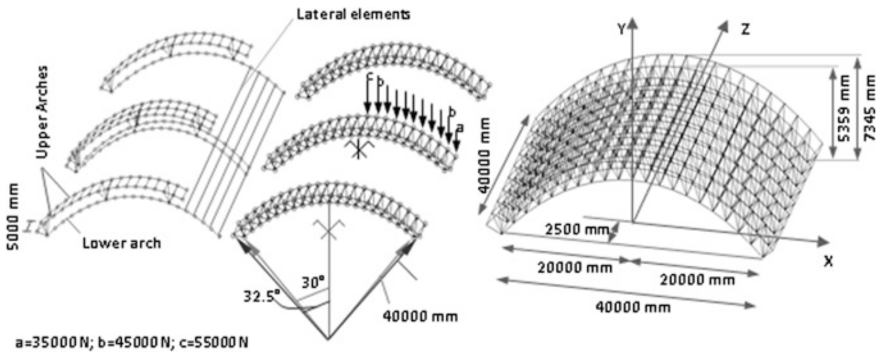


Fig. 8 1512-bar shed truss

Table 3 Result for the 1512-bar shed truss

Variables	Optimal cross-section areas (mm^2)	
A_1	690.32	
A_2	2045.16	
A_3	5419.34	
A_4	515.48	
Weight (kN)	692.43	
Max. def. (mm)	49.31	
Note The coded values design variables for the automatically created individuals		
Coded values	Weight (kN)	Violation (stress + displacement)
10 17 23 8	871.78	8.893

weight of 692.43 kN was obtained considering the constraints, which gives a feasibility to construct it in practice. The algorithm obtained the optimum solution after approximately 43,600 searches.

7 Conclusion

An optimum design approach is proposed based on the GA for truss structures with the help of self-adaptive strategies for member grouping, penalty function, mutation and crossover operators, and the initial population, which is then employed for the optimization of large 3D truss structures. An investigation has been performed to demonstrate the performance and workability of the enhanced GA (eGA). It is shown from the design examples that the eGA works well for the large structural systems. It is also worth pointing out that self-adaptive strategies in eGA help user to start the GA process automatically. It can be expected that proposed eGA may also be a useful search technique and a tool for solving discrete sizing variables of the large 3D truss structures.

References

1. Horst, R., Pardolos, P.M.: Handbook of global optimization. Kluwer Academic Publishers, Dordrecht (1995)
2. Nocedal, J., Wright, J.S.: Numerical optimization. Springer, New York (1999)
3. Chong, E.K.P., Zak, S.H.: Introduction to Optimization. Wiley, New York (2002)
4. Paton, R.: Computing with Biological Metaphors. Chapman & Hall, London (1994)
5. Adami, C.: An Introduction to Artificial Life. Springer, New York (1998)
6. Matheck, C.: Design in Nature: Learning from Trees. Springer, Berlin (1998)
7. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, Cambridge (1998)
8. Flake, G.W.: The Computational Beauty of Nature. MIT Press, Cambridge (2000)

9. Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
10. Glover, F., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht (2003)
11. Dreo, J., Petrowski, A., Siarry, P., Taillard, E.: Meta-Heuristics for Hard Optimization. Springer, Berlin (2006)
12. Sean, L.: Essentials of Metaheuristics (2015). http://cs.gmu.edu/∼sean/book/metaheuristics/Essentials.pdf
13. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co., Reading (1989)
14. Rajeev, S., Krishnamoorthy, C.S.: Discrete Optimization of Structures Using Genetic Algorithms. J. Struct. Eng. **118**(5), 1233–1250 (1992)
15. Tang, X.,·Bassir, D.H., Zhang, W.: Shape, Sizing Optimization and Material Selection Based on Mixed Variables and Genetic Algorithm. Optim Eng **12**, 111–128 (2011)
16. Ahmadi, M., Arabi, M., Hoag, D.L., Engel, B.A.: A mixed discrete-continuous variable multiobjective genetic algorithm for targeted implementation of nonpoint source pollution control practices. Water Resour. Res. **49**, 8344–8356 (2013)
17. Yuan, Q.K., Li, S.J., Jiang, L.L., Tang, W.Y.: A mixed-coding genetic algorithm and its application on gear reducer optimization. Fuzzy Info. Eng. **2**(AISC 62), 753–759 (2009)
18. Rao, S.S., Xiong, T.: A hybrid genetic algorithm for mixed-discrete design optimization. J. Mech. Des. **127**(6), 1100–1112 (2004)
19. Kumar, A.: Encoding scheme in genetic algorithm. Int. J. Adv. Res. IT Eng. **2**(3), 1–7 (2013)
20. Kumar R, Jyotishree (2012) Novel encoding scheme in genetic algorithms for better fitness. Int. J. Eng. Adv. Tech. **1**(6), 214–218
21. Zhu, J., Zhou, D., Li, F., Fu, T.: Improved real coded genetic algorithm and its simulation. J. Softw. **9**(2), 389–397 (2014)
22. Nanakorn, P., Meesomklin, K.: An adaptive function in genetic algorithms for structural design optimization. Comp. Struct. **79**(29–30), 2527–2539 (2001)
23. Kramer, O., Schwefel, H.P.: On three new approaches to handle constraints within evolution strategies. Nat. Comp. **5**, 363–385 (2006)
24. Lemonge, A.C.C., Barbosa, H.J.C.: An adaptive penalty scheme for genetic algorithms in structural optimization. Int. J. Numer. Meth. Eng. **59**, 703–736 (2004)
25. Coello, C.A.C.: Use of a self-adaptive penalty approach for engineering optimization problems. Comp. Ind. **41**, 113–127 (2000)
26. Lin, C.H.: A rough penalty genetic algorithm for constrained optimization. Inform. Sci. **241**, 119–137 (2013)
27. Lemonge, A.C.C., Barbosa, H.J.C., Bernardino, H.S.: A family of adaptive penalty schemes for steady-state genetic algorithms. Proceeding in WCCI 2012, June, pp. 10–15. Brisbane, Australia (2012)
28. Kaya, M.: The effects of two new crossover operators on genetic algorithm performance. Appl. Soft Comput. **11**, 881–890 (2011)
29. Thanh, P.D., Binh H.T.T., Lam, B.T.: New mechanism of combination crossover operators in genetic algorithm for solving the traveling salesman problem. Knowl. Syst. Eng. (AISC 326), 753–759 (2015)
30. Deep, K., Thakur, M.: A new mutation operator for real coded genetic algorithms. Appl. Math. Comput. **193**(1), 211–230 (2007)
31. Toğan, V., Daloğlu, A.T.: Optimization of 3d trusses with adaptive approach in genetic algorithms. Eng. Struct. **28**, 1019–1027 (2006)
32. Jenkins, W.M.: A decimal-coded evolutionary algorithm for constrained optimization. Comput. Struct. **80**(5–6), 471–480 (2002)
33. Srivinas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994)
34. Toğan, V., Daloğlu, A.: An improved genetic algorithm with initial population and selfadaptive member grouping. Comput. Struct. **86**, 1204–1218 (2008)

35. Toğan, V., Daloğlu, A.: Adaptive approaches in genetic algorithms to catch the global optimum. Proceeding in ACE 2006, October, pp. 11–13. İstanbul, Turkey (2006)
36. Toğan, V., Daloğlu, A.: optimization of truss systems with metaheuristic algorithms and automatically member grouping. Proceeding in 4th National Steel Structures Symposium, October, pp. 24–26. İstanbul, Turkey (2011)
37. Bekiroğlu, S.: Optimum design of steel frame with genetic algorithm (in Turkish). M.Sc. thesis, Karadeniz Technical University (2003)
38. Krishnamoorthy, C.S., Venkatesh, P.P., Sudarshan, R.: Object-oriented framework for genetic algorithms with application to space truss optimization. J. Comput. Civil Eng. **16**, 66–75 (2002)
39. Sudarshan, R.: Genetic algorithms and application to the optimization of space trusses. A Project Report, Madras (India), Indian Institute of Technology (2000)
40. Galante, M.: Genetic algorithms as an approach to optimize real-world trusses. Int. J. Numer. Meth. Eng. **39**, 361–382 (1996)
41. American Institute of Steel Construction (AISC).: Manual of steel construction-allowable stress design, 9th edn. Chicago (1989)