INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J. Optim. Civil Eng., 2022; 12(3):279-312

DESIGN AND APPLICATION OF A HYBRID META-HEURISTIC OPTIMIZATION ALGORITHM BASED ON THE COMBINATION OF PSO, GSA, GWO AND CELLULAR AUTOMATION

F. Biabani, A. Razzazi, S. Shojaee[*, †] and S. Hamzehei-Javaran
Department of Civil Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

ABSTRACT

Presently, the introduction of intelligent models to optimize structural problems has become an important issue in civil engineering and almost all other fields of engineering. Optimization models in artificial intelligence have enabled us to provide powerful and practical solutions to structural optimization problems. In this study, a novel method for optimizing structures as well as solving structure-related problems is presented. The main purpose of this paper is to present an algorithm that addresses the major drawbacks of commonly-used algorithms including the Grey Wolf Optimization Algorithm (GWO), the Gravitational Search Algorithm (GSA), and the Particle Swarm Optimization Algorithm (PSO), and at the same time benefits from a high convergence rate. Also, another advantage of the proposed CGPGC algorithm is its considerable flexibility to solve a variety of optimization problems. To this end, we were inspired by the GSA law of gravity, the GWO's top three search factors, the PSO algorithm in calculating speed, and the cellular machine theory in the realm of population segmentation. The use of cellular neighborhood reduces the likelihood of getting caught in the local optimal trap and increases the rate of convergence to the global optimal point. Achieving reasonable results in mathematical functions (CEC 2005) and spatial structures (with a large number of variables) in comparison with those from GWO, GSA, PSO, and some other common heuristic algorithms shows an enhancement in the performance of the introduced method compared to the other ones.

Keywords: truss optimization; CGPGC; grey wolf optimizer; gravitational search algorithm; particle swarm optimization; objective function; CEC functions.

Received:25 February 2022; Accepted: 10 April 2022

---

[*]Corresponding author: School of Civil Engineering, Iran University of Science and Technology, P.O. Box 16846-13114, Iran
[†]E-mail address: saeed.shojaee@uk.ac.ir (S. Shojaee)

1. INTRODUCTION

Nowadays, with the advancement of engineering science, especially computer science and software, researchers and engineers are searching for fast, economical and at the same time, workable solutions with high efficiency for their products and structures. In modern times, optimization has been able to greatly help researchers and meet their needs [1]. The goal of optimization is to find a quick, short, and efficient way to achieve the best results. As mentioned, today optimization plays an important role in meeting human needs, especially its widespread use in civil engineering and, above all, in construction. The purpose of structural optimization is to minimize the weight of the structure and reduce costs. To minimize the weight of the structure, its cross-sectional area should be reduced. However, it is clear that the cross-section can be reduced to the extent that, firstly, the applied stress to the structure does not exceed the allowable stress, and, secondly, the displacement in the nodes of the structure does not exceed the allowable displacement. Optimization methods can be divided into two general categories: mathematical and meta-heuristic methods. Mathematical methods have a high convergence rate and accuracy. However, due to the use of gradient information and the complexity of their relationships, they need a suitable starting point. Plus, these methods require a lot of storage space; therefore, researchers are looking for ways to address these disadvantages [2].

Inspired by nature and its governing laws, which have always been a good teacher to mankind, researchers, in recent years, have introduced new methods, called meta-exploratory methods, in order to overcome the drawbacks of mathematical methods. Unlike these methods, meta-heuristic algorithms enable us to find the best general solution without the use of objective function derivatives and the need for a suitable initial value, with less complex formulas. Even though the answer of these methods cannot be considered the best absolute solution to the problem, satisfactory results can be obtained with a simpler process and a more computationally efficient approach compared to mathematical methods [3].

The optimization algorithms have also been widely used in a variety of civil and structural engineering applications. Among the researches conducted in the field of structural optimization, Shojaee and Darvishi [1] optimized the size and geometry of truss structures using the DNA calculation method and GCA generalized convex approximation method. In another study, Kaveh and Talatahari [4], Kaveh and Malakoutirad [5] and Shojaee et al. [6] performed optimization of truss structures by using the proposed hybrid method. The optimization of the size and geometry of truss structure using a new proposed method of combining optimized discrete particle swarm optimization (IDPSO) and asymmetric motion method (MMA) was conducted by Shojaee et al. [6]. Chaotic enhanced colliding bodies algorithms for size optimization of truss structures by Kaveh et al. [7] optimized the size of truss structures using a new proposed method (ECBO). Salajegheh et al. [8] used a combined method of particle swarm optimization (PSO) and gravitational search algorithm (GSA) by the first-order gradient method and a new optimization algorithm as GPSG for optimal design of structures by considering the frequency constraint. A hybrid algorithm based on the PSO particle swarm optimizer and cultural algorithm (CA) (PSOC) was used by Salajegheh et al. [9] for the optimal design of truss structures. Sizing and layout optimization of truss structures with artificial bee colony algorithm by Jawad et al. [10] used a swarm intelligence-based optimization method called the Bee Colony Algorithm (ABC) to carry out truss structures optimization. The aim was to optimize the geometry and size of truss structure members with limited displacement, stress, and buckling. The convolution neural network (CNN) models have also been used in optimal shear wall design in terms of geometry and topology by Pizarro et al. [11].

In the light of the previous studies, the main purpose of this paper is to present an algorithm that addresses the major drawbacks of the commonly-used algorithms including the Grey Wolf Optimization Algorithm (GWO), the Gravitational Search Algorithm (GSA), and the Particle Swarm Optimization Algorithm (PSO) and at the same time, benefits from a high convergence rate. The suggested method has high control on exploration and exploitation compared to PSO, GWO, and GSA. To this end, we drew the inspiration from the GSA Gravity Law feature, the top three GWO search factors, the PSO algorithm to calculate speed, and cell neighborhood to divide the population into smaller parts; the use of which has increased the rate of convergence in the method to the global optimal point. In this study, the proposed CGPGC algorithm is carefully examined and new insights are presented. The examples that are discussed here are divided into two types of benchmark functions (CEC2005) and optimal design of truss structures.

This paper is organized as follows: The typical PSO, GWO, GSA methods, and cellular automation theory are briefly presented in Section 2 and the fundamentals and basic ideas of the proposed method are reviewed in this section. In Section 3, the optimization problem of two types of benchmark functions (CEC2005) and optimal design of truss structures are generally discussed. The efficiency of the proposed method is confirmed through numerical examples provided in Section 4. Finally, the conclusions of this research are summarized in Section 5.

2. BASIC IDEAS

The CGPGC method is based on the hybridization of the PSO, GSA, cellular automation theory, and GWO methods. A unique attractive feature of the proposed method is that it uses smaller components called cell neighborhoods, instead of using the entire population for updates. Calculating the force on particles, using a cell neighborhood, increases the likelihood of searching for new points, reduces convergence to a local optimum, and also increases the overall convergence rate. Another unique attractive feature of the proposed method is the fact that the stable scheme is frequently used with limits adjusting any parameter and the method is highly capable of making a balance between exploration and exploitation. Achieving reasonable results in mathematical functions (CEC 2005) and Large-scale space structures with a high number of variables compared to the PSO, GSA, and GWO methods and some other known metaheuristic algorithms indicates an improvement in the performance of the method compared to other methods. In order to make the paper self-explanatory, prior to proposing the CGPGC optimization method, the characteristics of the PSO, GSA, cellular automation theory, and GWO methods are briefly explained in the following four sections.

2.1 Introduction to PSO

The basic steps of particle swarm optimization (PSO) are outlined here. PSO is an algorithm with a memory such that an individual's properties are simulated with a particle. These particles are referred to as swarm. Each particle of the swarm represents a potential solution of the optimization problem. The *i*-th particle in *t*-th iteration is associated with a position vector, $\boldsymbol{X}_i^t$, and a velocity vector, $\boldsymbol{V}_i^t$, shown as the following:

$$
\begin{aligned}
X_i^t &= \{ x_{i1}^t, x_{i2}^t, ..., x_{iD}^t \} \\
V_i^t &= \{ v_{i1}^t, v_{i2}^t, ..., v_{iD}^t \}
\end{aligned}
\tag{1}
$$

The initial particles are randomly generated from the uniform distribution. The steps of the PSO algorithm are then applied separately to the initial population, and the velocity and position of the particles are updated according to Eqs. (2) and (3).

$$V_i^{t+1} = w^t V_i^t + c_1 r_1 (pbest_i^t - X_i^t) + c_2 r_2 (gbest^t - X_i^t) \tag{2}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{3}$$

where r_1 and r_2 are two uniform random sequences generated from the interval [0, 1]; c_1 and c_2 are the cognitive and social scaling parameters, respectively and ω^t is the inertia weight that controls the influence of the previous velocity. At each step, the velocity and position are checked to ensure if their parameters are within the predefined limits. The next step is to determine and update the global best position (gbest) and personal best position (pbest). At this step, a new population will be created provided that the termination criteria of the PSO algorithm are satisfied; otherwise, updating the velocity and position of particles will continue.

The [10], proposed that the cognitive and social scaling parameters c_1 and c_2 should be selected as $c_1=c_2=2$ to allow the product c_1r_1 or c_2r_2 to have a mean of 1. The performance of PSO is very sensitive to the inertia weight (ω) parameter, which may decrease with the number of iterations as follows:

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} . t \tag{4}$$

where $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of ω , respectively; and t_{max} is the maximum number of optimization iteration.

One of the advantages of this algorithm is its memory, which means that in each step, the information of the previous steps is stored. Compared to the other evolutionary algorithms based on heuristics, the weaknesses of this algorithm consist of adjusting and implementing parameters and difficulties to control the balance between exploration and exploitation. In the exploitation phase, the PSO algorithm may end up in premature convergence and in the exploration case, there is a chance to face time delay during the convergence when searching for an optimum. It can be anticipated that the algorithm may perform well in one specific problem but may not perform at all in a slightly varied one. In addition, in the process of using the algorithm, different strength levels of social and cognitive behavior would potentially have an influence on the success model used as well. In other words, the access to get a better answer would be justifiable in, for instance, unimodal problems because there is more likelihood to find better solutions, whereas in multimodal problems, the same access of individual particles is contractionary. Accordingly, the parameters used to control the behavior of the algorithm would strongly depend on the type of the problem surface and the actual situation the algorithm is facing in an iteration.

2.2 Introduction to GSA

The gravity search algorithm (GSA) contains a set of particles randomly assigned in the design space and updated over time [13]. The steps for the GS algorithm are summarized as follows:

Step1. The position of each particle is defined as Eq. (5). In summary, the acceleration vector of the *i*-th particle in iteration t can be specified as [13]:

$$a_i^t = G(t) \sum_{j=1}^{n} \left[rand_j \frac{M_j(t)}{R_{ij}(t)+\varepsilon} (X_j^t - X_i^t) \right] \tag{5}$$

where, G is the gravitational constant. M_j represents the mass of the *j*-th particle and can be evaluated according to the objective functions of the particles [13]. R_{ij} is the distance between the particles *i* and *j*. *rand*$rand_j$ is a random scalar with uniform distribution in the interval of zero and one and ε is a small number to prevent numerical errors. The gravitational constant G_0 is defined as [13]:

$$G(t) = G_0 \times e^{-\alpha \frac{t}{T}} \tag{6}$$

where the parameters G_0 and α are two constant coefficients. t represents the current iteration and T is the maximum number of iterations. The updated mass of the particles is evaluated by Eq. (7) and normalized according to Eq. (8);

$$m_i(t) = \frac{F_i(t) - worst(t)}{best(t) - worst(t)} \tag{7}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{NP} m_j(t)} \tag{8}$$

in which,

$$best(t) = \min F_i(t) \qquad for \quad i = 1 : NP \tag{9}$$

$$worst(t) = \max F_i(t) \qquad for \quad i = 1 : NP \tag{10}$$

and the fitness value (objective function) of the *i*-th particles is evaluated as $F_i(t)$ at time *t*.

Step 2. The updated gravitational velocity vector is calculated using Eq. (11),

$$V_i^{t+1} = rand_i \times V_i^t + a_i^t \tag{11}$$

Step 3. The position of the particles is updated according to Eq. (12),

$$x_i^t(t+1) = x_i^t(t) + v_i^t(t+1) \tag{12}$$

Similar to the PSO algorithm, there are two major issues associated with the search performance of GSA. In the exploitation phase, the GSA may end up in premature convergence due to rapid reduction in diversity. As well as that, more ineffective iterations are needed to get a more accurate estimation of the local optima. In addition, it is difficult to have a good balance between exploration and exploitation, so the parameters used to control the behavior of the algorithm would strongly depend on the type of the problem surface and the actual situation the algorithm is facing in an iteration.

2.3 Introduction to GWO

The Grey Wolf Optimizer (GWO), which was introduced by Mirjalili et al. [14], is a population-based algorithm inspired by the social and hierarchical behavior of wolves in the hunting mechanism. In this algorithm, four types of grey wolves, including alpha, beta, delta, and omega, are considered to simulate the hierarchy of leadership. These four wolf groups take three main steps: observing and pursuing the prey, approaching and encircling it and ultimately, hunting attacks. The simplicity and broad ability of solving large-scale problems are two outstanding features of this algorithm.

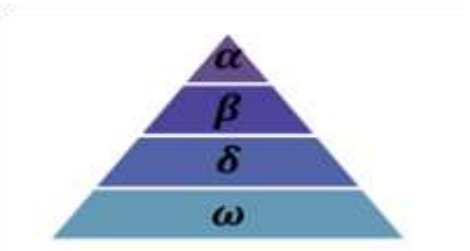


Figure 1. The social hierarchy of a pack of grey wolves

One of the most impressive aspects of the wolves' social life is their hierarchy structure at group hunting. The Grey Wolf Optimizer (GWO) [14] is inspired by the leadership hierarchy and hunting behavior of wolves. Social hierarchy, tracking (exploration), encircling, and attacking (exploitation) are the main four sections of the GWO algorithm. Herein, all wolves are categorized as alpha (*α*), beta (*β*), delta (*δ*), and omega (*ω*), which respectively represent the first, second, third, and other remaining candidate solutions. The status of each of them in the hierarchy are illustrated in Fig. 1. Assuming that alpha, beta, and delta have better knowledge about the potential location of the prey and the other wolves follow them, the following equations show the updated position of each wolf at (*t* + 1)-th iteration:

$$\boldsymbol{X}_1(t) = \boldsymbol{X}_\alpha(t) - \boldsymbol{A}_1.(\boldsymbol{D}_a) \tag{13}$$

$$\boldsymbol{X}_2(t) = \boldsymbol{X}_\beta(t) - \boldsymbol{A}_2.(\boldsymbol{D}_\beta) \tag{14}$$

$$\boldsymbol{X}_3(t) = \boldsymbol{X}_\delta(t) - \boldsymbol{A}_3.(\boldsymbol{D}_\delta) \tag{15}$$

$$\boldsymbol{A}_i = 2.\boldsymbol{a}.\boldsymbol{r}_{2i} - \boldsymbol{a} \qquad i = 1, 2, 3 \tag{16}$$

where $X_{t\alpha}$, $X_{t\beta}$, *and* $X_{t\delta}$ are the positions of the best three solutions. $\mathbf{A}_i$ is a random vector in the interval [−2a, 2a], where ***a*** is linearly decreased from 2 to 0 over the cycles of optimization. The D vectors mathematically simulate the encircling behavior through the following equations:

$$\boldsymbol{X}(t+1) = \frac{\boldsymbol{X}_1 + \boldsymbol{X}_2 + \boldsymbol{X}_3}{3} \tag{17}$$

$$\boldsymbol{D}_\alpha = \left|\boldsymbol{C}_1.\boldsymbol{X}_\alpha(t) - \boldsymbol{X}(t)\right| \tag{18}$$

$$\boldsymbol{D}_\delta = \left|\boldsymbol{C}_3.\boldsymbol{X}_\delta(t) - \boldsymbol{X}(t)\right| \tag{19}$$

$$\boldsymbol{D}_\beta = \left|\boldsymbol{C}_2.\boldsymbol{X}_\beta(t) - \boldsymbol{X}(t)\right| \tag{20}$$

$$\boldsymbol{C}_1 = 2.\boldsymbol{r}_{1i} \qquad i = 1, 2, 3 \tag{21}$$

By mimicking the social leadership and hunting behavior of grey wolves in nature, GWO performs the search in a problem's landscape with a distinct characteristic of balancing exploration and exploitation, and has shown very competitive results compared to the other well-known meta-heuristic algorithms. Nevertheless, compared to some state-of-the-art optimization algorithms, GWO is usually outperformed due to its oversimplified search dynamics. However, GWO has poor exploration capability and suffers from local optima stagnation. Furthermore, due to its imperfect search structure and possible risk of being trapped in local optima, its application has been limited.

2.4 Theory of cellular automation (CA)

The theory of cellular automation was first introduced in 1950 by Wolfram, Ulam and Von Neumann ([15-17]), who initially sought to obtain a graphical network with simple rules. Cellular automation is a dynamic system that is discrete in time and space and benefits from good convergence. The use of the cell automation method requires the definition of parts called cells, which form a cellular network together, describing the problem, in which each cell has properties called cell state. Different parameters can be introduced for cell states. These simple rules, which are applied to find the new state of each cell, are called transfer laws or local laws. Neighborhood is another major component of the cellular automation method, and the number of cells participating in the definition of local laws is called the neighborhood number [17]. Adjacent cells of each cell that interact with each other are called cell neighborhoods. A neighboring cell affects itself at each stage according to local laws [18] (Fig. 1). Fig. 2 shows the general process of the cellular method. In the cellular automata method, cells are defined as neighborhoods whose central cell state affects their cellular state in the next time step (Moore's neighborhood is used in this study (Fig. 2)). Therefore, a different arrangement for the neighborhood type of cells can be achieved [19]. Fig. 3 shows two models of standard neighborhoods proposed in different articles for 2D cellular network. In Van Neumann's neighborhood, each cell will be considered as a neighborhood with the other four cells around it in four main directions. In Moore's neighborhood, each cell, in addition to the four main directions, will also be associated with cells in its sub-directions. It forms this neighborhood with a total of 8 cells around it.

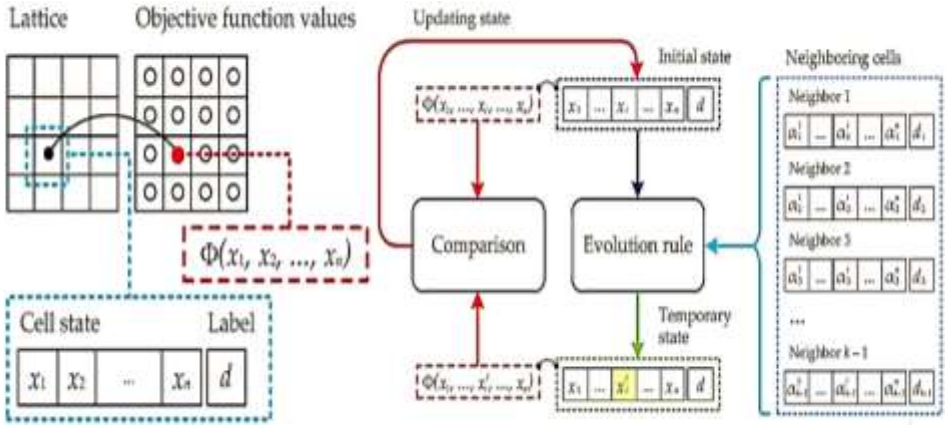


Figure 2. How the cellular method works [16]

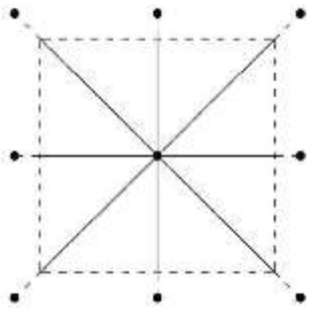
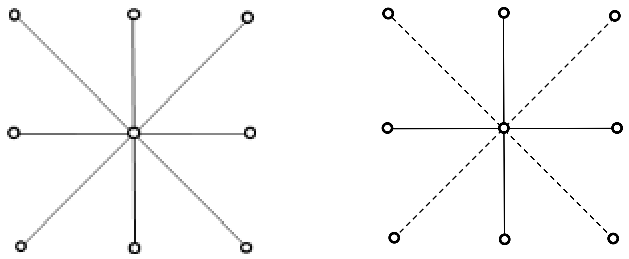


Figure 3. How the cells are selected


a) Vennie Neighborhood b) Moore Neighborhood
Figure 4. Types of Standard Neighborhoods

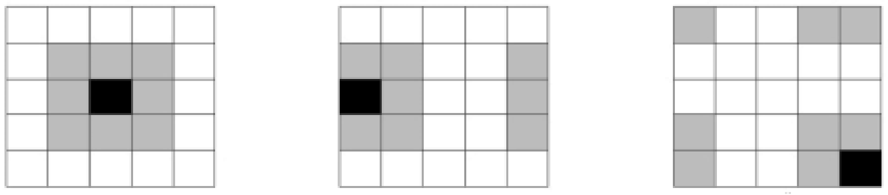


Figure 5. Neighborhood selection based on the selected cell

For a better explanation of the neighborhood, we can refer to Fig. 5, in which the neighboring cells show a desired cell, with the selected cell in the middle or corner of the cellular automation network. Fig. 3 shows how a neighborhood is selected, and in Fig. 5, the selected cell is shown in black and its neighborhood is shown in grey.

2.5 The fundamentals of CGPGC

In this section, we combine the three methods of GSA, PSO, and GWO with the theory of cellular automation. The purpose of combining these methods is to provide an efficient method that has a high rate of convergence to the global optimal point and does not fall into the optimal local trap. Different algorithms have their own strengths and weaknesses, so to achieve better results, it is necessary to combine algorithms or use computational solutions. Therefore, in order to increase the efficiency of the mentioned algorithms, the GSA algorithm has been used as the basic method and the strengths of the PSO and GWO algorithms have been taken advantage of on the side. As well as that, the theory of cellular automation has been used. The GSA algorithm is known as an algorithm based on collective intelligence, which seeks the optimal response in a vector and multidimensional space. In this algorithm, particles make systematic and classic movements in their gravitational field in proportion to their mass, and the force that the particles exert on each other acts as a communication signal and determines the position of each particle. Thus, the particles make their next move in the search space intelligently. In this algorithm, for each mass, one considers an active gravitational mass, which measures the intensity of the gravitational force around an object, as well as a passive gravitational mass, which is a measure of the intensity of the force interacting. This work creates a parameter-less structure variant, which means it is virtually independent of the underlying examined problem type. To this end, we drew inspiration from the GSA gravity law feature, accompanied by the top three GWO search factors, the PSO algorithm to calculate speed, and also the theory of cellular automation.

In order to calculate F_{ij}^{d} (*t*), which represents the force applied from mass *i* to mass *j* at time *t* and dimension *d*, one can employ:

$$F_{ij}^{d}(t) = G(t)\frac{M_{pi}(t)\times M_{aj}(t)}{R_{ij}(t)^{Rpower}+\varepsilon}(x_{j}^{d}(t)-x_{i}^{d}(t)) \tag{22}$$

In this relation and at time *t*, *Rpower* is a fixed number, which is equal to 0.1, and *ε* is a very small number, *G*(*t*) is the constant of gravity, $M_{pi}(t)$ is the passive gravitational mass *i* and $M_{aj}(t)$ is the active gravitational mass *j* and finally, R_{ij} is the Euclidean distance between the two masses of the GSA algorithm, which is known as an algorithm based on collective intelligence and seeks the optimal response in a vector and multidimensional space.

$$R_{ij}(t) = \left\| x_{i}(t) \quad . \quad x_{j}(t) \right\|_{2} \tag{23}$$

The coefficient *G*(*t*) can be written as follows:

$$G(t) = \ln\left(\frac{iter}{\max - iter}\right) \tag{24}$$

In this regard, *iter* represents the current number of iteration and *max-iter* represents the maximum number of iterations. Using this coefficient means that there is no need to adjust the constant coefficients required in the formula *G*(*t*) of the GSA algorithm, which adds another advantage to the proposed algorithm. Therefore, in order to calculate all of the forces acting on the mass *i* at time *t* and at dimension *d*, and considering a random coefficient in the interval [0,1], we can write:

$$F_{i}^{d}(t) = \Sigma_{j=1}^{N} rand_{j} \quad F_{ij}^{d}(t) \tag{25}$$

However, in order to improve the discovery power of the algorithm, only the set containing top members is allowed to influence other members.

$$F_i^d(t) = \sum_{j \in nbest, j \neq i} rand_j \; F_{ij}^d(t) \tag{26}$$

in which, *nbest* is calculated according to the following equation:

$$nbest = np(2 + (1 - iter / max - iter) * (Cp - 2)) / 100 \tag{27}$$

in which, *Cp* is a fixed number and *np* is the number of particles. Once this step is taken, the acceleration of objects in dimension *d* can be determined. According to the Newton's second law, the acceleration of any object is equal to:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)} \tag{28}$$

In this case, M_{ii} (*t*) is the inertial mass of the *i*-th particle. It should be noted that stochastic coefficients are used in thet relations to maintain the random nature of particle motion in the search space. The following equations are used to calculate the objects in this algorithm:

$$M_{ii}(t) = M_{pi}(t) = M_{ai}(t) = M_i(t) \quad , \quad i = 1, 2, ..., N \tag{29}$$

$$m_i(t) = \frac{value_i(t) - worst(t)}{best(t) - worst(t)} \tag{30}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{31}$$

in which, $value_i$ (*t*) indicates the suitability of the particle *i* at time *t*, and also *best*(*t*) and *worst*(*t*) indicate the suitability of the best and worst particles among the particle set. Given that the selection of *best*(*t*) and *worst*(*t*) is performed based on the objective function of the whole population, the theory of cellular automation has been used to improve the process, and results in increasing the rate of convergence, and also not falling into the optimal local trap. In the proposed method, Moore's neighborhood (Figs. 4 and 5) is used and the best and worst values are calculated by comparing the objective function of each cell (particle) and its eight neighboring cells. The quasi-mass calculation code based on the cellular method is as follows:

Pseudocode 1 calculates the objects by the cellular method

```
Placing particles in a square cellular network
Calling neighboring particles of the desired particle from the cellular network
Choosing the best and worst in the cellular neighborhood
Calculate cell mass based on the best and worst cells
```

According to the above-mentioned points, in order to increase the efficiency of the algorithm, the velocity is calculated in three steps, which are inspired by the third step of the PSO algorithm. In the first step, the velocity is calculated from the sum of velocities in the previous step and the force of gravity according to the following equation:

$$v_i^d(t+1) = rand \times v_i(t) + a_i(t) \tag{32}$$

and then, using the velocity calculated in the first step, the velocity in the second step is updated according to the following equation:

$$v_i(t+1) = rand \times v_i(t) + C_k \times a(t) + (2 - C_k) \times x_{mean-gbest}^d - x_i(t) \tag{33}$$

The coefficient C_k is calculated by the following equation:

$$C_k = 2 - 0.25 \times \log \frac{ncn}{t} \tag{34}$$

Also, the initial value of *ncn* is considered equal to 1 and is added to the initial population at any time. In the final step, using the velocity calculated in the previous step, while employing the PSO method, we have:

$$V_i^d(t+1)_{IGSA} = \beta V_i^d(t)_{GSA} + C_1 \times \varphi_1 \times (x_{pbest_i}^d - x_i^d(t)) + C_2 \times \varphi_2 \times (x_{gbest}^d - x_i^d(t)) \tag{35}$$

in which, φ_1 and φ_2 are random variables in the range [0,1] and C_1 and C_2 are fixed coefficients. In addition, since the GWO algorithm considers the effect of top 3 particles to find the best solution, $x^d_{mean\text{-}gbest}$ is used instead of x^d_{gbest} in the PSO formula, so as to be used in the proposed algorithm for the velocity.

$$X_{mean-gbest}^d = (X_{alpha}^d + X_{beta}^d + X_{delta}^d)/3 \tag{36}$$

where, x^d_{alpha}, x^d_{beta} and x^d_{delta} represent the position of the top 3 particles in the algorithm. To calculate the new position of each particle, it is possible to write the sum of the calculated values using vector summation.

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{37}$$

To remove the worst particle and replace it with a new or an improved one, it is necessary to generate a random value after identifying the worst mass called the gamma particle. If the random value generated is less than a computational value dependent on the current iteration and the maximum number of iterations, the gamma particle value is replaced with a new value. Otherwise, it is replaced with the average x^d_{alpha}, x^d_{beta} and x^d_{delta} particles. The pseudocode of the deletion process is as follows:

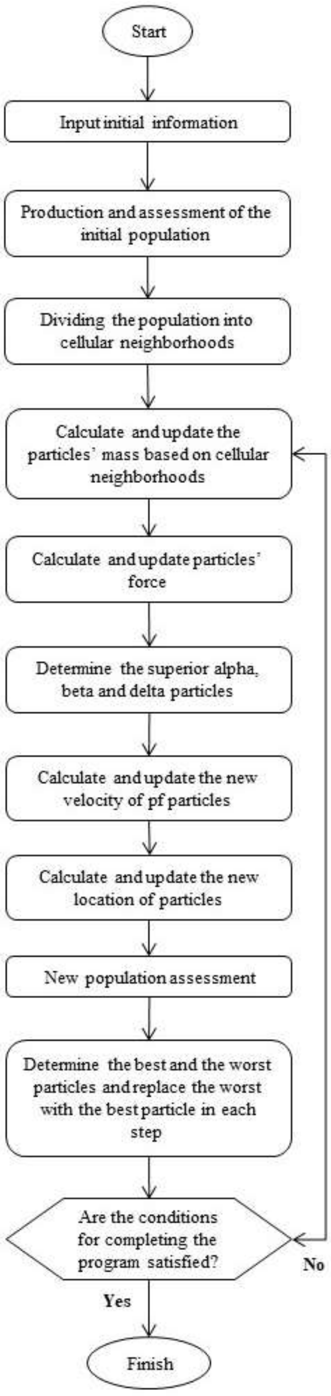

Start
Input initial information
Production and assessment of the initial population
Dividing the population into cellular neighborhoods
Calculate and update the particles' mass based on cellular neighborhoods
Calculate and update particles' force
Determine the superior alpha, beta and delta particles
Calculate and update the new velocity of pf particles
Calculate and update the new location of particles
New population assessment
Determine the best and the worst particles and replace the worst with the best particle in each step
Are the conditions for completing the program satisfied?
No
Yes
Finish

Figure 6. The flowchart of the CGPGC algorithm

Pseudocode 2 calculates the worst elimination method (worst elimination)

```
Create a random number
**If** (maximum iteration) / (iteration number – 1) > random number **then**
Create new position for particle = *gamma*
**Otherwise**
$(x^d_{alpha} + x^d_{beta} + x^d_{delta}) / 3 = gamma$
**End**
```

3. PROBLEM FORMULATION

The examples, which are discussed here, are divided into two types of benchmark functions (CEC2005) and optimal design of truss structures. These problems have been chosen to demonstrate the reliability and capability of the presented method.

3.1 Benchmark functions: CEC2005

To evaluate the performance of the proposed method in a more systematic manner, numerous examples of unconstrained problems of mathematical unimodal, multimodal, expanded and hybrid composition functions of CEC2005 are cited [20].

These functions play an important role in the development of the proposed search algorithms as well as in the assessment of algorithmic ideas. In this evaluation, the efficiency of the proposed method is confirmed by specifying a common termination criterion, size and scalability of problems, initialization scheme and linkages. To show the behavior of these functions, some sketches of them are demonstrated in figs. 2, 3 and 4. An unconstrained optimization problem can be formulated in the following form:

$$\begin{aligned}&\text{Minimize} \quad f(X)\\ &X = \{x_1, x_2, ..., x_j, ..., x_n\} \in R^d\end{aligned} \tag{38}$$

where $f(X)$ represents the objective function and n is the number of variables. A given set of values is expressed by R^d, where the design variables x_j can take values only from this set.

3.2 Optimal design of truss structures

A structural optimization problem can be formulated in the following form:

$$\begin{aligned}&\text{Minimize} \quad f(X)\\ &\text{Subject to} \quad g_i(X) \leq 0 \qquad i = 1, 2, ..., m\\ &X = \{x_1, x_2, ..., x_j, ..., x_n\} \hat{I} R^d\end{aligned} \tag{39}$$

where, $g(X)$ is the behavioral constraint and m is the number of constraints. In size optimization problems, the main aim is usually to minimize the weight of the truss structures under design constraints. The design variables are chosen to be the cross-section areas of the elements, which are usually selected from a set of discrete values. Therefore, the optimization problem can be reformulated in the following form:

$$\begin{aligned}
&\text{Minimize} \quad f(\mathrm{X}) = W(\boldsymbol{A}_i) = \sum_{i=1}^{Ne} \rho_i A_i L_i \\
&\text{Subject to} \quad g_{Si} = \frac{\sigma_i(A_i)}{\sigma_{all}} - 1 \le 0 \qquad i = 1,2,\ldots,N_e \\
&\qquad g_{Di} = \frac{\Delta_j(A_i)}{\Delta_{all}} - 1 \le 0 \quad j = 1,2,\ldots,N_n \\
&\qquad \boldsymbol{A}_i \in \boldsymbol{A}_e = \{A_{e1}, A_{e2}, \ldots, A_{ep}\}
\end{aligned} \tag{40}$$

where W is the structural weight, ρ_i, A_i and L_i represent the material density, cross-section area and length of the i-th element, respectively, σ_i and σ_{all} are the stress in the i-th element and the allowable axial stress, Δ_j and Δ_{all} denote the displacement of the j-th node and the allowable displacement, N_e and N_n stand for the number of elements and nodes in the structure, and finally, A_e is the available profile list. A number of constraint-handling techniques have been proposed to solve constrained optimization problems. In this study, the penalty function is used to deal with the constrained search spaces as:

$$\tilde{f}(X) = \begin{cases} f(X) & \text{if } X \in R^d \\ f(X) + \sum_i \max(g_i(\mathrm{X}), 0.0) & \text{otherwise} \end{cases} \tag{41}$$

in which, $f_s(X)$ is a modified function. In addition, R^d denotes the feasible search space.

4. NUMERICAL PROBLEMS

In this research, the proposed method is applied in 10 well-known CEC_2005 benchmark functions [20] to test the exploration, exploitation, local optima trap avoidance, and convergence properties. Furthermore, the new method has been used to minimize the weight of large-scale structures with a high number of variables of 942 members and 72 members in two cases considered as real-life engineering applications. All of the computations have been performed using the MATLAB software in the Microsoft Windows 10 environment using common configurations. The stiffness method is applied to analyze the structures. The results of the new method are compared with those of different references. The tuning parameters of the new algorithm are summarized in Table 1.

Table 1: The CGPGC algorithm parameters

Parameter	Description	Value
R power	power of R coefficient	0.1
W	initial weight	0.9
C_1 , C_2	learning coefficient	2
Number of Runs	_	20
Iteration	_	100, 500
Number of Variables	_	10, 30
Number of Particles	–	25
Number of Analyses	Iteration × Number of Particle	2500, 12500
Cp	Constant value	100

4.1 Benchmark functions: CEC2005

In order to show the efficiency and robustness of the proposed method, it has been tested on 9 CEC_2005 benchmark functions shown in Table 2. The search bound denotes the upper and lower bounds of the search space and the minimum answer is the best optimal answer. According to the CEC_2005 functions' characteristics, they are divided into 4 types, which are U: Unimodal, BM: Basic Multimodal, EM: Expanded Multimodal, and HCM: Hybrid Composition Multimodal.

Table 2: Information about CEC functions [18]

Function	Type	Initialization Range	Search Bound	Minimum Answer
C1: Shifted Sphere Function	U	[-100,100]	[-100,100]	-450
C5: Schwefel's Problem 2.6 with Global Optimum on Bounds	U	[-100,100]	[-100,100]	-310
C6: Shifted Rosenbrock's Function	BM	[-100,100]	[-100,100]	390
C8: Shifted Rotated Ackley's Function with Global Optimum on Bounds	BM	[-32,32]	[-32,32]	-140
C11: Shifted Rotated Weierstrass Function	BM	[-0.5,0.5]	[-0.5,0.5]	90
C13: Expanded Extended Griewank's plus Rosenbrock's Function (CEC8CEC2)	EM	[-3,1]	[-3,1]	-130
C14: Shifted Rotated Expanded Scaffer's CEC6	EM	[-100,100]	[-100,100]	-300

C16: Rotated Hybrid Composition Function	HCM	[-5,5]	[-5,5]	120
C17: Rotated Hybrid Composition Function with Noise in Fitness	HCM	[-5,5]	[-5,5]	120
C25: Rotated Hybrid Composition Function without Bounds	HCM	[-2,5]	[-5,5]	260

Due to the randomness of the metaheuristic methods, each of these functions was run 20 times independently with a variable number of 10 and 30 with an iteration number of 100 and 500. The results obtained are the values of average, median, best solution, average iteration, average time, standard deviation, and violation, which are given in Table 3. The results of studying the performance of CGPGC algorithm in 2005 standard functions showed that this algorithm performs very well in solving these problems. According to the results of Table 3, this algorithm has a powerful performance in solving these functions in 92% of the cases, which is shown in Figs. (7-13).

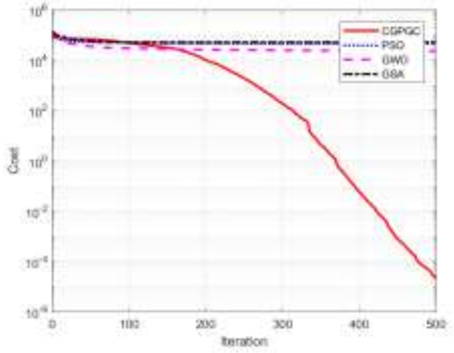
Table 3: CEC 2005 Results

		D = 10 & T = 100				D = 30 & T = 100				D = 30 & T = 500			
	Answer	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA
CEC1	Average	0.036	5504.63	1410.795	29624.489	209.675	44881.102	43240.154	94863.487	2.0631e-05	46734.113	23683.789	51930.193
	Median	0.036	5504.613	1410.795	29624.489	209.675	44881.102	43240.154	94863.487	9.0528e-06	43749.807	22052.025	48316.120
	Best	0.017	3687.172	1198.790	27945.023	192.736	22912.039	34678.503	78285.778	1.0012e-06	11248.407	9497.558	38766.569
	Avg itera	100	100	99.5	26.5	99	100	98	14	500	134.9	498.15	138.25
	Avg time	0.930	0.077	0.101	0.068	0.882	0.087	0.099	0.058	3.436	0.210	0.323	0.338
	std	0.027	2570.250	299.820	2375.123	23.956	31068.947	12108.003	23444.421	3.1601e-05	19956.291	8068.587	10731.614
	Violation	450.017	4137.172	2072.801	31753.955	642.736	23362.039	52251.805	111891.197	450	52477.099	28225.822	44778.662
CEC5	Average	22.555	12459.963	7190.737	17865.686	8191.240	31801.768	24825.995	60695.107	4543.813	29975.131	19040.839	35605.57
	Median	22.555	12459.963	7190.737	17865.686	8191.240	31801.768	24825.995	60695.107	4273.878	29655.829	19813.351	33795.141
	Best	8.417	9865.571	4410.343	17299.103	5856.475	25858.112	22794.991	55865.429	2722.142	13259.1881	12117.287	27240.375
	Avg iter	99	100	98.5	18	96	100	97	4	494.15	228.45	497.7	120.15
	Avg time	1.146	0.123	0.153	0.083	1.125	0.129	0.177	0.070	3.705	0.300	0.544	0.355
	std	19.993	3669.024	3932.070	801.270	3301.856	8405.599	2872.274	6830.196	1443.460	7955.253	3636.261	6120.747
	Violation	346.693	10175.571	4720.343	17609.103	6166.475	26168.112	27167.000	56175.429	6503.088	27038.036	17182.387	40207.534
CEC6	Average	15.632	79504363.574	1169125577.583	4690992688.765	3219433.308	102671583420.524	18813926877.987	96143990639.40	5321.185	37298273736.321	5117057749.999	25256049988.193
	Median	15.632	79504363.574	1169125577.583	4690992688.765	3219433.308	102671583420.524	18813926877.987	96143990639.402	2632.466	33612808160.013	4925264829.654	21727029408.908
	Best	10.306	1431454.799	6980884.206	2942831561.711	2350380.249	24402383810.560	15280457996.53	91875141882.521	23.5939	7911130209.636	410154478.7926	7231428892.2767
	Avg iter	99	100	99	27	100	3	97	63.5	499.1	147.55	497.3	142.35
	Avg time	1.041	0.081	0.098	0.075	0.676	0.058	0.089	0.056	3.0572	0.24796	0.28362	0.29479
	std	7.532	110411766.443	1643520786.813	2472273175.092	1229026.62	110689363604.498	4997079614,380	6037063807,699	5639.382	28046108101,5731	2967043878,153	16795413847,806
	Violation	369.041	1431064.799	6980494.206	6439153425.819	4088096.367	24402383420,560	22347395369.444	91875141492.521	307.855	9332284073.852	8140978081,529	62021539383,929

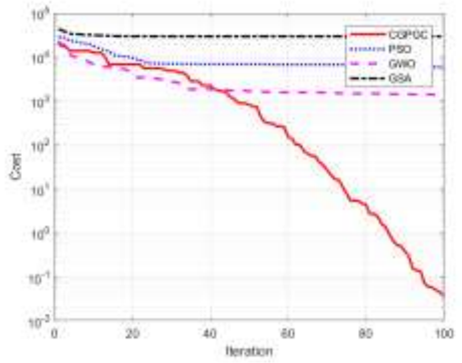
		D = 10 & T = 100				D = 30 & T = 100				D = 30 & T = 500			
	Answer	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA
CEC8	Average	20.464	20.884	20.631	20.965	21.204	21.252	21.238	21.247	21.068	21.153	21.178	21.208
	Median	20.464	20.884	20.631	20.965	21.204	21.252	21.238	21.247	21.065	21.173	21.186	21.206
	Best	20.383	20.797	20.626	20.957	21.176	21.238	21.214	21.229	20.926	20.890	20.954	20.980
	Avg iter	49.5	15	38	13	34	94	30	7	80.3	28.7	56.45	23
	Avg time	0.800	0.089	0.094	0.075	0.767	0.073	0.123	0.052	4.075	0.267	0.328	0.284
	std	0.113	0.123	0.006	0.010	0.039	0.019	0.033	0.025	0.061	0.078	0.071	0.091
	Violation	160.383	160.797	160.626	160.957	161.176	161.266	161.214	161.265	161.055	161.166	161.170	161.316
CEC11	Average	22.555	12459.963	7190.737	17865.686	8191.240	31801.768	24825.995	60695.107	4543.813	29975.131	19040.839	35605.57
	Median	22.555	12459.963	7190.737	17865.686	8191.240	31801.768	24825.995	60695.107	4273.878	29655.829	19813.351	33795.141
	Best	8.417	9865.571	4410.343	17299.103	5856.475	25858.112	22794.991	55865.429	2722.142	13259.1881	12117.287	27240.375
	Avg iter	99	100	98.5	18	96	100	97	4	494.15	228.45	497.7	120.15
	Avg time	1.146	0.123	0.153	0.083	1.125	0.129	0.177	0.070	3.705	0.300	0.544	0.355
	std	19.993	3669.024	3932.070	801.270	3301.856	8405.599	2872.274	6830.196	1443.460	7955.253	3636.261	6120.747
	Violation	346.693	10175.571	4720.343	17609.103	6166.475	26168.112	27167.000	56175.429	6503.088	27038.036	17182.387	40207.534
CEC13	Average	10.872	13.606	9.474	12.286	43.394	40.672	31.018	43.160	40.873	37.398	27.002	41.684
	Median	10.872	13.606	9.474	12.286	43.394	40.672	31.018	43.160	42.294	37.576	27.698	40.538
	Best	10.646	13.021	9.401	10.599	43.320	35.296	29.879	37.868	28.640	30.145	19.828	35.405
	Avg iter	82.5	71	96.5	19	43	60	99	4	84.05	300.3	497.25	89
	Avg time	0.852	0.087	0.128	0.093	1.006	0.101	0.131	0.124	5.685	0.614	0.686	0.668
	std	0.319	0.827	0.102	2.386	0.105	7.602	1.611	7.484	4.087	4.120	3.229	3.567
	Violation	-78.901	-76.978	-80.598	-76.026	-46.530	-54.703	-57.842	-52.131	-46.717	-46.174	-57.686	-49.198

		D = 10 & T = 100				D = 30 & T = 100				D = 30 & T = 500			
	Answer	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA	CGPGC	PSO	GWO	GSA
CEC14	Average	4.005	4.364	4.223	4.722	14.110	14.329	13.404	14.377	13.72	13.927	13.367	14.239
	Median	4.005	4.364	4.223	4.722	14.110	14.329	13.404	14.377	13.763	13.889	13.399	14.276
	Best	3.834	4.098	3.938	4.655	14.091	14.316	13.314	14.339	13.320	13.357	12.521	13.608
	Avg iter	79	90	94	3	40	73	88	55.5	120.2	162.15	263.45	5.1
	Avg time	0.726	0.071	0.068	0.064	0.816	0.087	0.114	0.078	4.840	0.484	0.582	0.510
	std	0.242	0.375	0.402	0.095	0.026	0.018	0.127	0.053	0.197	0.245	0.378	0.328
	Violation	304.176	304.098	303.938	304.790	314.128	314.342	313.314	314.414	313.320	314.117	312.977	314.544
CEC16	Average	214.613	451.717	358.880	454.712	398.844	951.520	946.079	794.069	331.158	852.430	602.297	653.225
	Median	214.613	451.717	358.880	454.712	398.844	951.520	946.079	794.069	310.109	831.159	600.888	658.705
	Best	195.432	234.462	327.311	344.278	362.055	880.364	811.708	764.971	203.109	424.598	242.127	389.079
	Avg iter	77	100	66.5	32	99	81	98.5	23	370.1	234.5	497.55	85.9
	Avg time	8.256	1.56	1.556	1.536	8.734	1.660	1.757	1.669	45.4887	8.5987	8.7258	8.682
	std	27.125	307.245	44.644	156.176	52.028	100.629	190.029	41.150	102.7761	256.5592	205.1345	149.6181
	Violation	113.794	114.462	207.311	224.278	242.055	902.676	691.708	703.166	108.6328	681.2848	617.5058	511.5766
CEC25	Average	1238.54	766.005	996.900	1638.164	984.718	1570.618	1427.009	2053.230	487.254	1508.061	1331.667	1920.554
	Median	1238.54	766.005	996.900	1638.164	984.718	1570.618	1427.009	2053.230	233.592	1509.97	1343.708	2015.711
	Best	1179.696	448.369	654.059	1368.392	675.014	1455.701	1402.901	1994.089	224.106	1180.450	904.327	758.515
	Avg iter	74	97	35	3	99.5	98	71.5	1	175.1	373.15	154	1.15
	Avg time	7.452	1.3944	1.420	1.455	8.182	1.564	1.591	1.574	46.615	8.958	9.265	9.132
	std	83.229	449.204	484.851	381. 515	437.987	162.516	34.094	83.638	581.2636	133.8174	112.9868	302.476
	Violation	1037.40	823.640	1079.742	1647.936	1034.421	1425.534	1191.118	1852.371	12.596	1202.169	1124.437	1597.734

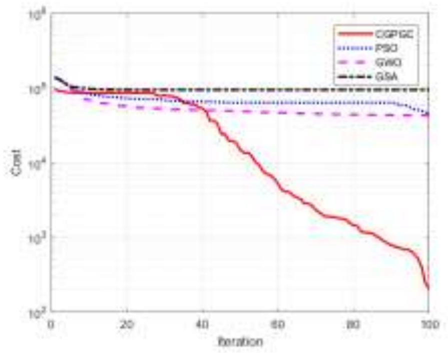
Avg iter =Average iteration Avg time= Average time(s) std= Standard deviation



a) D = 10 & T = 100

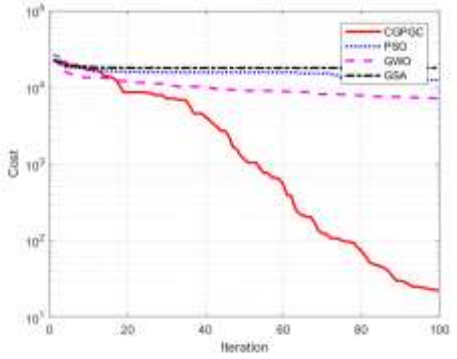


b) D = 30 & T = 100

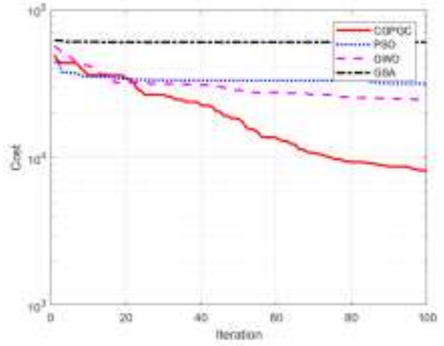


c) D = 30 & T = 500

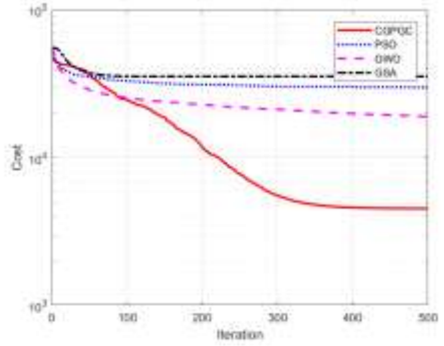
Figure 7. Iteration histories for F1



a) D = 10 & T = 100

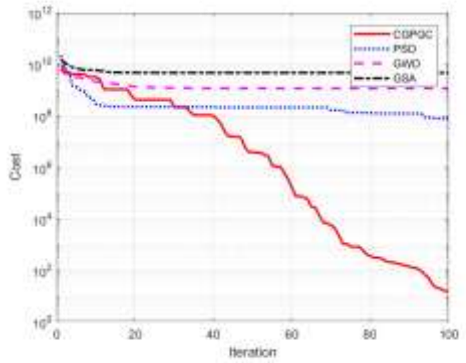


b) D = 30 & T = 100

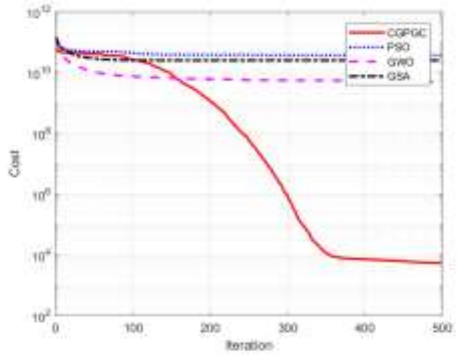


c) D = 30 & T = 500

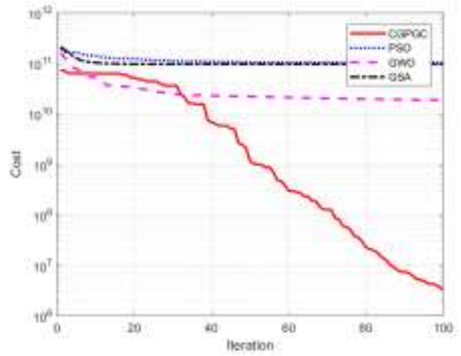
Figure 8. Iteration histories for F5



a) D = 10 & T = 100

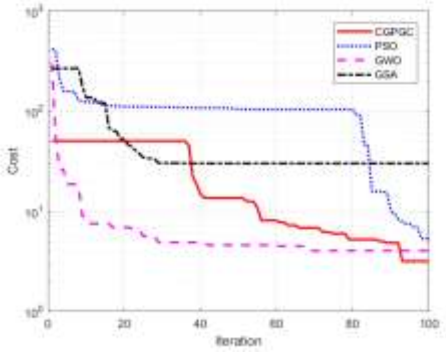


b) D = 30 & T = 100

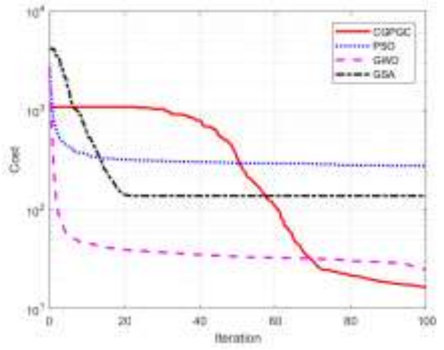


c) D = 30 & T = 500

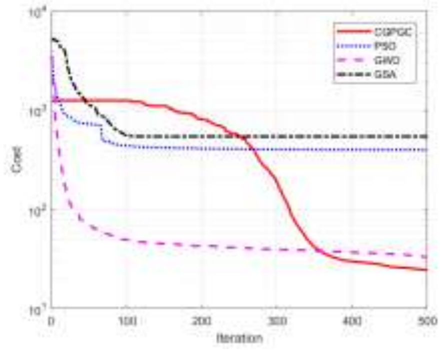
Figure 9. Iteration histories for F6



a) D = 10 & T = 100

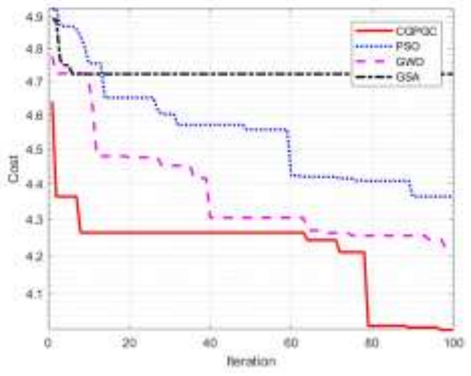


b) D = 30 & T = 100

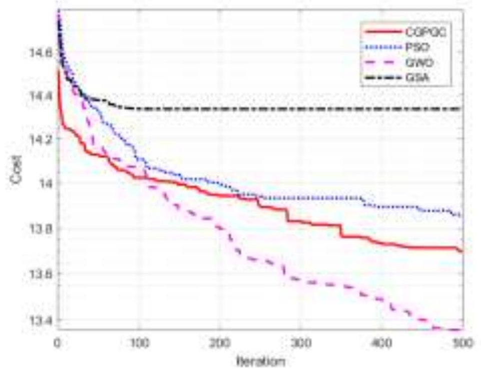


c) D = 30 & T = 500

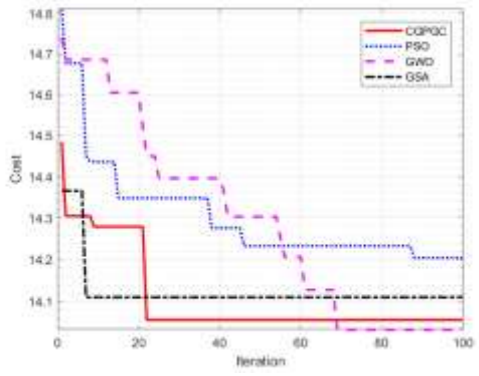
Figure 10. Iteration histories for F13



a) D = 10 & T = 100

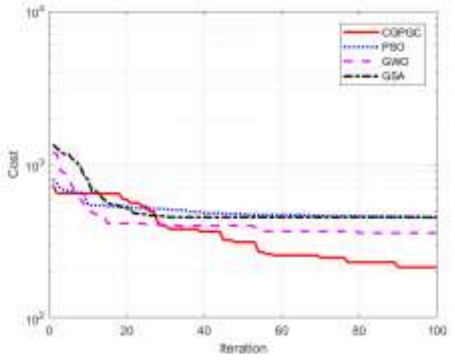


b) D = 30 & T = 100

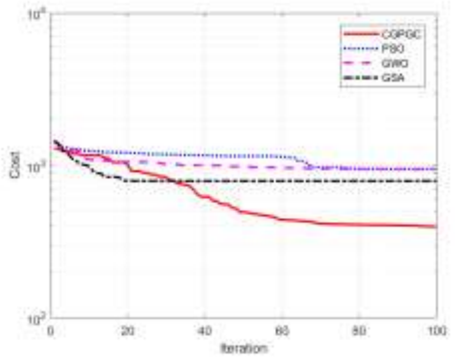


c) D = 30 & T = 500

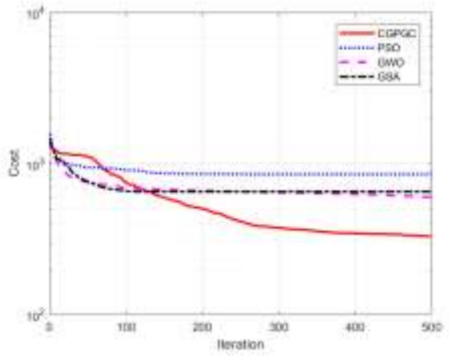
Figure 11. Iteration histories for F14



a) D = 10 & T = 100

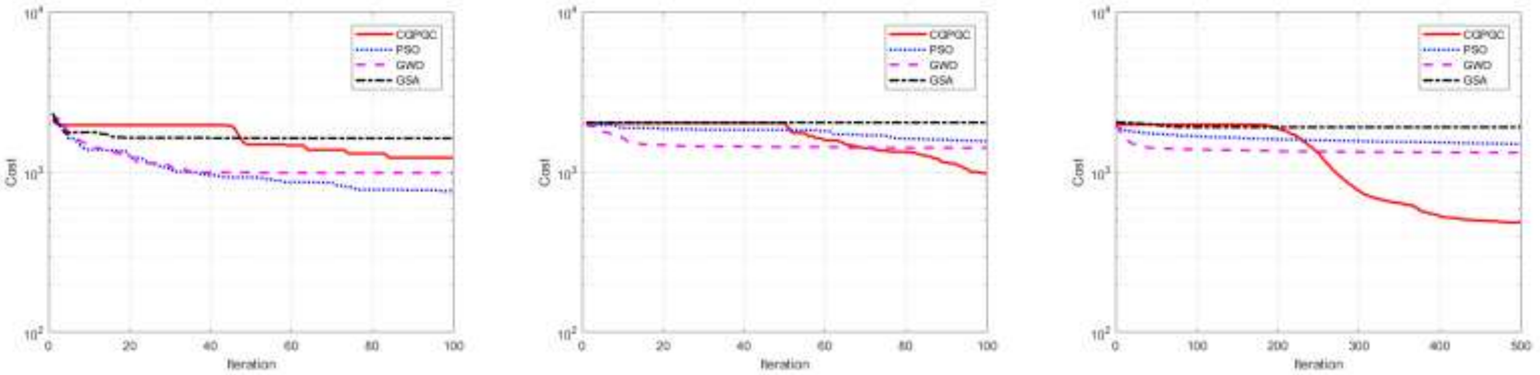


b) D = 30 & T = 100



c) D = 30 & T = 500

Figure 12. Iteration histories for F16



a) D = 10 & T = 100 b) D = 30 & T = 100 c) D = 30 & T = 500

Figure 13. Iteration histories for F25

4.2 Optimization of truss structures

In this section, a number of trusses are examined and analyzed to evaluate the performance of the CGPGC algorithm in structural problems. These examples include a 72-member truss with two types of loading and a 942-member space truss. Then, the results of trusses' analysis using this algorithm are compared with a number of algorithms, which are given below.

4.2.1 72-bar space truss

The 72-member space truss structure, shown in Fig. (14), is analyzed in this example. The material density equals 0.1 $\mathrm{lb/in^3}$ (2767.990 $\mathrm{kg/m^3}$) and the modulus of elasticity is 10000 ksi (68950.)Mpa This space truss is subjected to the following two loading conditions: In the first case, P_x= 5 kips , P_y= 5 kips , and P_z= - 5 kips are applied to the node 17, where the minimum cross-sectional area of each member is 0.1 in^2 (0.6452 cm^2), and in the second case, P_x= 0 , P_y= 0, and P_z= - 5 kips are applied to the nodes 17, 18, 19 and 20, where the minimum cross-sectional area of each member is 0.01 in^2 (0.06452 cm^2). In this case, the structure is symmetric about the *x* and *y* axes [23].

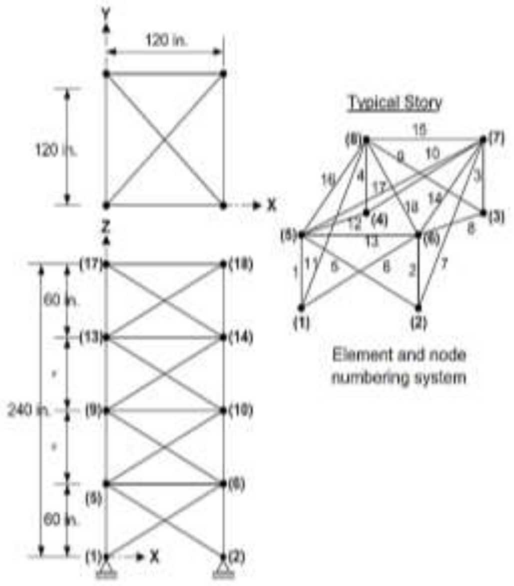


Figure 14. A 72-bar space truss [19]

Taking these conditions into account, the truss members are divided into 16 groups, which can be seen in Table (4). The members are with the stress limitations of ± 25 ksi(172.375 Mpa) and the maximum allowable displacement of the nodes should not exceed ± 25 in(0.635 cm) in the *x* and *y* directions. The results from the CGPGC algorithm and other algorithms can be seen in Tables (5) and (6). The convergence history diagram is also shown in Figs. (15) and (16), respectively.

Table 4: Elements' groups of the 72-bar space truss [21]

Group Number	Members	Group Number	Members
1	A1~A4	9	A37~A40
2	A5~A12	10	A11~A48
3	A13~A16	11	A49~A52
4	A17~A18	12	A53~A54
5	A19~A22	13	A55~A58
6	A23~A30	14	A59~A66
7	A31~A34	15	A67~A70
8	A35~A36	16	A71~A72

Table 5: Optimum values of the cross-sectional areas of the 72-bar truss, case 1

Area (in2)	DHPSACO [22]	IMBA [25]	EFA [26]	MCOA [28]	CGPGC
A1	1.800	1.990	1.990	1.990	1.8689
A2	0.442	0.442	0.563	0.563	0.50901
A3	0.141	0.111	0.111	0.111	0.10021
A4	0.111	0.111	0.111	0.111	0.10012
A5	1.228	1.266	1.228	1.228	1.2672
A6	0.563	0.563	0.442	0.442	0.50658
A7	0.111	0.111	0.111	0.111	0.10004
A8	0.111	0.111	0.111	0.111	0.10031
A9	0.563	0.422	0.563	0.563	0.53123
A10	0.563	0.422	0.563	0.563	0.52042
A11	0.111	0.111	0.111	0.111	0.10003
A12	0.250	0.111	0.111	0.111	0.10024
A13	0.196	0.196	0.196	0.196	0.15624
A14	0.563	0.563	0.563	0.563	0.54552
A15	0.442	0.442	0.391	0.391	0.44161
A16	0.563	0.602	0.563	0.563	0.55252
Weight (lb)	390.380	389.334	389.334	389.334	379.7492
Average weight(lb)	-	389.457	390.913	389.823	380.2289
Std	-	0.84	1.161	0.840	0.45733
Number of Analysis	-	6250	3123	6250	7488

Table 6: Optimum values of the cross-sectional areas of the 72-bar truss, case 2

Area (in2)	Fuzzy GA [27]	CGPGC
A1	1.732	1.867
A2	0.522	0.532
A3	0.01	0.010
A4	0.013	0.010
A5	1.345	1.272
A6	0.551	0.509
A7	0.01	0.010
A8	0.013	0.010
A9	0.492	0.519
A10	0.545	0.512
A11	0.066	0.010
A12	0.013	0.081
A13	0.178	0.169
A14	0.524	0.564
A15	0.396	0.435
A16	0.595	0.562
Weight (lb)	364.40	364.051
Avg. W (lb)	-	364.962
Std	-	0.746
Number of Analysis	14669	7488

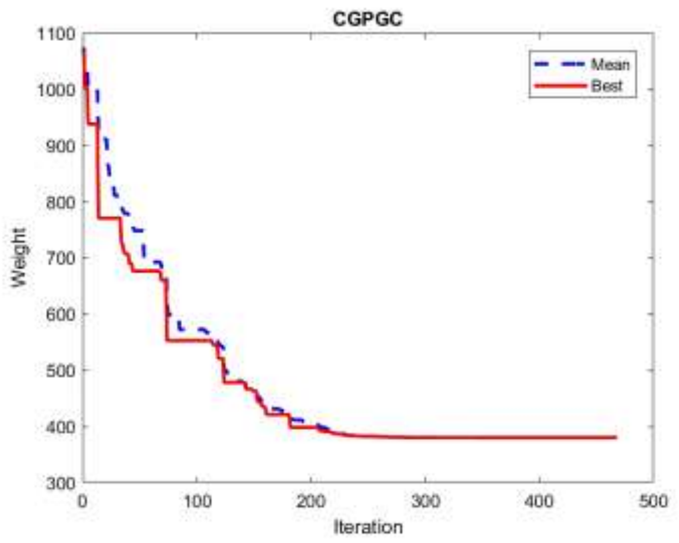


Figure 15. The convergence history of the 72-bar space truss, case 1

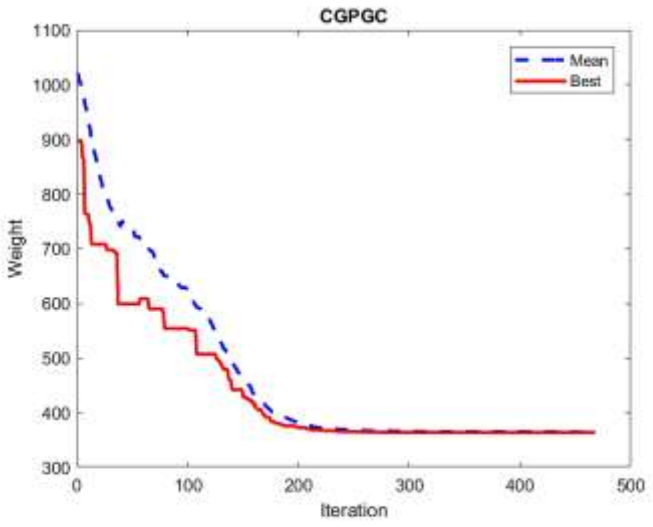


Figure 16. The convergence history of the 72-bar space truss, case 2

4.2.2 942-bar space truss

The 26-story space truss examined in this example contains 942 members and 244 nodes, as shown in Fig. (17). Due to symmetry, the space truss is divided into 59 groups. The material density is 0.1 $^{\text{lb}}/_{\text{in}^3}$ and the modulus of elasticity is 1e7 psi. The area of the permissible sections in this example has been selected in the range of 1 to 200 in^2 [28]. The liading of this structure includes vertical downward loads and horizontal loads, which are given in Table (7). The tensile and compressive stresses of this structure is 170 Mpa (25.0 ksi) [28] . The results of the CGPGC and other algorithms can be seen in Table (8) and also, the convergence history diagram is shown in Fig. (18).

Table 7: Loading details of the 942-bar space truss [28]

	Direction	Load Value	Displacement Value
First vertical	z	13.344 KN (3 kips)	-
Second vertical	z	26.688 KN (6 kips)	-
Third vertical	z	40.032 KN (9 kips)	-
Horizontal at each node on the left side	x	6.672 KN (1.5 kips)	-
Horizontal at each node on the right side	x	4.448 KN (1 kips)	-

Horizontal At each node on the back side	y	4.448 KN (1 kips)	-
Horizontal At each node on the front side	y	4.448 KN (1 kips)	-
First vertical	x, y, z	-	0.381 m (15.0 in)

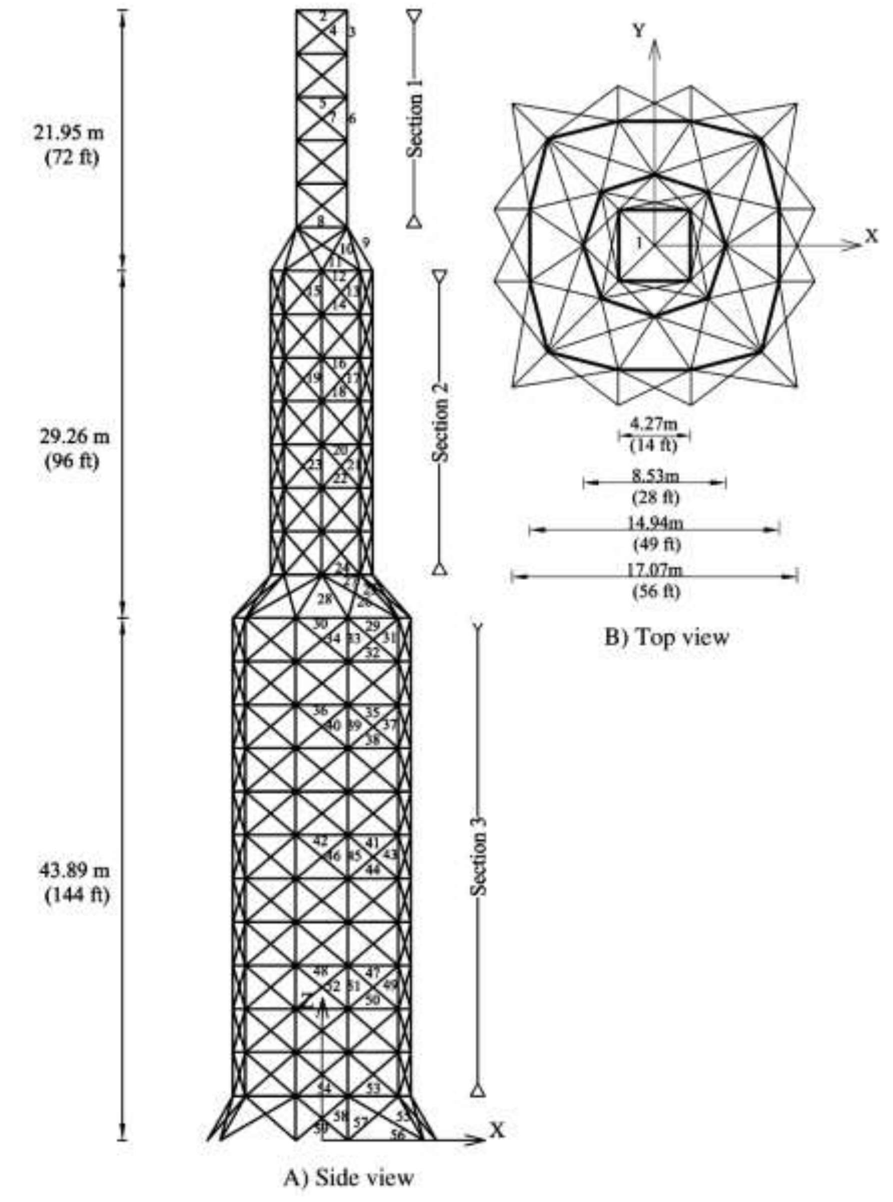


Figure 17. The 942-bar space truss [28]

Table 8: Optimum values of the cross-sectional areas of the 942-bar space truss

Member group	Optimal cross-sectional areas		
	ES [30]	GNMS [31]	CGPGC
1	1.02	2.786	1.2636
2	1.037	1.357	1.06019
3	2.943	5.036	3.53655
4	1.92	2.24	1.84504
5	1.025	1.223	1.02016
6	14.961	14.958	15.4152
7	3.074	2.957	3.10758
8	6.78	10.904	6.86835
9	18.58	14.418	18.0247
10	2.415	3.709	3.45962
11	6.584	5.708	5.47809
12	6.291	4.926	6.09398
13	15.383	14.175	15.781
14	2.1	1.904	2.20256
15	6.021	2.81	4.23461
16	1.022	1	1.01026
17	23.099	18.807	23.0325
18	2.889	2.615	2.67212
19	7.96	12.533	9.06512
20	1.008	1.131	1.08406
21	28.548	30.512	29.1676
22	3.349	3.346	3.42252
23	16.144	17.045	17.0619
24	24.822	18.079	26.4077
25	38.401	39.272	37.0048
26	3.787	2.606	1.36634
27	12.32	9.83	12.8834
28	17.036	13.113	17.059
29	14.733	13.69	14.8719
30	15.031	16.978	16.9973
31	38.597	37.601	38.5
32	3.511	3.06	3.55134
33	2.997	5.511	3.02627
34	3.06	1.801	2.61534
35	1.086	1.157	1.00589
36	1.462	1.242	1.34471
37	59.433	62.774	58.0722
38	3.632	3.328	3.45736

39	1.887	4.237	1.93195
40	4.072	1.72	3.39306
41	1.595	1.015	1.37232
42	3.671	5.643	3.33084
43	79.511	78.009	79.9039
44	3.394	3.221	3.54972
45	1.581	3.593	1.62271
46	4.204	4.767	3.86949
47	1.329	1.153	1.44632
48	2.242	2.17	2.49582
49	96.886	99.641	95.8963
50	3.71	4.147	3.60003
51	1.055	2.16	1.01013
52	4.566	4.15	4.59175
53	9.606	11.207	8.01014
54	2.984	11.09	4.07714
55	45.917	35.92	44.324
56	1	2.194	1.0452
57	62.246	66.171	66.5871
58	2.977	3.34	2.78116
59	1	4.053	1.12176
Weight	141241	142296	141090.751
Avg. W	N/A	N/A	141454.1967
Std	N/A	N/A	297.0353
Number of Analysis	150000	N/A	60000

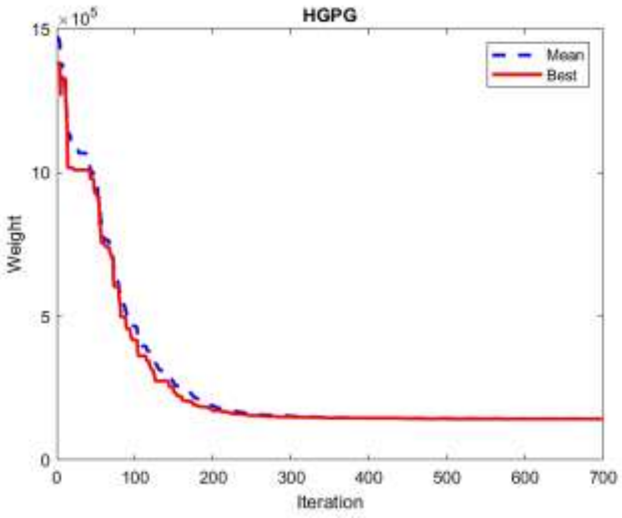


Figure 18. The convergence history of the 942-bar space truss

5. CONCLUSION

Nowadays, proposing practical and efficient AI methods and applying them to various optimization problems for a wide variety of subjects has become an undeniable necessity. In this regard, different types of evolutionary-based algorithms have been proposed for optimization, not only in the realm of structures, but also in many other research domains. Therefore, in this study, we proposed a new and effective hybrid method by combining PSO, GWO, and GSA algorithms, where it is tried to benefit from the strengths of these algorithms and at the same time, improve the weaknesses of each. This optimization makes our proposed model (called CGPGC) an effective approach to be used in optimizing a variety of real problems. The cellular automation method was used along with the three aforementioned evolutionary algorithms so as to increase the rate of convergence of the proposed CGPGC method into the general optimal point, and so as not to get stuck in the trap of local optimality. The obtained results of the proposed method were tested on the CEC2005 standard functions and large-scale space truss structures with high number of variables (a 72-member space truss structure with 16 variables and two different loading conditions, as well as a 942-member space truss structure with 59 variables). The final outcomes were compared with those of other evolutionary algorithms, which indicted the remarkable performance of the proposed CGPGC method.

REFERENCES

1. Darvishi P, Shojaee S. Size and geometry optimization of truss structures using the combination of DNA computing algorithm and generalized convex approximation method, *Int J Optim Civil Eng* 2018; **8**(4): 625-56.
2. Kaveh A, Khayatazad M. A new meta-heuristic method: Ray optimization, *Comput Struct* 2012; **112–113**: 283–94.
3. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T. An improved ray optimization algorithm for design of truss structures, *Period Polytech Civil Eng* 2013; **57**(2): 97–112.
4. Kaveh A, Talatahari S. Hybrid charged system search and particle swarm optimization for engineering design problems, *Eng Computat* 2011.
5. Kaveh A, Malakoutirad S. Hybrid genetic algorithm and particle swarm optimization for the force method-based simultaneous analysis and design, *Iranian J Sci Technol, Tran B-Eng* 2010; **34**(1).
6. 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.
7. Kaveh A, Dadras Eslamlou A, Montazeran AH. Chaotic enhanced colliding bodies algorithms for size optimization of truss structures, *Acta Mech* 2018; **229**(7): 2883–907.
8. Salajegheh F, Salajegheh E, Shojaee S. Optimum design of truss structures with frequency constraints by an enhanced PSOG method based on emigration philosophy, *Eng Optim*, Published online: 19 Dec 2021.
9. Jafari M, Salajegheh E, Salajegheh J. Optimal design of truss structures using a hybrid

method based on particle swarm optimizer and cultural algorithm, *Struct* 2021; **32**: 391–405.

10. Jawad FKJ, Ozturk C, Dansheng W, Mahmood M, Al-azzawi O, Al-jemely A. Sizing and layout optimization of truss structures with artificial bee colony algorithm, *Struct* 2021; **30**: 546-59.
11. Pizarro PN, Massone LM, Rojas FR, Ruiz RO. Use of convolutional networks in the conceptual structural design of shear wall buildings layout, *Eng Struct* 2021; **239**: 112311.
12. Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables, *Comput Struct* 2009; **87**(7–8): 435-43.
13. Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA : A gravitational search algorithm, *Inf Sci (Ny)* 2009; **179**(13): 2232-48.
14. Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer, *Adv. Eng. Softw* 2014; **69**: 46–61.
15. Von Neumann J, Burks AW. Theory of self-reproducing automata, *IEEE Trans, Neural Networks* 1966; **5**(1): 3-14.
16. Ulam S. Random processes and transformations, *in Proceedings of the International Congress on Mathematics* 1952; pp. 264-275.
17. Wolfram S. *Cellular Automata and Complexity: Collected Papers*, CRC Press, 2018.
18. Missoum S, Gürdal Z, Setoodeh S. Local update schemes for cellular automata in structural design, *in 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization* 2002; pp. 5519.
19. Evsutin O, Shelupanov A, Meshcheryakov R, Bondarenko D, Rashchupkina A. The algorithm of continuous optimization based on the modified cellular automaton, *Symmet (Basel)* 2016; **9**: 84.
20. Suganthan PN, Hansen N, Liang J, Deb K. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization, *Natural Comput* 2005; 341-357.
21. Lee KS, Geem ZW. A new structural optimization method based on the harmony search algorithm, *Comput Struct* 2004; **82**(9–10): 781-98.
22. Kaveh A, Talatahari S. A particle swarm ant colony optimization for truss structures with discrete variables, *J Constr Steel Res* 2009; **65**: 1558-68.
23. Camp CV,. Bichon BJ. Design of space trusses using ant colony optimization, *Struct Eng* 2004; **130**(5): 741-51.
24. Perez RE, Behdinan K. Particle swarm approach for structural design optimization, *Comput Struct* 2007; **85**(19–20): 1579-88.

a. Sadollah A, Eskandar H, Bahreininejad A, Kim JH. Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures, *Comput Struct* 2015; **149**: 1-16.

25. Le DT, Bui DK, Ngo TD, Nguyen QH, Nguyen-Xuan H. A novel hybrid methods combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures, *Comput Struct* 2019; **212**: 20–42.
26. Sarma KC, Adeli H. Fuzzy genetic algorithm for optimization of steel structures, *J Struct Eng* 2000; **126**(5): 596-604.

27. Pierezan J, Coelho LDS, Mariani VC, Segundo EHV, Prayogo D. Chaotic coyote algorithm applied to truss optimization problems, *Comput Struct* 2021; **242**: 1-10.
28. Ronagh M. Plastic hinge length of RC columns subjected to both far-fault and near-fault ground motions having forward directivity, Struct Des Tall Spec Build 2011; **24**(2014): 421–39.
29. Hasançebi O, Erbatur F. On efficient use of simulated annealing in complex structural optimization problems, *Acta Mech* 2002; **157**(1–4): 27–50.
30. Hasançebi O. Adaptive evolution strategies in structural optimization: Enhancing their computational performance with applications to large-scale structures, Comput. Struct 2008; **86**(1–2): 119-132.
31. Rahami H, Kaveh A, Aslani M, Najian Asl R. A hybrid modified genetic-nelder mead simplex algorithm for large-scale truss optimization, Int J Optim Civil Eng 2011; **1**(1): 29-46.