Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2018; 8(4):625-656

SIZE AND GEOMETRY OPTIMIZATION OF TRUSS STRUCTURES USING THE COMBINATION OF DNA COMPUTING ALGORITHM AND GENERALIZED CONVEX APPROXIMATION METHOD

P. Darvishi and S. Shojaee[*, †]
Department of Civil Engineering, Shahid Bahonar University, Kerman, Iran

ABSTRACT

In recent years, the optimization of truss structures has been considered due to their several applications and their simple structure and rapid analysis. DNA computing algorithm is a non-gradient-based method derived from numerical modeling of DNA-based computing performance by new computers with DNA memory known as molecular computers. DNA computing algorithm works based on collective intelligence. It works with doing random search in the search space and creating the initial random population by modeling DNA-based computing operators and applies the operators derived from genetic algorithm to achieve the optimum solution of the objective function. Generalized Convex Approximation (GCA) method is a gradient-based method that with approximation of the main function and starting from a point, finds the optimum solution using information about functions and their gradient. In this research, in order to minimize the weight of truss, the cross-section areas of the elements as discrete variables are optimized by DNA computing algorithm, and the coordinates of truss nodes as continuous variables are optimized by Generalized Convex Approximation (GCA) method. Therefore, to simultaneously optimize the size and geometry of truss structures, these two methods are used in combination. The results of numerical examples show the proper functioning of this process.

Keywords: Optimization; Truss; DNA computing algorithm; Generalized Convex Approximation (GCA) method.

Received: 28 November 2017; Accepted: 12 February 2018

[*]Corresponding author: Department of civil engineering, Shahid Bahonar University, Kerman, Iran
[†]E-mail address: saeed.shojaee@uk.ac.ir (S. Shojaee)

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

1. INTRODUCTION

Nowadays, optimization plays an important role in solving many engineering problems such as structural engineering. So, the growth and development of computer science, and also the importance of achieving optimum solution in short time have caused the researchers welcome using meta-heuristic algorithms with random search in solving optimization problems. The meta-heuristic algorithms such as genetic algorithm, ant colony algorithm, particle swarm optimization algorithm and simulated annealing algorithm are the most attractive. These algorithms do not need common computations such as computing gradient of functions. So, they are more rapid than ordinary algorithms in solving optimization problems.

Also, because of random search in the search space during the process of solving optimization problems, the exploration and exploitation performance of these algorithms rise and therefore, the chance of achieving global optimum solution increases.

DNA-based computing is a new research method. In this method, the computation is performed based on DNA (Deoxyribonucleic acid) molecules and using new computers with DNA memory called molecular computers. At first, DNA-based computing was proposed by Adelman in 1994 to solve travelling salesman problem [1]. By successful solving of the travelling salesman problem, the ability to do DNA-based computing in solving NP (Non Polynomial) problems was distinguished. This method was used to solve other problems such as vertex coloring problem [2], the minimum spanning tree problem [3], the knapsack problem [4] and the N-Queens Problem [5].

In DNA-based computing, by taking advantage of high volume data storage feature in molecular computers, DNA strings are produced. Then, by applying operators in parallel on all strings, DNA strings are purified. By evaluating the remained strings, the computational problem will be solved [6].

DNA computing algorithm is a meta-heuristic algorithm and a non-gradient-based method resulted from numerical modeling of DNA-based computing performance by molecular computers that works based on collective intelligence, doing random search in search space, creating initial random population by modeling DNA-based computing operators and applying operators derived from genetic algorithm to achieve the optimum solution for objective function [7].

Generalized Convex Approximation (GCA) method is a step-by-step mathematical programming method that converges to the optimum solution by producing convex approximations of optimization problem called sub-problem and solving these sub-problems [8].

In this research, in order to minimize the weight of truss structures, the cross-section areas of the elements as discrete variables are optimized by DNA computing algorithm (size optimization), and the coordinates of truss nodes as continuous variables are optimized by Generalized Convex Approximation (GCA) method (geometry optimization).

Therefore, to simultaneously optimize the size and geometry of truss structures, these two methods are used in combination and the results will be discussed in the numerical examples.

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

2. THE PROBLEM OF SIZE AND GEOMETRY OPTIMIZATION OF TRUSS STRUCTURES

The general form of an optimization problem is formulated as Equation (1):

$$\begin{aligned} &Minimize: f(X) \\ &Subject\ to: \frac{g_i(X)}{\bar{g}_i(X)} - 1 \le 0 \qquad i = 1,2,\ldots,m \\ &X = \{x_1, x_2, \ldots, x_j, \ldots, x_n\} \end{aligned} \tag{1}$$

where, $f(X)$ is the objective function, X is the set of design variables, m is the number of constrains, n is the number of variables and $\bar{g}_i(X)$ is the allowable value of n-th constraint [9].

In size and geometry optimization of truss structures, the goal is minimizing the weight of truss structures under design constraints. In size optimization, the cross-section areas of the elements are selected as discrete variables, and in geometry optimization, the coordinates of truss nodes are selected as continuous variables [10].

The general form of optimization problem of truss structures is formulated as Equation (2) with selecting the truss weight as objective function, and selecting constraints such as axial stresses of the elements, displacements in the degrees of freedom of truss nodes and slenderness ratios of the elements:

$$\begin{aligned} &Minimize: W(X) = \sum_{i=1}^{n_e} \gamma_i A_i L_i \\ &Subject\ to: \frac{|\sigma_i(X)|}{\bar{\sigma}_i} - 1 \le 0 \qquad i = 1,2,\ldots,n_e \\ &\frac{|\Delta_j(X)|}{\bar{\Delta}_j} - 1 \le 0 \qquad j = 1,2,\ldots,n_j \\ &\frac{\lambda_i(X)}{\bar{\lambda}_i} - 1 \le 0 \qquad i = 1,2,\ldots,n_e \\ &x_j^{\min} \le x_j \le x_j^{\max} \\ &A_i \in S \end{aligned} \tag{2}$$

where X is the set of design variables, $W(X)$ is the truss weight as objective function, n_e is the number of truss elements, n_j is the total number of the degrees of freedom of truss nodes, γ_i is the specific weight of i-th element, A_i is the cross-section area of i-th element, L_i is the length of i-th element, $\sigma_i(X)$ is the stress of i-th element, $\bar{\sigma}_i$ is the allowable stress of i-th element, $\Delta_j(X)$ is the displacement in the j-th degree of freedom, $\bar{\Delta}_j$ is the allowable displacement in the j-th degree of freedom, $\lambda_i(X)$ is the slenderness ratio of i-th element, $\bar{\lambda}_i$ is the allowable slenderness ratio of i-th element, x_j is the coordinate of the j-th degree of freedom, x_j^{min} and x_j^{max} are respectively the lower limit and upper limit of the j-th degree of freedom and S is the cross-section list of elements [11].

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

3. DNA-BASED COMPUTING BY MOLECULAR COMPUTERS

DNA molecule is made up of two protein strings wrapped and called polynucleotide. These strings are made of ingredients known as organic base called nucleotide. Nucleotides are divided into four groups: Adenin, Guanin, Cytosine and Thymine which are respectively abbreviated with the letters A, G, C and T. In each DNA molecule by using the complementarity feature of bases A with T and C with G and hydrogen bond between complementary bases, polynucleotide strings bond together.

DNA-based computing is a new research method to solve computational problems which computation is done based on DNA molecule and using new computers with DNA memory called molecular computers [1]. Molecular computers are the next generation computers. They use DNA molecules as storage memory of information [12]. DNA molecules can store more information than current computers memory. Therefore, molecular computers store more information than silicone computers. Also, DNA molecules provide the energy of molecular computers in such a way that by breaking the bonds between DNA strings, the released energy is used in molecular computers [13].

DNA-based computing was first used by Adelman in 1994 to solve travelling salesman problem. Then this method was used to solve NP problems [14]. In solving a computational problem by DNA-based computing, trillions DNA strings with bases A, G, C and T are produced as the solutions which is due to high capacity of information storage. These strings are placed in test tubes. Then by applying specific and biochemical operators in parallel on all strings, DNA strings are purified and the needed strings remain. After that, by evaluating the remained strings as desired solutions to the problem, the final solution of the computational problem is achieved.

The operators used in DNA-based computing are:

(1) Synthesis operator that produces DNA strings based on bases A, G, C and T with a specific length. It is possible to assign these strings as sub-string to design variables of problem.

(2) Ligation operator with salt and water and ligase pastes the sub-strings related to the problem variables together and some strings are produced as the computational problem solutions.

(3) PCR[1] operator amplifies the desired complementary strings. In this way, according to Fig. 1, if the double-string molecule of XYZ↕ is available, the temperature is increased to 95 celsius degree and the two strings of DNA molecule by melting process are converted to single strings of XYZ↑ and XYZ↓. Then, the initial signals of the beginning and end of the considered string that are Z↓ and X↑ are added to combine with molecules and to achieve YZ↑, X↕, Z↕ and XY↓. After that, polymerase enzyme can produce two strings of XYZ↕. At this step, two versions of the original string have been produced. By repeating this process it is possible to produce 2^n versions of complementary string after n steps. Then, by the bonding of DNA strings with complementary strings, DNA strings with desired coding are purified. Therefore, the strings which present the problem desired variables remain.

[1] Polymerase Chain Reaction

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

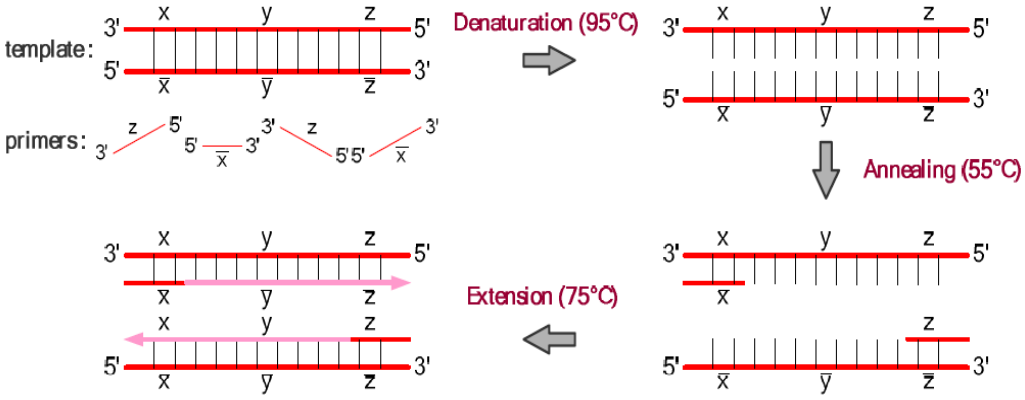

Figure 1. The process of PCR operator [15]

(4) Gel electrophoresis operator in which DNA strings with different lengths are placed on a gel surface. Then, with addition of electrical current to the surface and increasing the temperature, the strings are negatively changed and move towards positively charged electrodes. In this process, due to higher speed of strings with less length, the strings are divided based on length. So, gel electrophoresis operator purifies the strings with desired length [6].

4. DNA COMPUTING ALGORITHM

DNA computing algorithm is a meta-heuristic algorithm and a non-gradient-based method derived from numerical modeling of DNA-based computing performance by molecular computers. DNA computing algorithm works based on collective intelligence. It works with doing random search in the search space and creating the initial random population by modeling DNA-based computing operators and applies the operators derived from genetic algorithm to achieve the optimum solution of the objective function [7]. Below, the details of the stages of DNA computing algorithm are discussed.

4.1 The production of the initial population individuals strings by modeling operators of DNA-based computing

In DNA computing algorithm, an individual is recognized as a point of search space or a solution of optimization problem. In this way, by randomly producing a number of individuals that are called the initial population, the optimization process starts from several different points of search space. The number of population individuals is called population size, which is typically determined at the beginning of the algorithm. In DNA computing algorithm, all population individuals have coded string with given length and based on DNA molecule bases (A, G, C and T), which is formed by pasting the defined sub-strings of design variables.

The strings of population individuals are produced by modeling DNA-based computing operators. So, by modeling of Synthesis operator, the sub-strings of design variables are randomly produced with given length. Then, by modeling of ligation operator, the sub-

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

strings related to design variables paste together and the general string of every individual in population is formed. The modeling of PCR operator causes the sub-string related to every variable be appointed to the place of that variable.

Also, by modeling of gel electrophoresis operator, the length of general string of every individual in population that is the sum of the lengths of sub-strings related to design variables is controlled. Fig. 2 shows an example of the string of an individual in population with length 12 which contains 3 design variables with sub-strings with length 4.

$$\underbrace{ATCG}_{x_1}\ \underbrace{AATC}_{x_2}\ \underbrace{GCTA}_{x_3}$$

Figure 2. An example of produced string by modeling DNA-based computing operators

4.2 Decoding the design variables

To decode the strings related to discrete variables, it is possible to assign numbers to the DNA strings and using the one-to-one correspondence of the numbers with the numerical values of the variables. In this way, a number is assigned to every sub-string. Then, with one-to-one correspondence of the available numbers with the table of numerical values related to every variable, the numerical value of that variable is specified.

4.3 Evaluating the fitness function

In DNA computing algorithm, the unconstrained function which must be maximized is called the fitness function. Generally, DNA computing algorithm is used to optimize such functions. So, in the unconstrained maximization problems, the fitness function is the objective function. But, in the constrained minimization problems, the fitness function is achieved by some reformations.

In this research, to solve constrained minimization problems, by using the exterior penalty function method, the constrained problem is converted to unconstrained problem [16]. In this way, according to the general form of the constrained minimization problem shown in Equation (1), the constraint violation coefficient is determined with Equation (3):

$$c = \sum_{i=1}^{m} \max\left\{\frac{g_i(X)}{\bar{g}_i(X)} - 1, 0\right\} \tag{3}$$

where, X is the set of design variables, m is the number of problem constraints and $\bar{g}_i(X)$ is the allowable amount of i-th constraint. Therefore, the objective function of the unconstrained problem is defined as Equation (4):

$$Minimize: fm = f\ (1 + kc) \tag{4}$$

where, f is the objective function, fm is the reformed objective function, c is the constraint violation coefficient and k is the constraint participation coefficient in the optimization problem. Finally, the fitness function is defined by the reverse of reformed objective

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

function and as Equation (5):

$$Maximize: h = \frac{1}{fm} = \frac{1}{f\ (1+kc)} \tag{5}$$

Also, in this research, to increase the convergence power of algorithm, the elitist approach has been used. In this way, after producing the population and applying algorithm operators, the most graceful individuals are selected as the next generation [17].

4.4 Selection operator

The selection operator increases the average fitness of population individuals in generations. In this research, the roulette wheel method has been used to perform selection operator. In this method, every population individual is given the probability of being selected and the participation in new generation production. So, every individual with more fitness, has more selection probability. In this way, if a population is N individuals and fitness function value of x_i is $f(x_i)$, the selection probability of x_i is determined with Equation (6):

$$p(x_i) = \frac{f\ (x_i)}{\sum_{i=1}^{N} f\ (x_i)} \tag{6}$$

Therefore, according to the cumulative probability of selecting individuals and by randomly selecting them, the individuals are selected for the production of the new generation [18].

4.5 Crossover operator

The crossover operator rises the exploration and exploitation performance during the algorithm process. In crossover operator by using a percentage of the population called the crossover probability (ρ_c), the number of produced individuals in this operator in every generation is determined. In this research, the crossover operator has been used as the one-point method.

The one-point crossover operator breaks the DNA strings of two selected individuals as parent at one point, and replaces the broken parts. Therefore, two new individuals called child are produced [18]. Fig. 3 is an example of the one-point crossover operator. The strings break has been happened between third and fourth nucleotides on the left of strings.

Parent 1	***ATA***	***CGGTC***
Parent 2	***GCT***	***AATTG***
Child 1	***ATA***	***AATTG***
Child 2	***GCT***	***CGGTC***

Figure 3. An example of one-point crossover operator

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

4.6 Mutation operator

The mutation operator prevents the getting caught of algorithm in the trap of local optimum points. In mutation operator, by using a percentage of the population called the mutation probability (ρ_m), the number of individuals that should be randomly under the mutation operator is determined. In this research, the mutation operator has been used with uniform method. In this way, by determining a string called mask, and containing numbers 0 and 1, the sub-strings of variables corresponding to number 1 are changed and a new sub-string is determined randomly [19].

As an example, if the selected individual for the mutation operator has 3 variables and every variable has a sub-string with length 4, the general string is as Fig. 4:

$$\underbrace{ATCC}_{x_1}\underbrace{GATT}_{x_2}\underbrace{ACTA}_{x_3}$$

Figure 4. An example of the string of selected individual for mutation operator

And if the mask string be formed as Fig. 5:

0	1	0

Figure 5. An example of mask string

Then, the sub-string related to x_2 should be changed. Therefore, a string like the string in Fig. 6 is created.

$$\underbrace{ATCC}_{x_1}\underbrace{GGAC}_{x_2}\underbrace{ACTA}_{x_3}$$

Figure 6. The changed string under mutation operator

5. GENERALIZED CONVEX APPROXIMATION (GCA) METHOD

In large scale optimization problems, specifically structures optimization problems, solving the problem is nearly impossible due to various problem constraints and difficulty in achieving derivatives of objective function and problem constraints at different stages. In this condition, the problem can be solved by convex approximation. In this way, the solution of the problem converges to the optimum solution by producing series of simple sub-problems which are approximations of the original problem and solving these sub-problems with different methods [10].

In this research, the Generalized Convex Approximation (GCA) method has been used [8]. The Generalized Convex Approximation (GCA) method is a method to solve

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

optimization problems as Equation (7):

$$\begin{aligned} &Minimize: f(X) \\ &Subject\ to: g_i(X) \le 0 \qquad i = 1,2,\ldots,m \\ &\qquad X = \{x_1, x_2, \ldots, x_j, \ldots, x_n\}^T \\ &\qquad x_j^{\min} \le x_j \le x_j^{\max} \qquad j = 1,2,\ldots,n \end{aligned} \tag{7}$$

where, n is the number of design variables, m is the number of problem constrains, X is the vector of design variables, $f(X)$ is the objective function, $g_i(X)$ is the i-th constraint function of the problem and x_j^{min} and x_j^{max} are respectively the lower limit and upper limit of the j-th variable of the problem. In this way, every optimization problem by using Generalized Convex Approximation (GCA) method converts to Equation (8):

$$\begin{aligned} &Minimize: \tilde{f}(X) \\ &Subject\ to: \tilde{g}_i(X) \le 0 \qquad i = 1,2,\ldots,m \\ &\qquad X = \{x_1, x_2, \ldots, x_j, \ldots, x_n\}^T \\ &\qquad L_j^{(k)} \le x_j \le U_j^{(k)} \qquad j = 1,2,\ldots,n \end{aligned} \tag{8}$$

where, n is the number of design variables, m is the number of problem constrains, X is the vector of design variables, $\tilde{f}(X)$ is the approximation of the objective function, $\tilde{g}_i(X)$ is the approximation of i-th constraint function and $L_j^{(k)}$ and $U_j^{(k)}$ are called moving limits which are updated at each iteration and are defined as Equations (9) and (10):

$$L_j^{(k)} = \frac{x_j^{\min} + x_j^{(k)}}{2} \tag{9}$$

$$U_j^{(k)} = \frac{x_j^{\max} + x_j^{(k)}}{2} \tag{10}$$

Therefore, at each iteration by solving the k-th approximation sub-problem with an approximation method and achieving the optimum design point $X^{(k)}$, the (k+1)-th sub-problem is approximated. Generally, these approximations are based on the information of current and previous design point. Then, by solving the (k+1)-th sub-problem, the optimum design point $X^{(k+1)}$ is achieved and this process continues until the problem converges to an optimum point.

In Generalized Convex Approximation (GCA) method, the approximation of the objective function and problem constraints are defined as follows:

$$\tilde{f}(X) = \tilde{f}_0 + \sum_{j=1}^{n} \tilde{f}_j(x_j) \tag{11}$$

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

where

$$\tilde{f}_0 = f(X^{(k)}) - \sum_{j=1}^{n} b_j (x_j^{(k)} - c_j)^{r_j} \qquad j = 1, 2, \ldots, n \tag{12}$$

$$\tilde{f}_j(x_j) = b_j (x_j - c_j)^{r_j} \qquad j = 1, 2, \ldots, n \tag{13}$$

where, $f(X^{(k)})$ is the value of original function for design point $X^{(k)}$, and cj and r_j and b_j are approximation parameters which are determined for every design variable. The value of the first and second order derivatives of the original function at every design point are defined as Equations (14) and (15):

$$f_j'(X^{(k)}) = \frac{\partial f(X^{(k)})}{\partial x_j} \tag{14}$$

$$f_j''(X^{(k)}) = \frac{\partial^2 f(X^{(k)})}{\partial x_j^2} \tag{15}$$

Also, due to the accuracy of the approximation,
$\tilde{f}_j''(X^{(k)}) = f_j''(X^{(k)})$.

In Generalized Convex Approximation (GCA) method, the approximation parameters are determined as follows:

$$c_j = \frac{-x_j^{(k)} + d^{\frac{1}{(r_j - 1)}} x_j^{(k-1)}}{-1 + d^{\frac{1}{(r_j - 1)}}} \tag{16}$$

$$r_j = 1 + \frac{x_j^{(k)} - c_j}{e} \tag{17}$$

$$b_j = \frac{f_j'(X^{(k)})}{r_j (x_j^{(k)} - c_j)^{r_j - 1}} \tag{18}$$

where

$$d = \frac{f_j'(X^{(k)})}{f_j'(X^{(k-1)})} \tag{19}$$

$$e = \frac{f_j'(X^{(k)})}{f_j''(X^{(k)})} \tag{20}$$

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

By combination of Equations (16) and (17), the *cj* parameter is defined as Equation (21):

$$c_j = \frac{-x_j^{(k)} + d^{\frac{e}{(x_j^{(k)} - c_j)}} x_j^{(k-1)}}{-1 + d^{\frac{e}{(x_j^{(k)} - c_j)}}} \tag{21}$$

where, by solving the equation $f(c_j)=0$ the value of *cj* is achieved. Also, to prevent the difficulty in computing the functions value and derivatives, the limitation $c_j \leq x_j^{min}$ is used. At the first iteration, the information at design point $X^{(k-1)}$ is not available. So, the approximation parameters are determined as Equation (22):

$$\begin{aligned} c_j &= 0 \\ r_j &= \frac{f_j''(X^{(k)})}{f_j'(X^{(k)})} x_j^{(k)} + 1 \\ b_j &= \frac{f_j'(X^{(k)})}{r_j (x_j^{(k)})^{r_j - 1}} \end{aligned} \tag{22}$$

In this research, the simplified form of Generalized Convex Approximation (GCA) method that uses first order derivative is used. In this method, the approximation process is like second order form. The only difference is in determining the approximation parameters. The approximation parameters *cj* and r_j and b_j in the simplified form of Generalized Convex Approximation (GCA) method are determined as follows:

$$(i)\ \frac{f_j'(X^{(k)})}{f_j'(X^{(k-1)})} > 0: \begin{cases} c_j = 0 \\ r_j = \begin{cases} 1 & if \quad \frac{x_j^{(k)}}{x_j^{(k-1)}} = 1 \\ 1 + \frac{\log(\frac{f_j'(X^{(k)})}{f_j'(X^{(k-1)})})}{\log(\frac{x_j^{(k)}}{x_j^{(k-1)}})} & else \end{cases} \\ b_j = \frac{f_j'(X^{(k)})}{r_j (x_j^{(k)})^{r_j - 1}} \end{cases} \tag{23}$$

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

$$(ii)\ \frac{f_j'(X^{(k)})}{f_j'(X^{(k-1)})}<0:\ \begin{cases} c_j=\dfrac{x_j^{(k)}-dx_j^{(k-1)}}{1-d}\le x_j^{\min} \\ r_j=2 \\ b_j=\dfrac{f_j'(X^{(k)})}{2(x_j^{(k)}-c_j)} \end{cases} \tag{24}$$

$$(iii)\ f_j'(X^{(k)})=0\ and\ f_j'(X^{(k-1)})=0:\ \begin{cases} c_j=0,\ r_j=1 \\ b_j=f_j'(X^{(k)})\ or\ f_j'(X^{(k-1)})=0 \end{cases} \tag{25}$$

$$(iv)\ f_j'(X^{(k)})=0\ and\ f_j'(X^{(k-1)})\neq 0:\ \begin{cases} c_j=0,\ r_j=2 \\ b_j=\begin{cases} f_j'(X^{(k-1)}) & if\ \ \dfrac{x_j^{(k)}}{x_j^{(k-1)}}=1 \\ \dfrac{f_j'(X^{(k-1)})}{2(x_j^{(k-1)}-x_j^{(k)})} & else \end{cases} \end{cases} \tag{26}$$

$$(v)\ f_j'(X^{(k)})\neq 0\ and\ f_j'(X^{(k-1)})=0:\ \begin{cases} c_j=0,\ r_j=2 \\ b_j=\begin{cases} f_j'(X^{(k)}) & if\ \ \dfrac{x_j^{(k)}}{x_j^{(k-1)}}=1 \\ \dfrac{f_j'(X^{(k)})}{2(x_j^{(k)}-x_j^{(k-1)})} & else \end{cases} \end{cases} \tag{27}$$

At the first iteration, since there is no information at design point $X^{(k-1)}$, the approximation parameters are determined as Equation (28):

$$\begin{aligned} c_j&=0 \\ r_j&=1 \\ b_j&=f_j'(X^{(k)}) \end{aligned} \tag{28}$$

The first and second order derivatives of the functions at Generalized Convex Approximation (GCA) method are achieved from Equations (29) and (30):

$$\tilde{f}_j'(X^{(k)})=b_j r_j x_j^{r_j-1} \tag{29}$$

$$\tilde{f}_j''(X^{(k)})=b_j r_j (r_j-1) x_j^{r_j-2} \tag{30}$$

In Generalized Convex Approximation (GCA) method, at each iteration after production of sub-problems that include the approximation of objective function and constraints of the original problem, many methods can be used to solve sub-problems. In this research, the exterior penalty function method has been used to solve the constrained sub-problems [9].

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

6. THE COMBINATION OF DNA COMPUTING ALGORITHM AND GENERALIZED CONVEX APPROXIMATION (GCA) METHOD

In this research, in order to minimize the weight of truss structures, the cross-section areas of the elements as discrete variables are optimized by DNA computing algorithm, and the coordinates of truss nodes as continuous variables are optimized by Generalized Convex Approximation (GCA) method.

In this way, at the first stage by selecting appropriate and similar cross sections for truss elements, the coordinates of truss nodes are optimized by Generalized Convex Approximation (GCA) method. Then, at the second stage with achieved coordinates from the first stage and by using DNA computing algorithm, the cross-section areas of the elements are optimized.

In Fig. 7 the flowchart of the process of size and geometry optimization of truss structures by using the combination of DNA computing algorithm and Generalized Convex Approximation (GCA) method is shown.

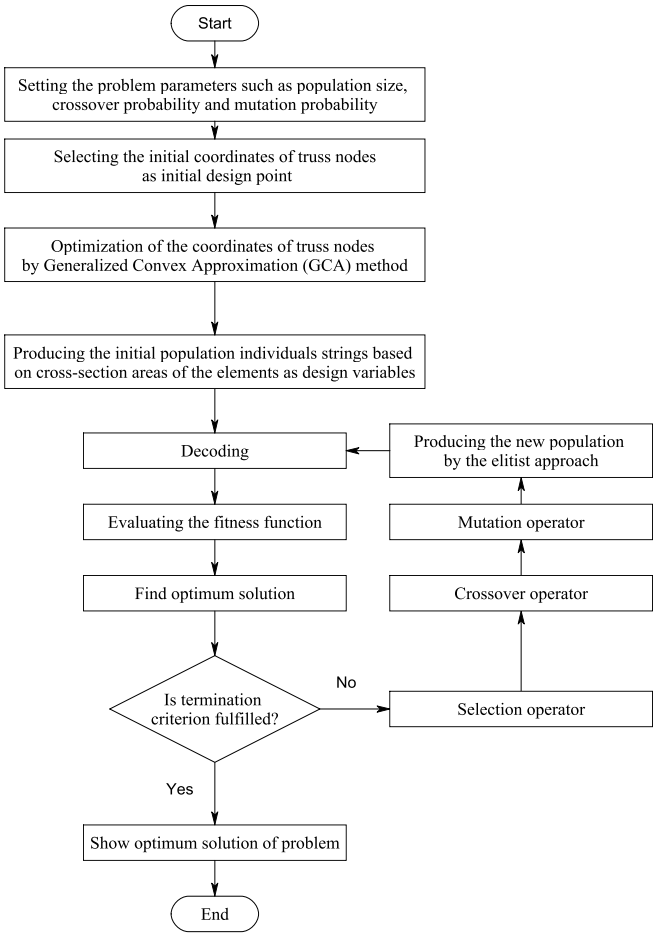

Figure 7. The flowchart of optimization of trusses using combined algorithm DNA-GCA

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

7. NUMERICAL EXAMPLES

In this research, five problems about minimizing the weight of truss structures in order to control the numerical efficacy of the combination of DNA computing algorithm and Generalized Convex Approximation (GCA) method have been solved. This method has been coded in MATLAB software. The truss structures have been analyzed by using the stiffness method. The results of optimization have been compared with the results of different references. The parameters set for solving the problems are summarized in Table 1.

Table 1: The parameters set for solving the problems

Parameter	Value
Population size	100
Crossover probability (ρ_c)	80%
Mutation probability (ρ_m)	10%

Since the discussed method for optimization process is stochastic, the optimization program has been run 10 times and the average of 10 runs and also the best run have been reported.

7.1 Twenty five-bar truss

A twenty five-bar truss as Fig. 8 has been considered. The coordinates of truss nodes have been shown in Table 2. The grouping of truss elements as the size variables has been shown in Table 3. The data to design truss has been shown in Table 4. Modulus of elasticity of the material is 68.95 Gpa and the specific weight is 2720 kg/m^3. The results of size and geometry optimization of the truss have been compared with other references in Table 5. The stresses of the elements have been shown in Table 6. The displacements of truss nodes have been shown in Table 7. The optimum geometry of the truss has been shown in Fig. 9. Also, the convergence process of truss weight has been shown in Fig. 10.

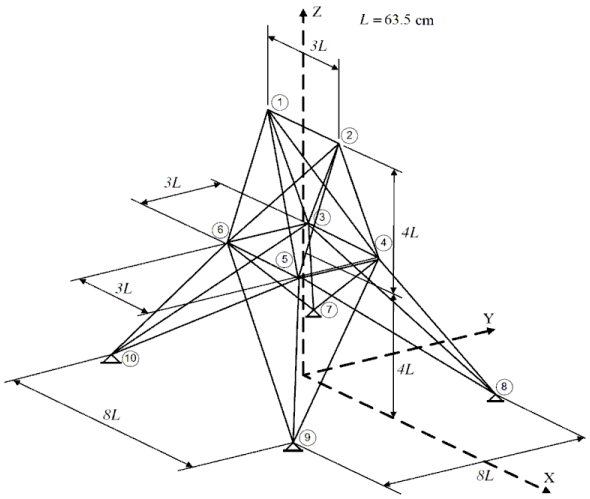

Figure 8. Twenty five-bar truss

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Table 2: The coordinates of truss nodes for twenty five-bar truss

Node	*x(cm)*	*y(cm)*	*z(cm)*
1	-95.25	0	508
2	95.25	0	508
3	-95.25	95.25	254
4	95.25	95.25	254
5	95.25	-95.25	254
6	-95.25	-95.25	254
7	-254	254	0
8	254	254	0
9	254	-254	0
10	-254	-254	0

Table 3: The grouping of truss elements for twenty five-bar truss

Group	Members (end nodes)
A_1	1(1,2)
A_2	2(1,4),3(2,3),4(1,5),5(2,6)
A_3	6(2,5),7(2,4),8(1,3),9(1,6)
A_4	10(3,6),11(4,5)
A_5	12(3,4),13(5,6)
A_6	14(3,10),15(6,7),16(4,9),17(5,8)
A_7	18(3,8),19(4,7),20(6,9),21(5,10)
A_8	22(3,7),23(4,8),24(5,9),25(6,10)

Table 4: The data to design twenty five-bar truss

Loading data	Node	F_x *(KN)*	F_y *(KN)*	F_z *(KN)*
	1	4.454	-44.537	-44.537
	2	0	-44.537	-44.537
	3	2.227	0	0
	6	2.672	0	0
Design variables	Size variables $A_1; A_2; A_3; A_4; A_5; A_6; A_7; A_8$ Geometry variables $x_4= x_5=- x_3=- x_6; y_4= y_3=- y_5=- y_6; z_4= z_3= z_5= z_6$ $x_8= x_9=- x_7=- x_{10}; y_8= y_7=- y_9=- y_{10}$			
Constraint data	Stress constraints $(\sigma_t)_i \le 275.8\ Mpa;$ $i=1,2,\ldots,25$ $\lvert(\sigma_c)_i\rvert \le 275.8\ Mpa;$ $i=1,2,\ldots,25$ Displacement constraints $\lvert\Delta_i\rvert \le 0.89\ cm;$ $i=1,2,\ldots,6$ Side constraints of geometry variables $50.8cm \le x_4 \le 152.4cm$ $101.6cm \le y_4 \le 203.2cm$ $228.6cm \le z_4 \le 330.2cm$ $101.6cm \le x_8 \le 203.2cm$			

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

	$254cm \le y_8 \le 355.6cm$
List of the available profiles	$A_i \in S=\{0.645I \ (I=1,2,...,26), \ 18.064, \ 19.355, \ 20.645, \ 21.935\}cm^2 \quad i=1,2,...,25$

Table 5: The optimization results of the size and geometry of twenty five-bar truss

Design variables		Wu and Chow [20]	Kaveh and Kalatjari [21]	Rahami et al [22]	Present work
Size variables (cm^2)	A_1	0.645	0.645	0.645	0.645
	A_2	1.29	0.645	0.645	0.645
	A_3	7.097	7.097	7.097	5.805
	A_4	1.29	0.645	0.645	0.645
	A_5	1.935	0.645	0.645	0.645
	A_6	0.645	0.645	0.645	0.645
	A_7	1.29	0.645	1.29	0.645
	A_8	5.806	6.452	5.16	6.45
Geometry variables (cm)	x_4	104.318	92.024	83.9436	92.29
	y_4	135.814	148.742	136.0584	143.093
	z_4	316.484	293.599	329.9694	320.342
	x_8	129.032	118.008	111.2078	129.108
	y_8	333.959	324.993	347.569	347.53
Weight (kg)		61.83	56.29	54.53	53.0067

Table 6: The stresses of the elements of twenty five-bar truss

Member	Stress (kg/cm^2)	Member	Stress (kg/cm^2)
1	135.74	14	-487.17
2	-211.50	15	467.94
3	383.53	16	-519.97
4	-960.02	17	426.99
5	-427.34	18	-434.61
6	-1077.97	19	629.03
7	116.89	20	-1302.82
8	168.65	21	-171.49
9	-1031.63	22	241.71
10	397.00	23	63.21
11	527.95	24	-1003.52
12	-143.53	25	-828.92
13	-379.01		

Table 7: The displacements of truss nodes for twenty five-bar truss

Node	Δx (cm)	Δy (cm)	Δz (cm)
1	0.8274	-0.8900	-0.4687
2	0.8649	-0.8806	-0.4616
3	0.6701	-0.4239	-0.1880
4	0.6316	-0.3932	-0.1366

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

5	0.6353	-0.6123	-0.1985
6	0.7368	-0.5887	-0.2560
7	0.0000	0.0000	0.0000
8	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000
10	0.0000	0.0000	0.0000

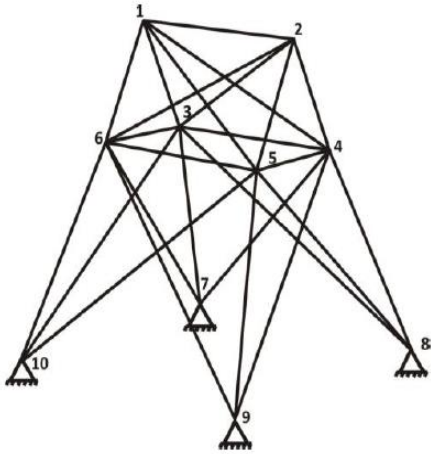

Figure 9. The optimum geometry of twenty five-bar truss

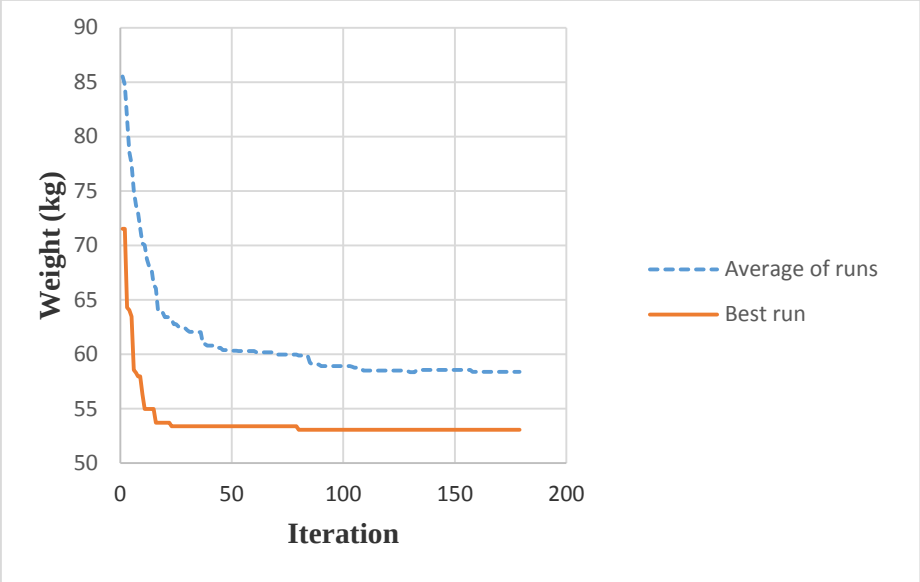

Figure 10. The weight convergence process of twenty five-bar truss

7.2 Eighteen-bar truss

An eighteen-bar truss as Fig. 11 has been considered. The data to design truss has been shown in Table 8. Modulus of elasticity of the material is 10000 ksi and the specific weight is 0.1 Ib/in^3. The results of size and geometry optimization of the truss have been compared

with other references in Table 9. The stresses of the elements have been shown in Table 10. The optimum geometry of the truss has been shown in Fig. 12. Also, the convergence process of truss weight has been shown in Fig. 13.

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

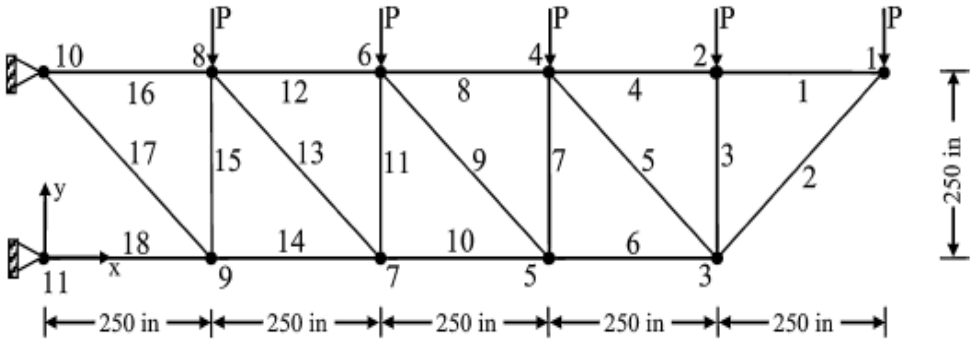

Figure 11. Eighteen-bar truss

Table 8: The data to design eighteen-bar truss

Loading data	Node	F_x (Kips)	F_y (Kips)	F_z (Kips)
	1	0	-20	0
	2	0	-20	0
	4	0	-20	0
	6	0	-20	0
	8	0	-20	0
Design variables	Size variables $A_1= A_4= A_8= A_{12}= A_{16}$; $A_2= A_6= A_{10}= A_{14}= A_{18}$; $A_3= A_7= A_{11}= A_{15}$; $A_5= A_9= A_{13}= A_{17}$ Geometry variables x_3; y_3; x_5; y_5; x_7; y_7; x_9; y_9			
Constraint data	Stress constraints $(\sigma_t)_i \le 20$ Ksi; $i=1,2,...,18$ $\lvert(\sigma_c)_i\rvert \le 20$ Ksi; $i=1,2,...,18$ Euler buckling stress constraints $\lvert(\sigma_c)_i\rvert \le \alpha A_i E/L_i^2$, $\alpha=4$; $i=1,2,...,18$ Side constraints of geometry variables $775\ in \le x_3 \le 1225\ in$ $-225\ in \le y_3 \le 245\ in$ $525\ in \le x_5 \le 975\ in$ $-225\ in \le y_5 \le 245\ in$ $275\ in \le x_7 \le 725\ in$ $-225\ in \le y_7 \le 245\ in$ $25\ in \le x_9 \le 475\ in$ $-225\ in \le y_9 \le 245\ in$			
List of the available profiles	$A_i \epsilon S=\{2.00, 2.25, ..., 21.50, 21.75\} in^2$			

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Table 9: The optimization results of the size and geometry of eighteen-bar truss

Design variables		Rajeev and Krishnamoorthy [23]	Yang [24]	Kang and Zong [25]	Present work
Size variables (in^2)	A_1	12.50	12.61	12.65	12.50
	A_2	16.25	18.10	7.22	17.50
	A_3	8.00	5.47	6.17	5.75
	A_5	4.00	3.54	3.55	3.75
Geometry variables (in)	x_3	891.90	914.50	903.10	907.2491
	y_3	145.30	183.00	174.30	179.8672
	x_5	610.60	647.00	630.30	636.7873
	y_5	118.20	147.40	136.30	141.8272
	x_7	385.40	414.20	402.10	407.9442
	y_7	72.50	100.40	90.50	94.05591
	x_9	184.40	200.00	195.30	198.7897
	y_9	23.40	31.90	30.60	29.5158
Weight (Ib)		4616.80	4552.80	4515.6	4512.262

Table 10: The stresses of the elements of eighteen-bar truss

Member	Stress (Ib/in^2)	Member	Stress (Ib/in^2)
1	7819.47	10	-12808
2	-5701.06	11	-6906.20
3	-5767.01	12	19168.13
4	9935.48	13	1753.73
5	11214.86	14	-14608.41
6	-9383.01	15	-4395.81
7	-9347.36	16	19999.99
8	16116.99	17	19911.73
9	6103.33	18	-17330.79

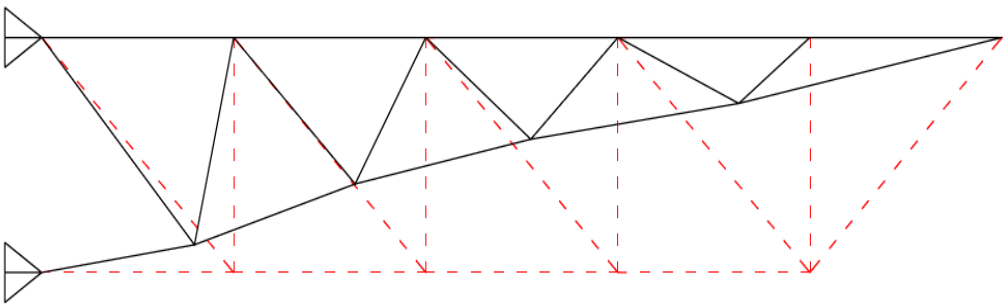

Figure 12. The optimum geometry of eighteen-bar truss

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

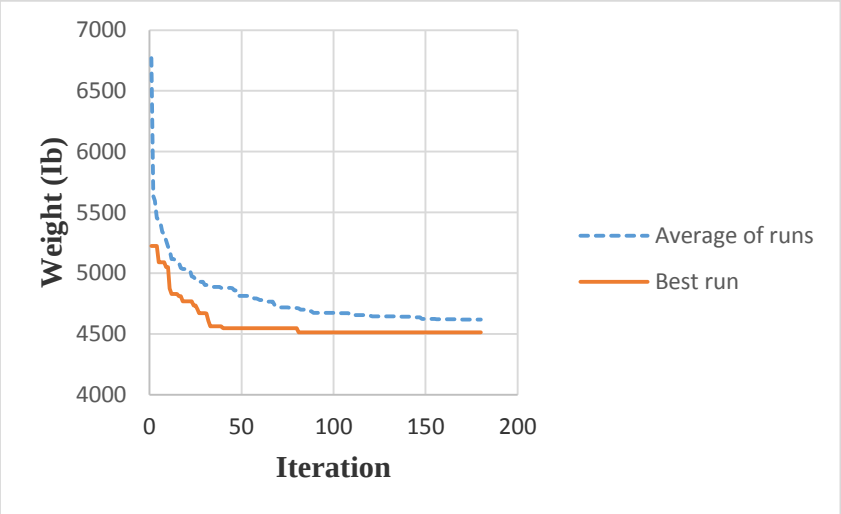

Figure 13. The weight convergence process of eighteen-bar truss

7.3 Fifteen-bar truss

A fifteen-bar truss as Fig. 14 has been considered. The data to design truss has been shown in Table 11. Modulus of elasticity of the material is 10000 ksi and the specific weight is 0.1 lb/in^3. The results of size and geometry optimization of the truss have been compared with other references in Table 12. The stresses of the elements have been shown in Table 13. The optimum geometry of the truss has been shown in Fig. 15. Also, the convergence process of truss weight has been shown in Fig. 16.

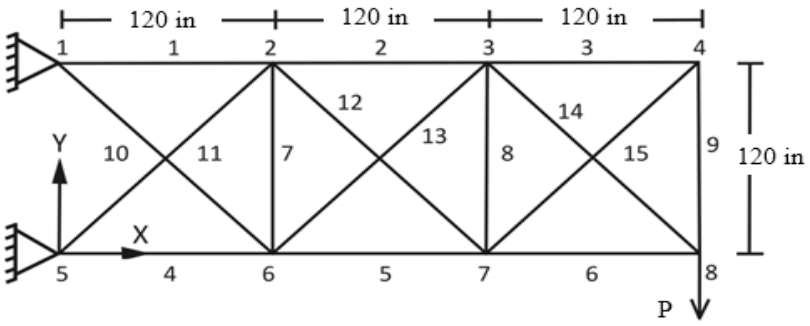

Figure 14. Fifteen-bar truss

Table 11: The data to design fifteen-bar truss

Loading data	Node	F_x *(Kips)*	F_y *(Kips)*	F_z *(Kips)*
	8	0	-10	0
Design variables	Size variables $A_1; A_2; A_3; A_4; A_5; A_6; A_7; A_8; A_9; A_{10}; A_{11}; A_{12}; A_{13}; A_{14}; A_{15}$ Geometry variables $x_2 = x_6; x_3 = x_7; y_2; y_3; y_4; y_6; y_7; y_8$			

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Constraint data	Stress constraints $(\sigma_t)_i \le 25$ Ksi; $i=1,2,...,15$ $\lvert(\sigma_c)_i\rvert \le 25$ Ksi; $i=1,2,...,15$ Side constraints of geometry variables $100\ in \le x_2 \le 140\ in$ $220\ in \le x_3 \le 260\ in$ $100\ in \le y_2 \le 140\ in$ $100\ in \le y_3 \le 140\ in$ $50\ in \le y_4 \le 90\ in$ $-20\ in \le y_6 \le 20\ in$ $-20\ in \le y_7 \le 20\ in$ $20\ in \le y_8 \le 60\ in$
List of the available profiles	$A_i \in S=\{0.111, 0.141, 0.174, 0.22, 0.27, 0.287, 0.347, 0.44, 0.539, 0.954, 1.081, 1.174, 1.333, 1.488, 1.764, 2.142, 2.697, 2.8, 3.131, 3.565, 3.813, 4.805, 5.952, 6.572, 7.192, 8.525, 9.3, 10.85, 13.33, 14.29, 17.17, 19.18\} in^2$

Table 12: The optimization results of the size and geometry of fifteen-bar truss

Design variables		Wu and Chow [26]	Hwang and He [27]	Tang et al [28]	Present work
Size variables (in^2)	A_1	1.174	0.954	1.081	1.081
	A_2	0.954	1.081	0.539	0.539
	A_3	0.44	0.44	0.287	0.27
	A_4	1.333	1.174	0.954	0.954
	A_5	0.954	1.488	0.954	0.954
	A_6	0.174	0.27	0.22	0.22
	A_7	0.44	0.27	0.111	0.111
	A_8	0.44	0.347	0.111	0.111
	A_9	1.081	0.22	0.287	0.27
	A_{10}	1.333	0.44	0.22	0.287
	A_{11}	0.174	0.22	0.44	0.44
	A_{12}	0.174	0.44	0.44	0.287
	A_{13}	0.347	0.347	0.111	0.141
	A_{14}	0.347	0.27	0.22	0.27
	A_{15}	0.44	0.22	0.347	0.27
Geometry variables (in)	x_2	123.189	118.346	133.612	123.5286
	x_3	231.595	225.209	234.752	239.1095
	y_2	107.189	119.046	100.449	123.7912
	y_3	119.175	105.086	104.738	115.2112
	y_4	60.462	63.375	73.762	72.968
	y_6	16.728	-20.0	-10.067	-8.153
	y_7	15.565	-20.0	-1.339	3.8959
	y_8	36.645	57.722	50.402	42.6028
Weight (Ib)		120.52	104.573	79.82	79.8065

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Table 13: The stresses of the elements of fifteen-bar truss

Member	Stress (Ib/in ^{2})	Member	Stress (Ib/in ^{2})
1	23249.16	9	19291.09
2	24547.4	10	24498.17
3	22192.73	11	-23161.28
4	-23953	12	23733.03
5	-16427.66	13	-24216.56
6	-24835.33	14	22481.39
7	5040.43	15	-24129.05
8	-14690.79		

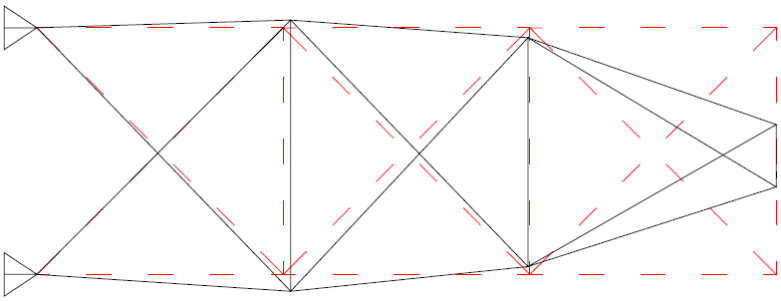

Figure 15. The optimum geometry of fifteen-bar truss

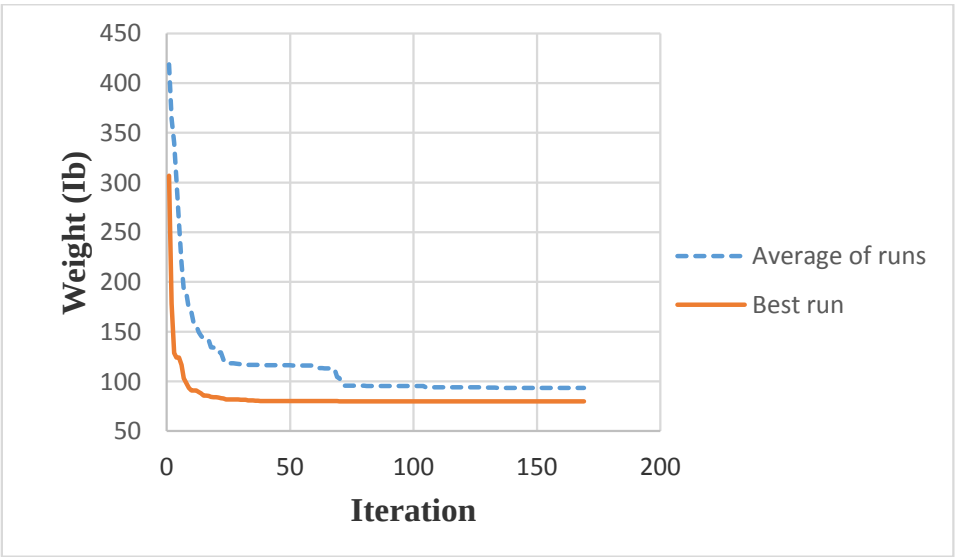

Figure 16. The weight convergence process of fifteen-bar truss

7.4 Thirty nine-bar truss

A thirty nine-bar truss as Fig. 17 has been considered. The bottom and top nodes are fixed while all the intermediate node positions will be redesigned. The coordinates of three bottom nodes 1, 2, 3 and three top nodes 13, 14 and 15 have been shown in Table 14. The grouping of truss elements as the size variables has been shown in Table 15. The data to design truss

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

has been shown in Table 16. Modulus of elasticity of the material is 210 Gpa and the specific weight is 7800 kg/m^3. The results of size and geometry optimization of the truss have been compared with other references in Table 17. The stresses of the elements have been shown in Table 18. The displacements of truss nodes have been shown in Table 19. The optimum geometry of the truss has been shown in Fig. 18. Also, the convergence process of truss weight has been shown in Fig. 19.

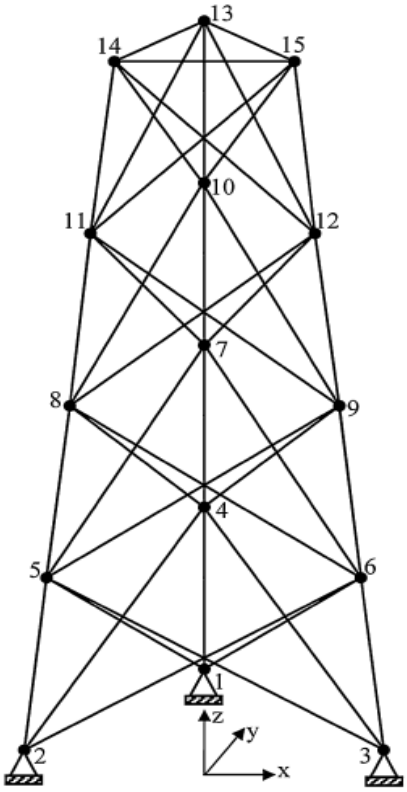

Figure 17. Thirty nine-bar truss

Table 14: The coordinates of truss fixed nodes for thirty nine-bar truss

Node	*x(m)*	*y(m)*	*z(m)*
1	0	1	0
2	$-\frac{\sqrt{3}}{2}$	-0.5	0
3	$\frac{\sqrt{3}}{2}$	-0.5	0
13	0	0.28	4
14	$-\frac{0.42}{\sqrt{3}}$	-0.14	4
15	$\frac{0.42}{\sqrt{3}}$	-0.14	4

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Table 15: The grouping of truss elements for thirty nine-bar truss

Group	Members (end nodes)
A_1	1(1,4), 2(2,5), 3(3,6)
A_2	4(4,7), 5(5,8), 6(6,9)
A_3	7(7,10), 8(8,11), 9(9,12)
A_4	10(10,13), 11(11,14), 12(12,15)
A_5	Rest of the elements

Table 16: The data to design thirty nine-bar truss

Loading data	Node	F_x (KN)	F_y (KN)	F_z (KN)
	13	0	10	0
	14	0	10	0
	15	0	10	0
Design variables	Size variables $A_1; A_2; A_3; A_4; A_5$ Geometry variables $y_4; z_4; y_7; z_7; y_{10}; z_{10}$			
Constraint data	Stress constraints $(\sigma_t)_i \leq 240$ Mpa; $i=1,2,...,39$ $\lvert(\sigma_c)_i\rvert \leq 240$ Mpa; $i=1,2,...,39$ Displacement constraints of node13 in direction y $\lvert\Delta_{y,13}\rvert \leq 4$ mm; Side constraints of geometry variables $0.28m \leq y_4 \leq 1m$ $0 \leq z_4 \leq 2m$ $0.28m \leq y_7 \leq 1m$ $1m \leq z_7 \leq 3m$ $0.28m \leq y_{10} \leq 1m$ $2m \leq z_{10} \leq 4m$			
List of the available profiles	$A_i \in S=\{0.1, 0.2, 0.3, ..., 13\} cm^2$			

Table 17: The optimization results of the size and geometry of thirty nine-bar truss

Design variables		Wang et al [29]	Present work
Size variables (cm^2)	A_1	11.01	12.7
	A_2	8.63	10.4
	A_3	6.69	7.2
	A_4	4.11	3.3
	A_5	4.37	1.7
Geometry variables (m)	y_4	0.805	0.8894
	z_4	1.186	1.3405
	y_7	0.654	0.6816
	z_7	2.204	2.312
	y_{10}	0.466	0.4824
	z_{10}	3.092	3.3029
Weight (kg)		203.18	137.826

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Table 18: The stresses of the elements of thirty nine-bar truss

Member	Stress (kg/cm^2)	Member	Stress (kg/cm^2)
1	-567.45	21	251.66
2	283.74	22	17.92
3	283.74	23	251.66
4	-535.83	24	17.92
5	267.89	25	-429.24
6	267.89	26	-429.24
7	-557.11	27	438.43
8	278.62	28	-9.59
9	278.62	29	438.43
10	-483.88	30	-9.59
11	241.99	31	-552.56
12	241.99	32	-552.56
13	-379.70	33	624.65
14	-379.70	34	-72.02
15	557.91	35	624.65
16	-178.43	36	-72.02
17	557.91	37	-200.87
18	-178.43	38	-200.87
19	-269.32	39	401.75
20	-269.32		

Table 19: The displacements of truss nodes for thirty nine-bar truss

Node	Δx (cm)	Δy (cm)	Δz (cm)
1	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0000
4	0.0000	0.1119	-0.0272
5	0.0360	0.0494	0.0136
6	-0.0360	0.0494	0.0136
7	0.0000	0.1538	-0.0442
8	-0.0210	0.1902	0.0221
9	0.0210	0.1902	0.0221
10	0.0000	0.3027	-0.0416
11	0.0271	0.2558	0.0208
12	-0.0271	0.2558	0.0208
13	0.0000	0.3998	-0.0308
14	-0.0046	0.4000	0.0154
15	0.0046	0.4000	0.0154

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

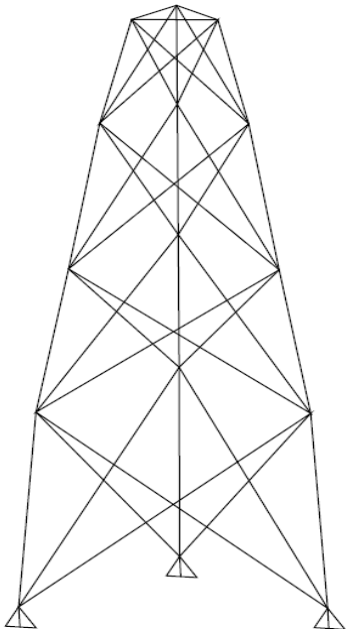

Figure 18. The optimum geometry of thirty nine-bar truss

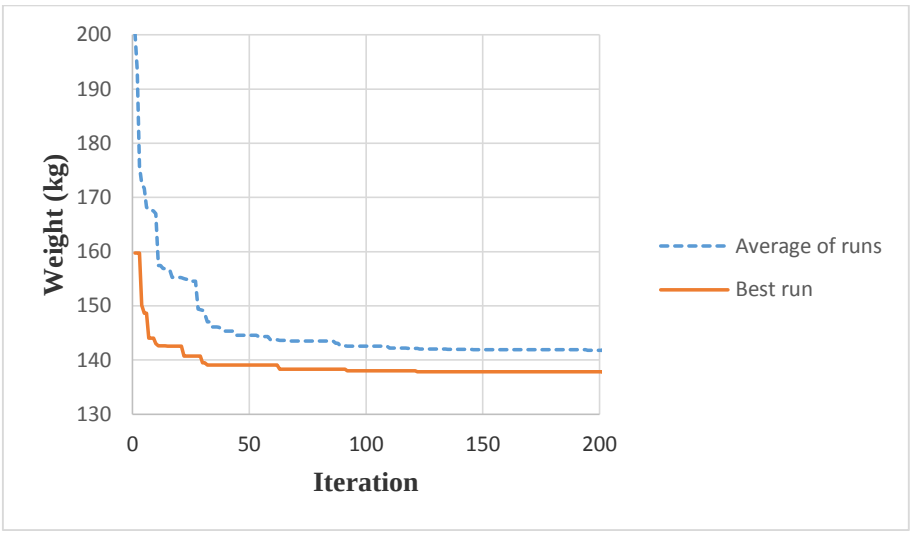

Figure 19. The weight convergence process of thirty nine-bar truss

7.5 Forty seven-bar truss

A forty seven-bar truss as Fig. 20 has been considered. The data to design truss has been shown in Table 20. Modulus of elasticity of the material is 30000 ksi and the specific weight is 0.3 Ib/in^3. The results of size and geometry optimization of the truss have been compared with other references in Table 21. The stresses of the elements have been shown in Table 22. The optimum geometry of the truss has been shown in Fig. 21. Also, the convergence process of truss weight has been shown in Fig. 22.

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

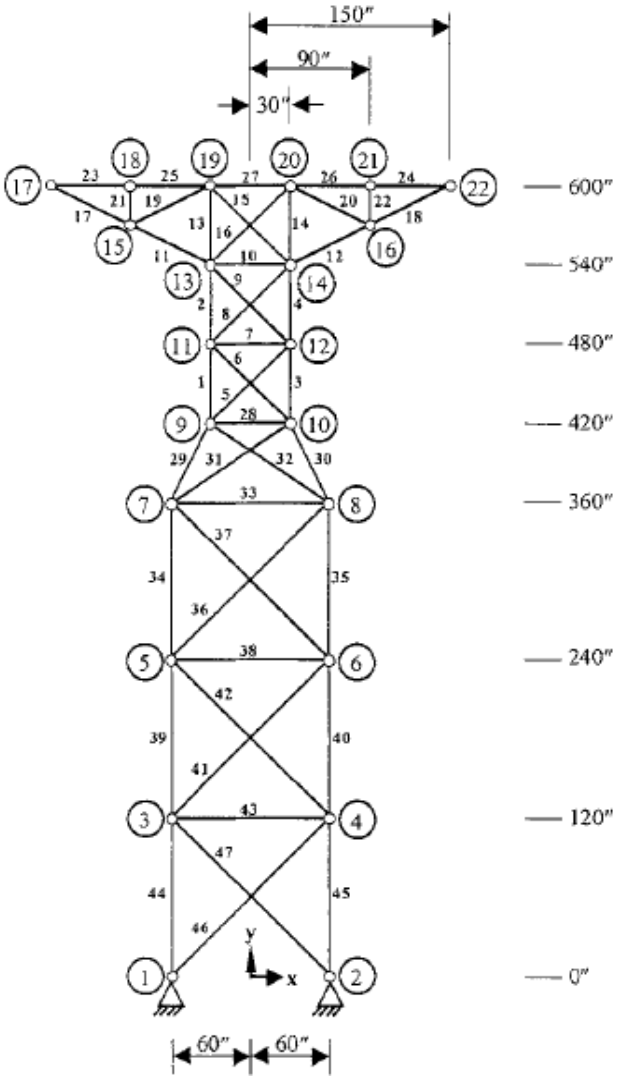

Figure 20. Forty seven-bar truss

Table 20: The data to design forty seven-bar truss

	Node	F_x (Kips)	F_y (Kips)	F_z (Kips)
Loading data	17	6	-14	0
	22	6	-14	0
Design variables	Size variables $A_3= A_1$; $A_4=A_2$; $A_5= A_6$; A_7; $A_8= A_9$; A_{10}; $A_{12}= A_{11}$; $A_{14}= A_{13}$; $A_{15}= A_{16}$; $A_{18}= A_{17}$; $A_{20}= A_{19}$; $A_{22}= A_{21}$; $A_{24}= A_{23}$; $A_{26}= A_{25}$; A_{27}; A_{28}; $A_{30}= A_{29}$; $A_{31}= A_{32}$; A_{33}; $A_{35}= A_{34}$; $A_{36}= A_{37}$; A_{38}; $A_{40}= A_{39}$; $A_{41}= A_{42}$; A_{43}; $A_{45}= A_{44}$; $A_{46}= A_{47}$ Geometry variables $x_2=-x_1$; $x_4=-x_3$; $y_4=y_3$; $x_6=-x_5$; $y_6=y_5$; $x_8=-x_7$; $y_8=y_7$; $x_{10}=-x_9$; $y_{10}=y_9$; $x_{12}=-x_{11}$; $y_{12}=y_{11}$; $x_{14}=-x_{13}$; $y_{14}=y_{13}$;			

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

Constraint data	$x_{20}=-x_{19}$; $y_{20}=y_{19}$; $x_{21}=-x_{18}$; $y_{21}=y_{18}$ Stress constraints $(\sigma_t)_i \le 20$ Ksi; $i=1,2,...,47$ $\lvert(\sigma_c)_i\rvert \le 15$ Ksi; $i=1,2,...,47$ Euler buckling stress constraints $\lvert(\sigma_c)_i\rvert \le \alpha A_i E/L_i^2$, $\alpha=3.96$; $i=1,2,...,47$ Side constraints of geometry variables $0 \le x_2 \le 150$ in $0 \le x_4 \le 150$ in $0 \le y_4 \le 240$ in $0 \le x_6 \le 150$ in 120 in $\le y_6 \le 360$ in $0 \le x_8 \le 150$ in 240 in $\le y_8 \le 420$ in $0 \le x_{10} \le 75$ in 360 in $\le y_{10} \le 480$ in $0 \le x_{12} \le 75$ in 420 in $\le y_{12} \le 540$ in $0 \le x_{14} \le 75$ in 480 in $\le y_{14} \le 600$ in $0 \le x_{20} \le 75$ in 540 in $\le y_{20} \le 660$ in $0 \le x_{21} \le 150$ in 540 in $\le y_{21} \le 660$ in
List of the available profiles	$A_i \in S=\{0.1, 0.2, 0.3, ..., 5.0\}$ in^2

Table 21: The optimization results of the size and geometry of forty seven-bar truss

Design variables		Hasancebi and Erbatur [30]	Salajegheh and Vanderplaats [31]	Hansen and Vanderplaats [32]	Present work
Size variables (in^2)	A_3	2.50	2.61	2.42	2.70
	A_4	2.20	2.56	2.35	2.50
	A_5	0.70	0.69	0.82	0.70
	A_7	0.10	0.47	0.10	0.10
	A_8	1.30	0.80	0.86	0.90
	A_{10}	1.30	1.13	1.15	1.10
	A_{12}	1.80	1.71	1.77	1.80
	A_{14}	0.50	0.77	0.67	0.70
	A_{15}	0.80	1.09	0.86	0.90
	A_{18}	1.20	1.34	1.24	1.30
	A_{20}	0.40	0.36	0.33	0.30
	A_{22}	1.20	0.97	1.22	1.10
	A_{24}	0.90	1.00	0.93	1.00
	A_{26}	1.00	1.03	0.86	0.90
	A_{27}	3.60	0.88	0.69	0.80
	A_{28}	0.10	0.55	0.15	0.10
	A_{30}	2.40	2.59	2.46	2.70
	A_{31}	1.10	0.84	0.90	0.80
	A_{33}	0.10	0.25	0.10	0.10

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

	A_{35}	2.70	2.86	2.74	3.00
	A_{36}	0.80	0.92	0.92	0.90
	A_{38}	0.10	0.67	0.1	0.10
	A_{40}	2.80	3.06	2.94	3.20
	A_{41}	1.30	1.04	1.13	1.00
	A_{43}	0.20	0.10	0.10	0.10
	A_{45}	3.00	3.13	3.12	3.30
	A_{46}	1.20	1.12	1.10	1.20
	x_2	114.00	107.76	107.10	100.9724
	x_4	97.00	89.15	91.20	80.4772
	y_4	125.00	137.98	122.80	136.8699
	x_6	76.00	66.75	74.20	64.3908
	y_6	261.00	254.47	241.40	247.0491
	x_8	69.00	57.38	65.50	55.2589
	y_8	316.00	342.16	324.60	338.4534
	x_{10}	56.00	49.85	57.10	48.7333
Geometry variables (in)	y_{10}	414.00	417.17	400.40	409.738
	x_{12}	50.00	44.66	49.30	43.4742
	y_{12}	463.00	475.35	472.30	472.1479
	x_{14}	54.00	41.09	47.40	44.8349
	y_{14}	524.00	513.15	507.50	512.1903
	x_{20}	1.00	17.90	3.90	3.8416
	y_{20}	587.00	597.92	586.50	591.1449
	x_{21}	99.00	93.54	83.30	84.50416
	y_{21}	631.00	623.94	636.00	630.3472
Weight (Ib)		1925.79	1900.00	1850.4	1860.161

Table 22: The stresses of the elements of forty seven-bar truss

Member	Stress (Ib/in ^{2})	Member	Stress (Ib/in ^{2})	Member	Stress (Ib/in ^{2})
1	2333.29	17	-14980.40	33	4712.87
2	791.45	18	-10016.58	34	5644.96
3	-12931.78	19	-4489.07	35	-14885.74
4	-10973.86	20	19865.83	36	3728.92
5	8021.24	21	-9476.08	37	-4488.52
6	-6709.07	22	-14644.85	38	-444.23
7	8073.72	23	12584.71	39	6299.89
8	4487.23	24	19449.1	40	-14991.32
9	-11371.45	25	12938.38	41	959.16
10	-14278.88	26	19995.67	42	-1750.75
11	-13684.85	27	18745.22	43	4377.84
12	-14414.45	28	-1248.42	44	6506.11
13	12992.18	29	4510.74	45	-14998.30
14	-4257.07	30	-14885.74	46	581.01
15	-3472.72	31	5019.47	47	-975.10
16	11865.98	32	-5976.77		

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

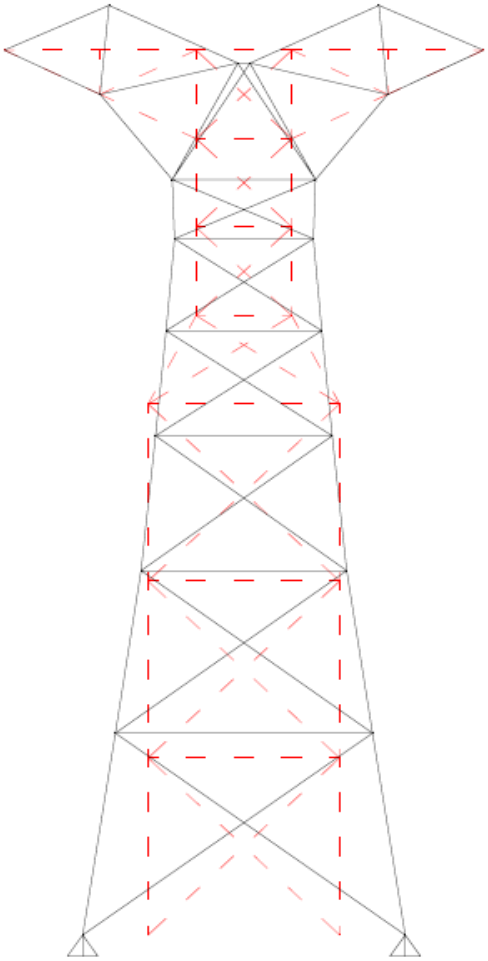

Figure 21. The optimum geometry of forty seven-bar truss

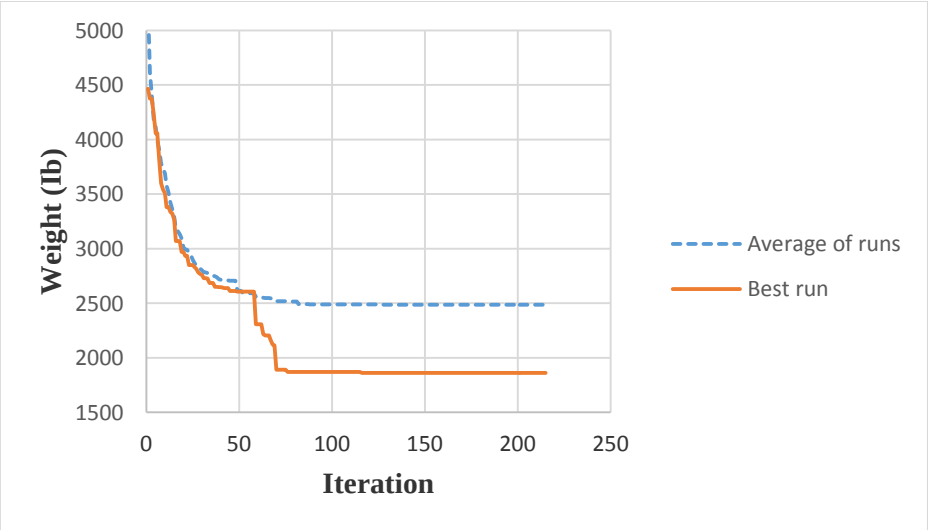

Figure 22. The weight convergence process of forty seven-bar truss

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

8. CONCLUSION

In this research, the combination of DNA computing algorithm and Generalized Convex Approximation (GCA) method has been used to optimize the size and geometry of truss structures under the constraints such as axial stresses of the elements, displacements in the degrees of freedom of truss nodes and slenderness ratios of the elements.

In this way, at the first stage by selecting appropriate and similar cross sections for truss elements, the coordinates of truss nodes as continuous variables have been optimized by Generalized Convex Approximation (GCA) method. Then, at the second stage with achieved coordinates from the first stage and by using DNA computing algorithm, the cross-section areas of the elements as discrete variables have been optimized.

According to the achieved results from numerical examples and comparing them with different references, it is concluded that the combination of DNA computing algorithm and Generalized Convex Approximation (GCA) method is an appropriate method to optimize truss structures.

REFERENCES

[1] Adleman LM. Molecular computation of solutions to combinatorial problems, *Sci* 1994; **266**(5187): 1021-4.
[2] Xu J, Qiang X, Yang Y, Wang B, Yang D, Luo L, Pan L, Wang Sh. An unenumerative DNA computing model for Vertex coloring problem, *IEEE Transact Nanobiosci* 2011; **10**(2): 94-8.
[3] Wang Z, Huang D, Meng H, Tang C. A new fast algorithm for solving the minimum spanning tree problem based on DNA molecules computation, *BioSyst* 2013; 114: 1-7.
[4] Henkel CV, Bäck T, Kok JN, Rozenberg G, Spaink HP. DNA computing of solutions to knapsack problems, *BioSyst* 2007; **88**: 156-62.
[5] Maazallahi R, Niknafs A, Arabkhedri P. A polynomial-time DNA computing solution for the N-Queens problem, *Proced Social Behav Sci* 2013; **83**: 622-8.
[6] Lee JY, Shin SY, Park TH, Zhang BT. Solving traveling salesman problems with DNA molecules encoding numerical values, *BioSyst* 2004; **78**: 39-47.
[7] Karakose M, Cigdem U. QPSO-based adaptive DNA computing algorithm, *Scient World J* 2013; **2013**: 8, Article ID 160687.
[8] Chickermane H, Gea HC. Structural optimization using a new local approximation method, *Int J Numer Meth Eng* 1996; **39**: 829-46.
[9] Rao SS. *Engineering Optimization: Theory and Practice*, 4th ed. Hoboken, New Jersey, John Wiley & Sons, Inc, 2009.
[10] Christensen WP, Klarbring A. *An Introduction to Structural Optimization*, Waterloo, Ontario, Canada: Department of Civil Engineering, University of Waterloo 2009; **153**.
[11] Shojaee S, Arjomand M, Khatibinia M. A hybrid algorithm for sizing and layout optimization of truss structures combining discrete PSO and convex approximation, *Int J Optim Civil Eng* 2013; **3**(1): 57-83.
[12] Vijay S. Is it possible to build computers from living cells? *J Bio Teach* 2004; **2**.

Downloaded from ijoce.iust.ac.ir at 16:42 IRST on Friday October 19th 2018

[13] Watada J. DNA computing and its application, *Stud in Computat Intellig (SCI)* 2008; **115**: 1065-89.
[14] Han A, Zhu D. DNA encoding methods in the field of DNA computing, *Stud Computat Intellig (SCI)* 2008; **94**: 293-322.
[15] Muhammad M.S, Ibrahim Z, Ueda S, Ono O, Khalid M. DNA computing for complex scheduling problem, *ICNC* 2005; *LNCS* 2005; **3611**: 1177-86.
[16] Rajeev S, Krishnamoorthy CS. Discrete optimization of structures using genetic algorithms, *J Struct Eng* 1992; **118**(5): 1233-50.
[17] Nayeem MA, Rahman Md. Kh, Rahman MS. Transit network design by genetic algorithm with elitism, *Transport Res Part C* 2014; **46**: 30-45.
[18] Holland JH. Adaptation *in Natural and Artificial Systems*, 2nd ed, MIT press, 1992.
[19] Ackley DH. A connectionist Machine for Genetic Hillclimbing, Kluwer, 1987.
[20] Wu SJ, Chow PT. Integrated discrete and configuration optimization of trusses using genetic algorithms, *Comput Struct* 1995; **55**: 695-702.
[21] Kaveh A, Kalatjari V. Size-geometry optimization of trusses by the force method and genetic algorithm, *Z Angew Math Mech* 2004; **84**: 347-57.
[22] Rahami H, Kaveh A, Gholipour Y. Sizing, geometry and topology optimization of trusses via force method and genetic algorithm, *Eng Struct* 2008; **30**: 2360-9.
[23] Rajeev S, Krishnamoorthy CS. Genetic algorithms-based methodologies for design optimization of trusses, *J Struct Eng ASCE* 1997; **123**(3): 350-8.
[24] Yang JP. Development of genetic algorithm-based approach for structural optimization, Ph.D. thesis. Singapore: Nanyang Technology University, 1996.
[25] Kang SL, Zong WG. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice, *Comput Meth Appl Mech Eng* 2005; **194**: 3902-33.
[26] Wu SJ, Chow PT. Steady-state genetic algorithm for discrete optimization of trusses, *Comput Struct* 1995; **56**(6): 979-91.
[27] Hwang SF, He RS. A hybrid real-parameter genetic algorithm for function optimization, *Adv Eng Inform* 2006; **20**: 7-21.
[28] Tang W, Tong L, Gu Y. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables, *Int J Numer Meth Eng* 2005; **62**: 1737-62.
[29] Wang D, Zhang WH, Jiang JS. Combined shape and sizing optimization of truss structures, *Comput Mech* 2002; **29**: 307-12.
[30] Hasancebi O, Erbatur F. Layout optimization of trusses using improved GA methodologies, *Acta Mech* 2001; 146: 87-107.
[31] Salsjegheh E, Vanderplaats GN. Optimum design of trusses with discrete sizing and shape variables, *Struct Optim* 1993; 6: 79-85.
[32] Hansen SR, Vanderplaats GN. Approximation method for configuration optimization of trusses, *AIAA J* 1990; **25**: 161-8.