Chapter 3
Charged System Search Algorithm

3.1 Introduction

This chapter consists of two parts. In the first part, an optimization algorithm based on some principles from physics and mechanics is presented, which is known as the charged system search (CSS) [1]. In this algorithm the governing Coulomb law from electrostatics and the governing laws of motion from the Newtonian mechanics are utilized. CSS is a multi-agent approach in which each agent is a charged particle (CP). CPs can affect each other based on their fitness values and their separation distances. The magnitude of the resultant force is determined by using the electrostatics laws, and the quality of the movement is determined using the governing laws of motion from the Newtonian mechanics. CSS can be utilized in all optimization fields; especially it is suitable for non-smooth or non-convex domains. CSS needs neither the gradient information nor the continuity of the search space.

In the second part, CSS is applied to optimal design of skeletal structures, and high performance of CSS is illustrated [2].

3.2 Charged System Search

3.2.1 Background

3.2.1.1 Electrical Laws

In physics, an electric charge creates an electric field in its surrounding space, which exerts a force on other electrically charged objects. The electric field surrounding a point charge is given by Coulomb's law. Coulomb confirmed that the electric force between two small charged spheres is proportional to the inverse square of their separation distance. The electric force between charged spheres

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-46173-1_3

Fig. 3.1 Coulomb's torsion balance, used to establish the inverse-square law for the electric force between two charges [1]

A and B in Fig. 3.1 causes the spheres to either attract or repel each other, and the resulting motion causes the suspended fiber to twist. Since the restoring torque of the twisted fiber is proportional to the angle through which the fiber rotates, a measurement of this angle provides a quantitative measure of the electric force of attraction or repulsion [3]. Coulomb's experiments showed that the electric force between two stationary charged particles:

- Is inversely proportional to the square of the separation distance between the particles and directed along the line joining them
- Is proportional to the product of the charges q_i and q_j on the two particles
- Is attractive if the charges are of opposite sign and repulsive if the charges have the same sign

From the above observations, Coulomb's law provides the magnitude of the electric force (Coulomb force) between the two point charges [3] as:

$$F_{ij} = k_e \frac{q_i q_j}{r_{ij}^2} \tag{3.1}$$

where k_e is a constant called the Coulomb constant and r_{ij} is the distance between the two charges.

Consider an insulating solid sphere of radius a, which has a uniform volume charge density and carries a total positive charge q_i. The electric field E_{ij} at a point outside the sphere is defined as:

$$E_{ij} = k_e \frac{q_i}{r_{ij}^2} \tag{3.2}$$

The magnitude of the electric field at a point inside the sphere can be obtained using Gauss's law. This is expressed as:

$$E_{ij} = k_e \frac{q_i}{a^3} r_{ij} \tag{3.3}$$

Note that this result shows that $E_{ij} \to 0$ as $r_{ij} \to 0$. Therefore, the result eliminates the problem that would exist at $r_{ij} = 0$ if E_{ij} is varied as $1/r_{ij}^2$ inside the sphere as it does outside the sphere. That is, if $E_{ij} \propto 1/r_{ij}^2$, the field will be infinite at $r_{ij} = 0$, which is physically impossible. Hence, the electric field inside the sphere varies linearly with r_{ij}. The field outside the sphere is the same as that of a point charge q_i located at $r_{ij} = 0$. Also the magnitudes of the electric fields for points inside and outside the sphere coincide when $r_{ij} = a$. A plot of E_{ij} versus r_{ij} is shown in Fig. 3.2, Ref. [3].

In order to calculate the equivalent electric field at a point ($\mathbf{r}_j$) due to a group of point charges, the superposition principle is applied to fields which follows directly from the superposition of the electric forces. Thus, the electric field of a group of charges can be expressed as:

$$E_j = \sum_{i=1, i \neq j}^{N} E_{ij} \tag{3.4}$$

where N is the total number of charged particles and E_{ij} is equal to:

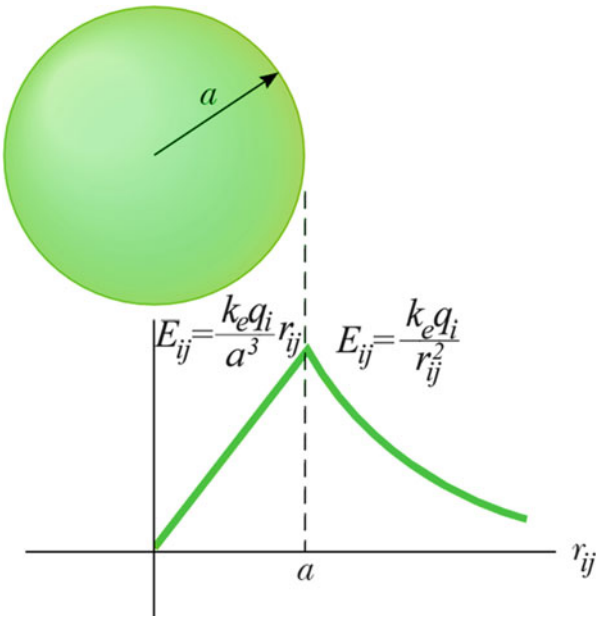


Fig. 3.2 A plot of E_{ij} versus r_{ij} for a uniformly charged insulating sphere [1]

$$E_{ij} = \begin{cases} \dfrac{k_e q_i}{a^3} r_{ij} & \text{if } r_{ij} < a \\ \dfrac{k_e q_i}{r_{ij}^2} & \text{if } r_{ij} \geq a \end{cases} \tag{3.5}$$

In order to obtain both the magnitude and direction of the resultant force on a charge q_j at position $\mathbf{r}_j$ due to the electric field of a charge q_i at position $\mathbf{r}_i$, the full vector form is required which can be expressed as:

$$\mathbf{F}_{ij} = E_{ij} q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{||\mathbf{r}_i - \mathbf{r}_j||} \tag{3.6}$$

For multiple charged particles, this can be summarized as follows:

$$\mathbf{F}_j = k_e q_j \sum_{i, i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) \frac{\mathbf{r}_i - \mathbf{r}_j}{||\mathbf{r}_i - \mathbf{r}_j||} \quad \left\langle \begin{matrix} i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{matrix} \right. \tag{3.7}$$

3.2.1.2 The Governing Laws of Motion from the Newtonian Mechanics

Newtonian mechanics or classical mechanics studies the motion of objects. In the study of motion, the moving object is described as a particle regardless of its size. In

general, a particle is a point-like mass having infinitesimal size. The motion of a particle is completely known if the particle's position in space is known at all times. The displacement of a particle is defined as the change in its position. As a particle moves from an initial position $\mathbf{r}_{old}$ to a final position $\mathbf{r}_{new}$, its displacement is given by:

$$\Delta\mathbf{r} = \mathbf{r}_{new} - \mathbf{r}_{old} \tag{3.8}$$

The slope of tangent line of the particle position represents the velocity of the particle as:

$$\mathbf{v} = \frac{\mathbf{r}_{new} - \mathbf{r}_{old}}{t_{new} - t_{old}} = \frac{\mathbf{r}_{new} - \mathbf{r}_{old}}{\Delta t} \tag{3.9}$$

When the velocity of a particle changes with time, the particle is said to be accelerated. The acceleration of the particle is defined as the change in the velocity divided by the time interval during which that change has occurred:

$$\mathbf{a} = \frac{\mathbf{v}_{new} - \mathbf{v}_{old}}{\Delta t} \tag{3.10}$$

Using Eqs. (3.8), (3.9), and (3.10), the displacement of any object as a function of time is obtained approximately as:

$$\mathbf{r}_{new} = \frac{1}{2}\mathbf{a} \cdot \Delta t^2 + \mathbf{v}_{old} \cdot \Delta t + \mathbf{r}_{old} \tag{3.11}$$

Another law utilized in this article is Newton's second law which explains the question of what happens to an object that has a nonzero resultant force acting on it: the acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass:

$$\mathbf{F} = m \cdot \mathbf{a} \tag{3.12}$$

where m is the mass of the object.

Substituting Eq. (3.12) in Eq. (3.11), we have

$$\mathbf{r}_{new} = \frac{1}{2}\frac{\mathbf{F}}{m} \cdot \Delta t^2 + \mathbf{v}_{old} \cdot \Delta t + \mathbf{r}_{old} \tag{3.13}$$

3.2.2 Presentation of Charged Search System

In this section, a new efficient optimization algorithm is established utilizing the aforementioned physics laws, which is called charged system search (CSS). In the

CSS, each solution candidate $\mathbf{X}_i$ containing a number of decision variables (i.e., $\mathbf{X}_i = \{x_{i,j}\}$) is considered as a charged particle. The charged particle is affected by the electric fields of the other agents. The magnitude of the resultant force is determined by using the electrostatics laws as discussed in Sect. 3.2.1.1, and the quality of the movement is determined using the governing laws of motion from the Newtonian mechanics. It seems that an agent with good results must exert a stronger force than the bad ones, so the amount of the charge will be defined considering the objective function value, *fit*(i). In order to introduce CSS, the following rules are developed:

Rule 1 Many of the natural evolution algorithms maintain a population of solutions which evolve through random alterations and selection [4, 5]. Similarly, CSS considers a number of charged particles (CPs). Each CP has a magnitude of charge (q_i) and as a result creates an electric field in its surrounding space. The magnitude of the charge is defined considering the quality of its solution as follows:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst}, \quad i = 1, 2, \ldots, N \tag{3.14}$$

where *fitbest* and *fitworst* are so far the best and the worst fitness of all particles; *fit* (i) represents the objective function value or the fitness of the agent i; and N is the total number of CPs. The separation distance r_{ij} between two charged particles is defined as follows:

$$r_{ij} = \frac{\left|\left|\mathbf{X}_i - \mathbf{X}_j\right|\right|}{\left|\left|(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best}\right|\right| + \varepsilon} \tag{3.15}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and the jth CPs, $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number to avoid singularities.

Rule 2 The initial positions of CPs are determined randomly in the search space:

$$x_{i,j}^{(o)} = x_{i,\min} + rand \cdot (x_{i,\max} - x_{i,\min}), \quad i = 1, 2, \ldots, n \tag{3.16}$$

where $x_{i,j}^{(o)}$ determines the initial value of the ith variable for the jth CP; $x_{i,\min}$ and $x_{i,\max}$ are the minimum and the maximum allowable values for the ith variable; *rand* is a random number in the interval [0,1]; and n is the number of variables. The initial velocities of charged particles are zero:

$$v_{i,j}^{(o)} = 0, \quad i = 1, 2, \ldots, n \tag{3.17}$$

Rule 3 Three conditions could be considered related to the kind of the attractive forces:

- Any CP can affect another one; i.e., a bad CP can affect a good one and vice versa ($p_{ij}=1$).
- A CP can attract another if its electric charge amount (fitness with revise relation in minimizing problems) is better than the other. In other words, a good CP attracts a bad CP:

$$p_{ij}=\begin{cases}1 & fit(j)>fit(i)\\ 0 & \text{else}\end{cases} \tag{3.18}$$

- All good CPs can attract bad CPs and only some of bad agents attract good agents, considering following probability function:

$$p_{ij}=\begin{cases}1 & \dfrac{fit(i)-fitbest}{fit(j)-fit(i)}>rand \vee fit(j)>fit(i)\\ 0 & \text{else}\end{cases} \tag{3.19}$$

According to the above conditions, when a good agent attracts a bad one, the exploitation ability for the algorithm is provided, and vice versa if a bad CP attracts a good CP, the exploration is provided. When a CP moves toward a good agent, it improves its performance, and so the self-adaptation principle is guaranteed. Moving a good CP toward a bad one may cause losing the previous good solution or at least increasing the computational cost to find a good solution. To resolve this problem, a memory which saves the best so far solutions can be considered. Therefore, it seems that the third of the above conditions is the best rule because of providing strong exploration ability and an efficient exploitation.

Rule 4 The value of the resultant electrical force acting on a CP is determined using Eq. (3.7) as:

$$\mathbf{F}_j=q_j\sum_{i,i\neq j}\left(\frac{q_i}{a^3}r_{ij}\cdot i_1+\frac{q_i}{r_{ij}^2}\cdot i_2\right)p_{ij}(\mathbf{X}_i-\mathbf{X}_j),\quad \begin{cases}j=1,2,\ldots,N\\ i_1=1,i_2=0\Leftrightarrow r_{ij}<a\\ i_1=0,i_2=1\Leftrightarrow r_{ij}\geq a\end{cases} \tag{3.20}$$

where $\mathbf{F}_j$ is the resultant force acting on the jth CP, as illustrated in Fig. 3.3.

In this algorithm, each CP is considered as a charged sphere with radius a, which has a uniform volume charge density. Here, the magnitude of a is set to unity; however, for more complex examples, the appropriate value for a must be defined considering the size of the search space. One can utilize the following equation as a general formula:

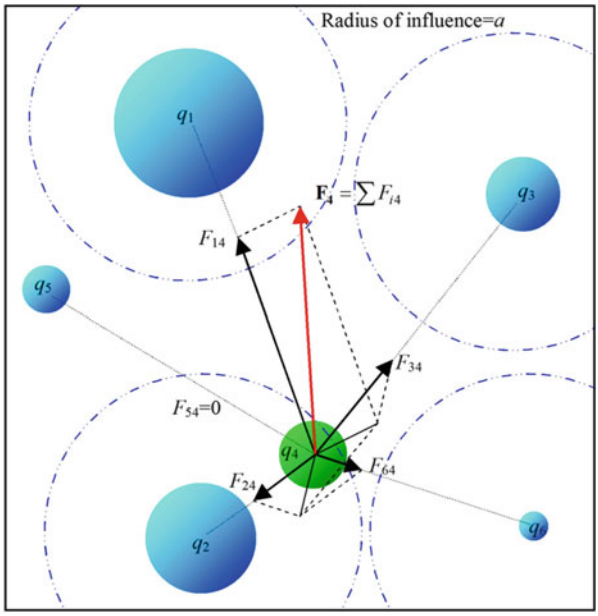


Fig. 3.3 Determining the resultant electrical force acting on a CP [1]

$$a = 0.10 \times \max\left(\left\{x_{i,\max} - x_{i,\min} \middle| i = 1, 2, \ldots, n\right\}\right) \tag{3.21}$$

According to this rule, in the first iterations where the agents are far from each other, the magnitude of the resultant force acting on a CP is inversely proportional to the square of the separation distance between the particles. Thus the exploration power in this condition is high because of performing more searches in the early iterations. It is necessary to increase the exploitation of the algorithm and to decrease the exploration gradually. After a number of searches where CPs are collected in a small space and the separation distance between the CPs becomes small say 0.1, then the resultant force becomes proportional to the separation distance of the particles instead of being inversely proportional to the square of the separation distance. According to Fig. 3.4, if the first equation $\left(F_{ij} \propto 1/r_{ij}^2\right)$ is used for $r_{ij} = 0.1$, we have $F_{ij} = 100 \times k_e q_i q_j$ that is a large value, compared to a force acting on a CP at $r_{ij} = 2$ $\left(F_{ij} = 0.25 \times k_e q_i q_j\right)$, and this great force causes particles to get farther from each other instead of getting nearer, while the second one $\left(F_{ij} \propto r_{ij}\right)$ guaranties that a convergence will happen. Therefore, the parameter a separates the global search phase and the local search phase, i.e., when majority of the agents are collected in a space with radius a, the global search is finished and the optimizing process is continued by improving the previous results, and thus the local search starts. Besides, using these principles controls the balance between the exploration and the exploitation.

It should be noted that this rule considers the competition step of the algorithm. Since the resultant force is proportional to the magnitude of the charge, a better

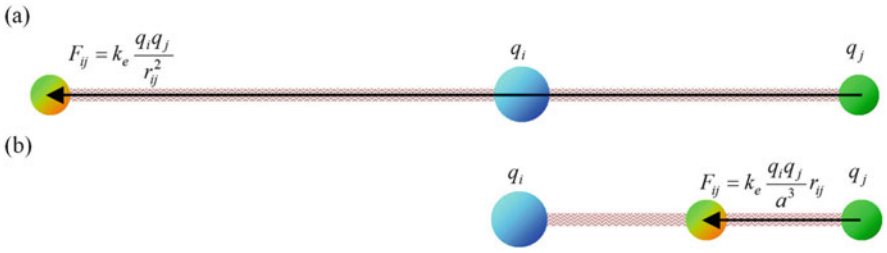


Fig. 3.4 A comparison between the equations [1]. (**a**) $F_{ij} \propto 1/r_{ij}^2$ and (**b**) $F_{ij} \propto r_{ij}$ when $r_{ij} < a$

fitness (great q_i) can create a bigger attracting force, so the tendency to move toward a good CP becomes more than a bad particle.

Rule 5 The new position and velocity of each CP is determined considering Eqs. (3.9) and (3.13) as follows:

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{3.22}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{3.23}$$

where k_a is the acceleration coefficient; k_v is the velocity coefficient to control the influence of the previous velocity; and $rand_{j1}$ and $rand_{j2}$ are two random numbers uniformly distributed in the range of (0,1). Here, m_j is the mass of the CPs which is equal to q_j. Δt is the time step and is set to unity.

The effect of the pervious velocity and the resultant force acting on a CP can be decreased or increased based on the values of the k_v and k_a, respectively. Excessive search in the early iterations may improve the exploration ability; however, it must be deceased gradually, as described before. Since k_a is the parameter related to the attracting forces, selecting a large value for this parameter may cause a fast convergence, and vice versa a small value can increase the computational time. In fact k_a is a control parameter of the exploitation. Therefore, choosing an incremental function can improve the performance of the algorithm. Also, the direction of the pervious velocity of a CP is not necessarily the same as the resultant force. Thus, it can be concluded that the velocity coefficient k_v controls the exploration process, and therefore, a decreasing function can be selected. Thus, k_v and k_a are defined as:

$$k_v = 0.5(1 - iter/iter_{\max}), \quad k_a = 0.5(1 + iter/iter_{\max}) \tag{3.24}$$

where $iter$ is the actual iteration number and $iter_{\max}$ is the maximum number of iterations. With this equation, k_v decreases linearly to zero while k_a increases to one when the number of iterations increases. In this way, the balance between the

exploration and exploitation is saved. Considering the values of these parameters, Eqs. (3.22) and (3.23) can be rewritten as:

$$\mathbf{X}_{j,new} = 0.5rand_{j1} \cdot (1 + iter/iter_{\max}) \cdot \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij} (\mathbf{X}_i - \mathbf{X}_j) + 0.5rand_{j2} \cdot (1 + iter/iter_{\max}) \cdot \mathbf{V}_{j,old} + \mathbf{X}_{j,old} \tag{3.25}$$

$$\mathbf{V}_{j,new} = \mathbf{X}_{j,new} - \mathbf{X}_{j,old} \tag{3.26}$$

Figure 3.5 illustrates the motion of a CP to its new position using this rule. The rules 5 and 6 provide the cooperation step of the CPs, where agents collaborate with each other by information transferring.

Rule 6 Considering a memory which saves the best CP vectors and their related objective function values can improve the algorithm's performance without increasing the computational cost. To fulfill this aim, charged memory (CM) is utilized to save a number of the best so far solutions. In this chapter, the size of the CM (i.e., *CMS*) is taken as *N*/4. Another benefit of the CM consists of utilizing this memory to guide the current CPs. In other words, the vectors stored in the CM can attract current CPs according to Eq. (3.20). Instead, it is assumed that the same number of the current worst particles cannot attract the others.

Rule 7 There are two major problems in relation to many metaheuristic algorithms; the first problem is the balance between exploration and exploitation in the

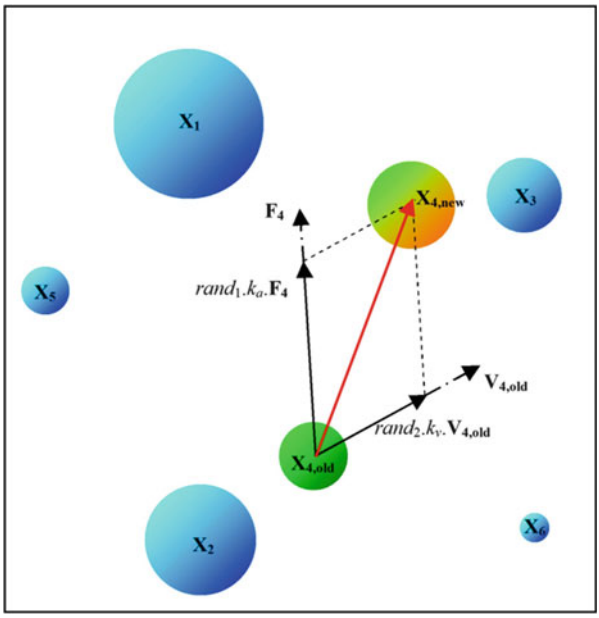


Fig. 3.5 The movement of a CP to the new position [1]

beginning, during, and at the end of the search, and second is how to deal with an agent violating the limits of the variables.

The first problem is solved naturally through the application of above-stated rules; however, in order to solve the second problem, one of the simplest approaches is utilizing the nearest limit values for the violated variable. Alternatively, one can force the violating particle to return to its previous position or one can reduce the maximum value of the velocity to allow fewer particles to violate the variable boundaries. Although these approaches are simple, they are not sufficiently efficient and may lead to reduced exploration of the search space. This problem has previously been addressed and solved using the harmony search-based handling approach [4, 6]. According to this mechanism, any component of the solution vector violating the variable boundaries can be regenerated from the CM as:

$$x_{i,j} = \begin{cases} \text{w.p. CMCR} \implies \text{select a new value for a variable from CM} \\ \qquad\qquad \implies \text{w.p. } (1 - \text{PAR}) \text{ do nothing} \\ \qquad\qquad \implies \text{w.p. PAR choose a neighboring value} \\ \text{w.p. } (1 - \text{CMCR}) \implies \text{select a new value randomly} \end{cases} \tag{3.27}$$

where "w.p." is the abbreviation for "with the probability"; $x_{i,j}$ is the ith component of the CP j; the charged memory considering rate (CMCR) varying between 0 and 1 sets the rate of choosing a value in the new vector from the historic values stored in the CM; and $(1 - \text{CMCR})$ sets the rate of randomly choosing one value from the possible range of values. The pitch adjusting process is performed only after a value is chosen from CM. The value $(1 - \text{PAR})$ sets the rate of doing nothing. For more details, the reader may refer to Refs. [4, 6].

Rule 8 The terminating criterion is one of the following:

- Maximum number of iterations: The optimization process is terminated after a fixed number of iterations, for example, 1000 iterations.
- Number of iterations without improvement: The optimization process is terminated after some fixed number of iterations without any improvement.
- Minimum objective function error: The difference between the values of the best objective function and the global optimum is less than a prefixed anticipated threshold.
- Difference between the best and the worst CPs: The optimization process is stopped if the difference between the objective values of the best and the worst CPs becomes less than a specified accuracy.
- Maximum distance of CPs: The maximum distance between CPs is less than a prefixed value.

Now we can establish a new optimization algorithm utilizing the above rules. The following pseudo code summarizes the CSS algorithm:

Level 1: Initialization

- **Step 1**: *Initialization*. Initialize CSS algorithm parameters; initialize an array of charged particles with random positions and their associated velocities (Rules 1 and 2).
- **Step 2**: *CP ranking*. Evaluate the values of the fitness function for the CPs, compare with each other, and sort increasingly.
- **Step 3**: *CM creation*. Store *CMS* number of the first CPs and their related values of the objective function in the CM.

Level 2: Search

- **Step 1**: *Attracting force determination*. Determine the probability of moving each CP toward others (Rule 3), and calculate the attracting force vector for each CP (Rule 4).
- **Step 2**: *Solution construction*. Move each CP to the new position and find the velocities (Rule 5).
- **Step 3**: *CP position correction*. If each CP exits from the allowable search space, correct its position using Rule 7.
- **Step 4**: *CP ranking*. Evaluate and compare the values of the objective function for the new CPs; and sort them increasingly.
- **Step 5**: *CM updating*. If some new CP vectors are better than the worst ones in the CM, include the better vectors in the CM and exclude the worst ones from the CM (Rule 6).

Level 3: Terminating Criterion Controlling

- Repeat search level steps until a terminating criterion is satisfied (Rule 8).

The flowchart of the CSS algorithm is illustrated in Fig. 3.6.

3.3 Validation of CSS

In order to verify the efficiency of the new algorithm, some numerical examples are considered from literature. The examples contain 18 unimodal and multimodal functions. These numerical examples are presented in Sect. 3.1. The performance of the CSS to optimize these functions is investigated in Sect. 3.2. In Sect. 3.3, some well-studied engineering design problems taken from the optimization literature are used to illustrate the way in which the proposed method works.

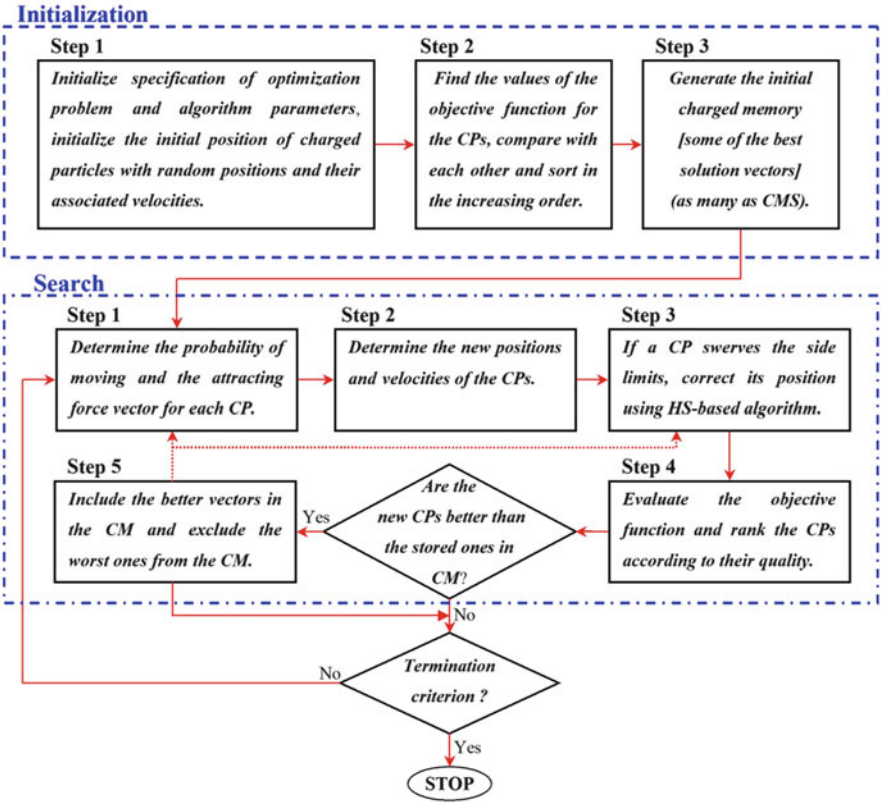


Fig. 3.6 The flowchart of the CSS [1]

3.3.1 Description of the Examples

In this section, a number of benchmark functions chosen from Ref. [7] are optimized using CSS and compared to GA and some of its variations to verify the efficiency of CSS. The description of these test problems is provided in Table 3.1. When the dimension is selected as 2, a perspective view and the related contour lines for some of these functions are illustrated in Fig. 3.7.

3.3.2 Results

Similar to the other metaheuristics, for the CSS a large value for the number of CPs increases the search strength of the algorithm as well as the computational cost, and vice versa a small number causes a quick convergence without performing a

Table 3.1 Specifications of the benchmark problems

Function name	Interval	Function	Global minimum
Aluffi-Pentini	$\mathbf{X} \in [-10,10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in [-100,100]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in [-50,50]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10,10]^2$	$f(\mathbf{X}) = \big(\lvert x_1\rvert - 5\big)^2 + \big(\lvert x_2\rvert - 5\big)^2$	0.0
Branin	$0 \le x_2 \le 15\ -5 \le x_1 \le 10$	$f(\mathbf{X}) = \big(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1\big)^2 + 10\big(1 - \frac{1}{8\pi}\big)\cos(x_1) + 10$	0.397887
Camel	$\mathbf{X} \in [-5,5]^2$	$f(\mathbf{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.0316
Cb3	$\mathbf{X} \in [-5,5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^5 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$n = 4,\ \mathbf{X} \in [-1,1]^n$	$f(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in [-5.12,5.12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$n = 2,4,8,\ \mathbf{X} \in [-1,1]^n$	$f(\mathbf{X}) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	−1
Goldstein and Price	$\mathbf{X} \in [-2,2]^2$	$f(\mathbf{X}) = \big[1 + (x_1 + x_2 + 1)^2\big(19 - 14x_1 + 3x_1^2 - {}^1 14x_2 + 6x_1x_2 + 3x_2^2\big]$ $\times \big[30 + (2x_1 - 3x_2)^2\big(18 - 32x_1 - 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\big]^2$	3.0
Griewank	$\mathbf{X} \in [-100,100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0

Hartman 3	$\mathbf{X} \in [0,1]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$ $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and $p = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$.	-3.862782
Hartman 6	$\mathbf{X} \in [0,1]^6$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	-3.322368
Rastrigin	$\mathbf{X} \in [-1,1]^2$	$f(\mathbf{X}) = \sum_{i=1}^{2} \left(x_i^2 - \cos\left(18x_i\right)\right)$	-2.0
Rosenbrock	$\mathbf{X} \in [-30,30]^n$, $n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2$	0.0

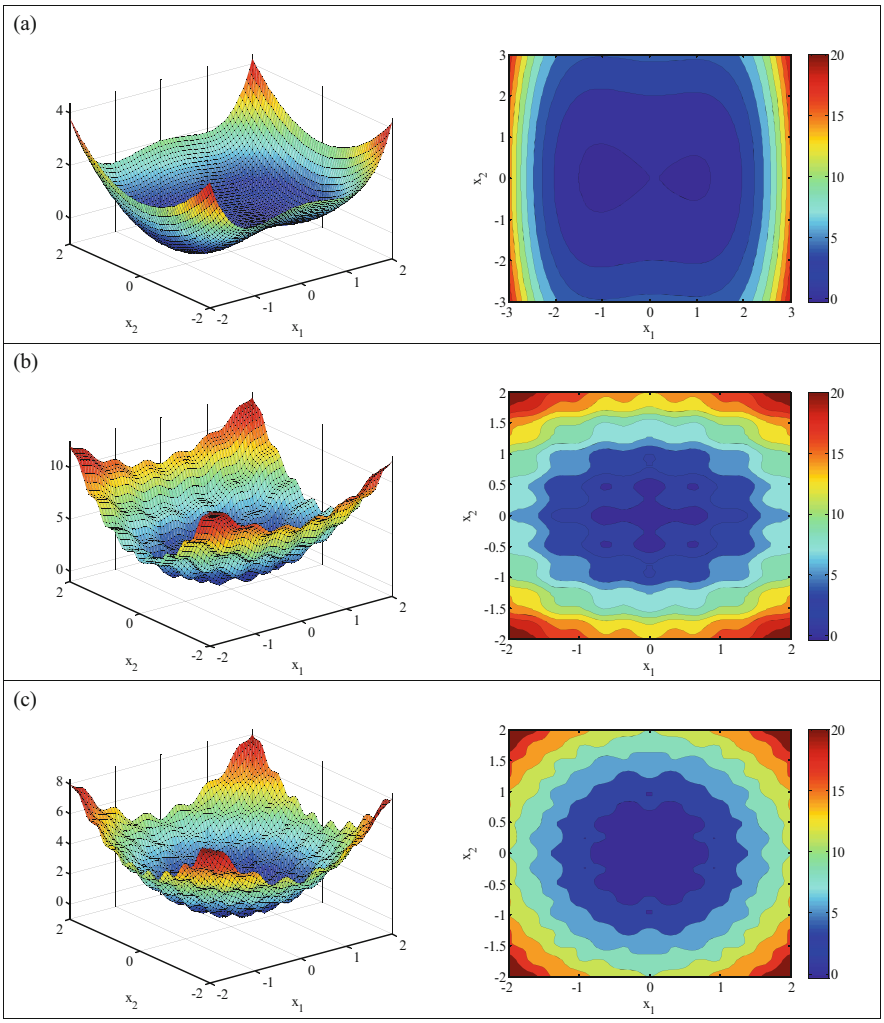


Fig. 3.7 A perspective view and the related contour lines for some of function when $n = 2$, [1]. (**a**) Aluffi-Pentini, (**b**) Bohachevsky 1, (**c**) Bohachevsky 2, (**d**) Becker and Lago, (**e**) Branin, (**f**) Camel, (**g**) Cb3, (**h**) Cosine mixture, (**i**) Exponential, (**j**) Griewank, (**k**) Rastrigin, (**l**) Rosenbrock

complete search. Here, the number of CPs is set to 20, and the maximum number of the permitted iterations is considered as 200. These values seem to be suitable for finding the optimum results. The value of HMCR is set to 0.95 and that of PAR is taken as 0.10 [4]. The results obtained by CSS are listed in Table 3.2 along with those obtained by GA and some of its variations, which are directly derived from [7]. The numbers denote the average number of function evaluations from 50 independent runs for every objective function described in Sect. 3.1. The numbers in

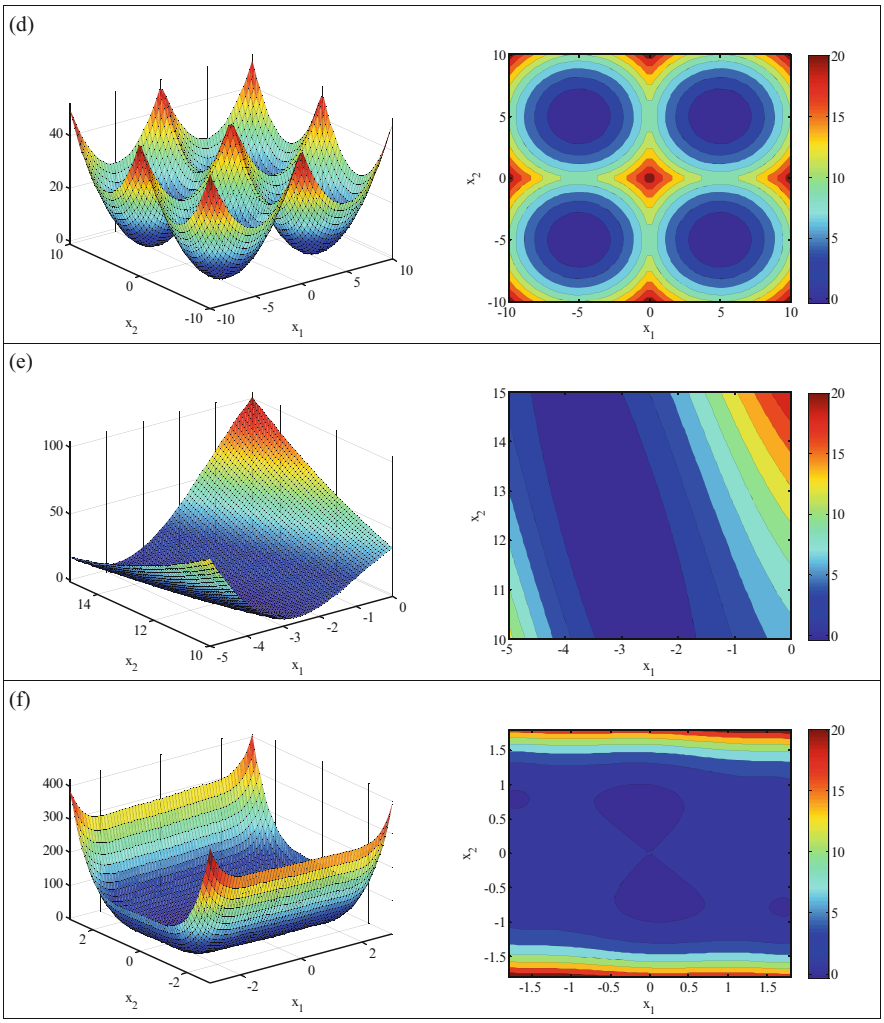


Fig. 3.7 (continued)

parentheses represent the fraction of successful runs in which the algorithm has located the global minimum with predefined accuracy, which is taken as $\varepsilon = f_{min} - f_{final} = 10^{-4}$. The absence of the parentheses denotes that the algorithm has been successful in all independent runs. Although the GEN-S-M-LS finds good results in some cases, it must be noted that GEN-S-M-LS utilizes some auxiliary mechanisms such as an improved stopping rule, a new mutation mechanism, and a repeated application of a local search procedure. To sum up, comparison of the results demonstrates that CSS has a faster convergence than original GA and its variations.

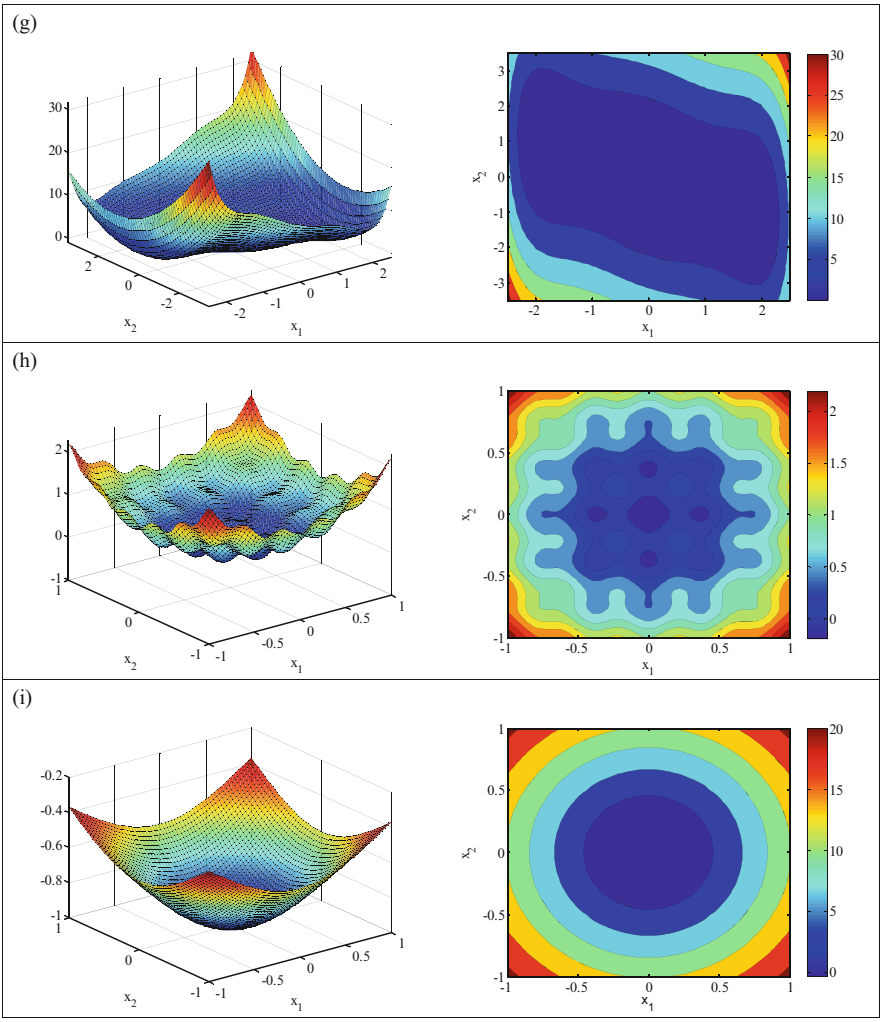


Fig. 3.7 (continued)

In order to have some general idea about the way the CSS works, Fig. 3.8 is prepared to show the positions of the current CPs and the stored CPs in the CM for the first example. It can be seen that in the first iterations, the CPs investigate the entire search space to discover a favorite space (global search). When this favorable region containing a global optimum is discovered, the movements of the CPs are limited to this space in order to provide more exploitation (local search).

For many metaheuristic algorithms, it is common property that if all the agents get gathered in a small space, i.e., if the agents are trapped in part of the search space, escaping from this may be very difficult. Since prevailing forces for the CSS

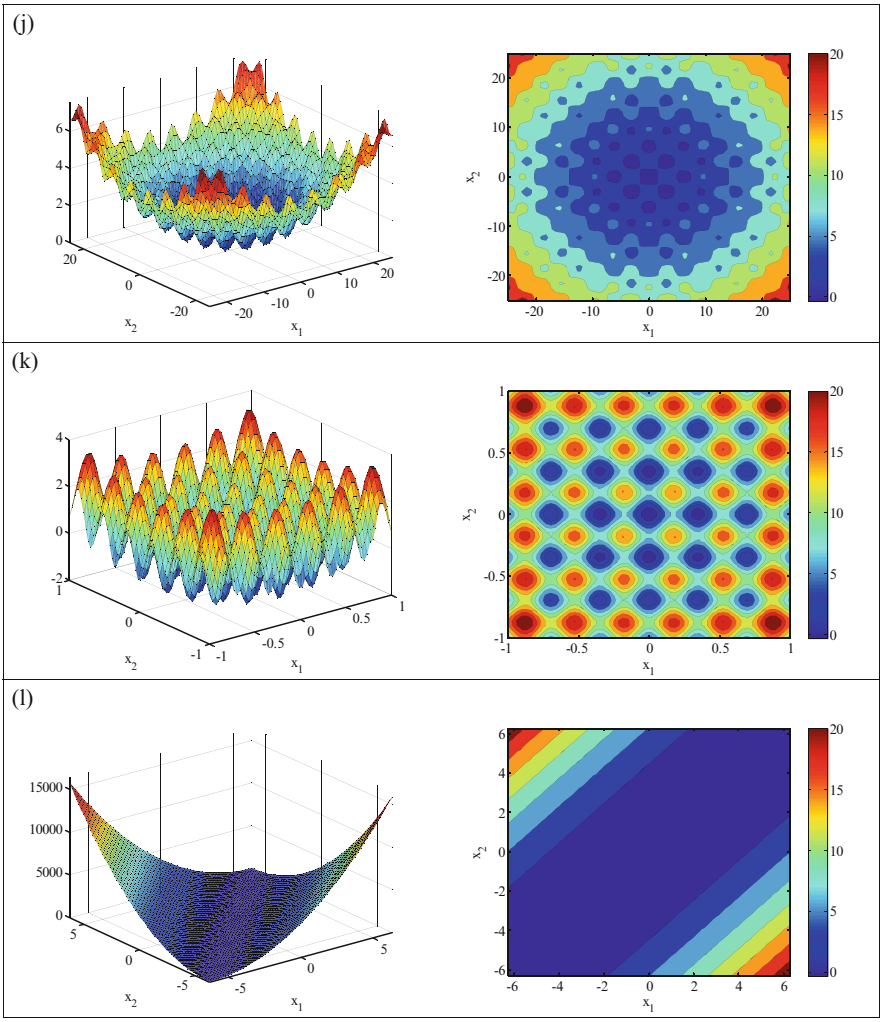


Fig. 3.7 (continued)

algorithm are attracting forces, it looks as if the above problem has remained unsolved for this method. However, having a good balance between the exploration and the exploitation, and considering three steps containing self-adaptation, cooperation, and competition in the CSS, can solve this problem. As illustrated in Fig. 3.9 which shows the positions of the CPs for the first example when all the initial agents are located in a small part of the space, CSS can escape from this space and go toward the favorite space.

Table 3.2 Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS
AP	1360 (0.99)	1360	1277	1253	804
Bf1	3992	3356	1640	1615	1187
Bf2	20,234	3373	1676	1636	742
BL	19,596	2412	2439	1436	423
Branin	1442	1418	1404	1257	852
Camel	1358	1358	1336	1300	575
Cb3	9771	2045	1163	1118	436
CM	2105	2105	1743	1539	1563
Dejoung	9900	3040	1462	1281	630
Exp2	938	936	817	807	132
Exp4	3237	3237	2054	1496	867
Exp8	3237	3237	2054	1496	1426
Goldstein and Price	1478	1478	1408	1325	682
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1551
Hartman3	1350	1350	1332	1274	860
Hartman6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1783
Rastrigin	1533 (0.97)	1523 (0.97)	1392	1381	1402
Rosenbrock	9380	3739	1675	1462	1452
Total	112,311 (96.72)	41,640 (96.77)	29,166 (98.16)	25,193 (98.16)	17,367

3.4 Charged System Search for Structural Optimization

3.4.1 Statement of the Optimization Design Problem

For optimum design of structures, the objective function can be expressed as:

$$\text{minimize} \qquad W(\mathbf{X}) = \sum_{i=1}^{n} \rho_i \cdot x_i \cdot L_i \tag{3.28}$$

where $W(\mathbf{X})$ is the weight of the structure; n is the number of members making up the structure; ρ_i represents the material density of member i; L_i is the length of member i; x_i is the cross-sectional area of member i chosen between $x_{\min}$ and $x_{\max}$; and *min* is the lower bound and *max* is the upper bound. This minimum design also has to satisfy inequality constraints that limit design variable sizes and structural responses (Lee and Geem [8]).

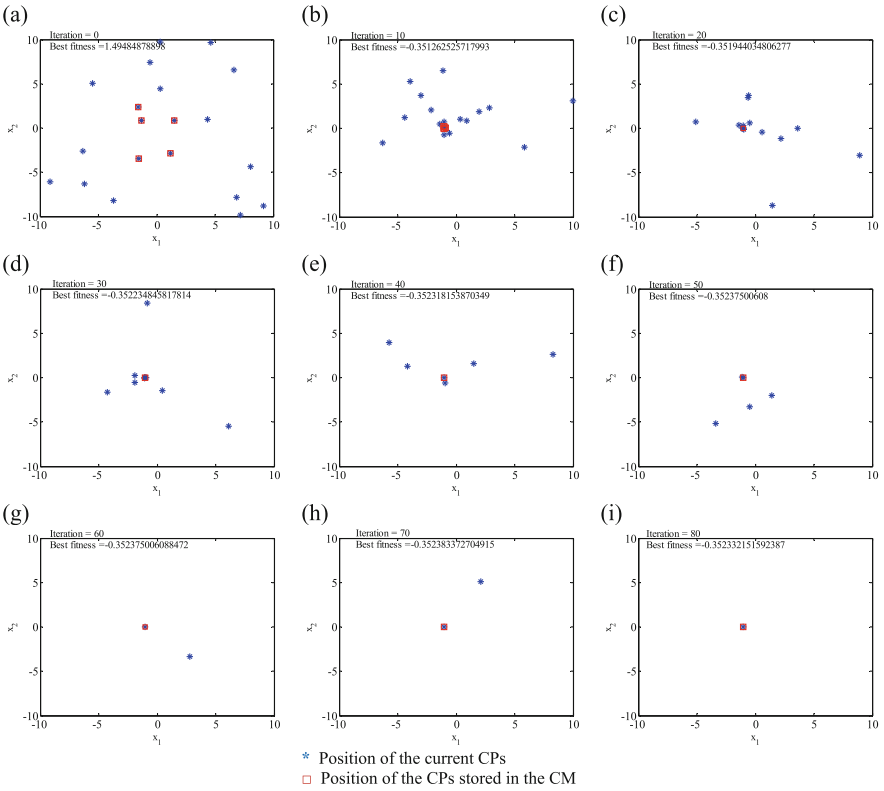


Fig. 3.8 The positions of the current CPs and the stored CPs in the CM for the first example [1]. *Asterisk* Position of the current CPs. *Square* Position of the CPs stored in the CM

3.4.1.1 Constraint Conditions for Truss Structures

For truss structures, the constraints are as follows:

$$\begin{aligned} &\delta_{\min} \le \delta_{\mathrm{i}} \le \delta_{\max} \qquad i = 1, 2, ..., m \\ &\sigma_{\min} \le \sigma_{\mathrm{i}} \le \sigma_{\max} \qquad i = 1, 2, ..., n \\ &\sigma_i^b \le \sigma_{\mathrm{i}} \le 0 \qquad i = 1, 2, ..., nc \end{aligned} \tag{3.29}$$

in which m is the number of nodes; nc denotes the number of compression elements; σ_i and δ_i are the stress and nodal displacement, respectively; and σ_i^b represents allowable buckling stress in member i when it is in compression.

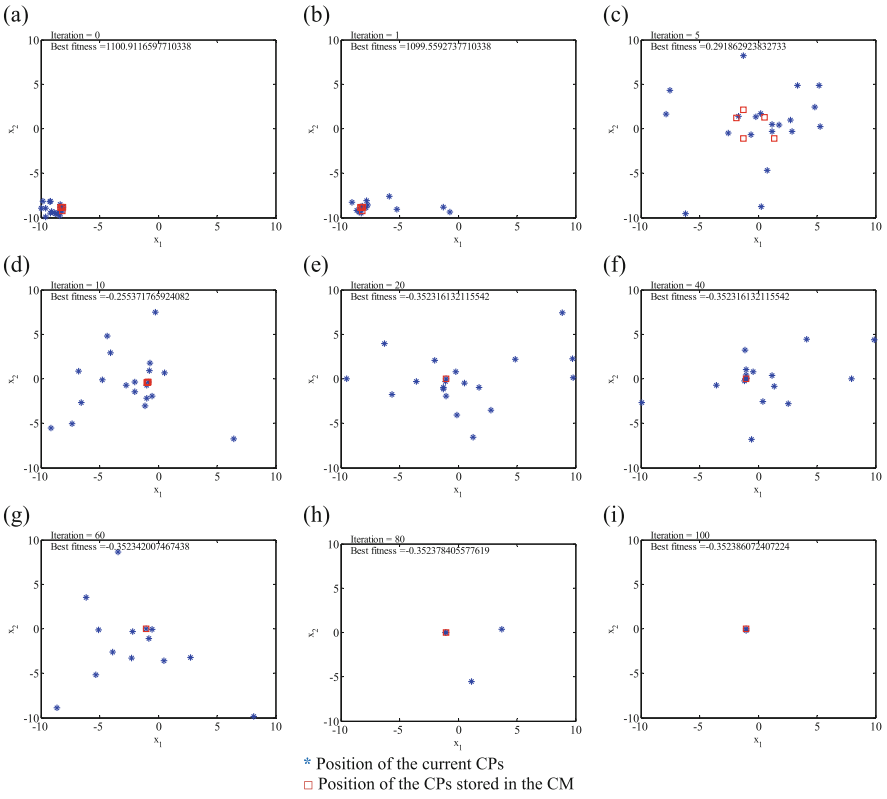


Fig. 3.9 The positions of the CPs for the first example when the all initial agents are introduced in a small part of the space [1]. *Asterisk* Position of the current CPs. *Square* Position of the CPs stored in the CM

3.4.1.2 Constraint Conditions for Frame Structures

For the frame structures, according to the ASD-AISC [9] code, the constraints are as follows:

The stress limitations:

$$\frac{f_a}{F_a}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}\le 0.15 \tag{3.30}$$

$$\frac{f_a}{F_a}+\frac{C_{mx}f_{bx}}{\left(1-\frac{f_a}{F'_{ex}}\right)F_{bx}}+\frac{C_{my}f_{by}}{\left(1-\frac{f_a}{F'_{ey}}\right)F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}>0.15 \tag{3.31}$$

$$\frac{f_a}{0.6F_y}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}>0.15 \tag{3.32}$$

The slenderness ratio limitation:

$$\begin{cases} \lambda_{\mathrm{i}} = \dfrac{k_i L_i}{r_i} \leq 300 \ \text{For tension members} \\ \lambda_{\mathrm{i}} = \dfrac{k_i L_i}{r_i} \leq 200 \ \text{For compression members} \end{cases} \tag{3.33}$$

where f_a $(=P/A_i)$ represents the computed axial stress. The computed flexural stresses due to bending of the member about its major (x) and minor (y) principal axes are denoted by f_{bx} and f_{by}, respectively. F'_{ex} and F'_{ey} denote the Euler stresses about principal axes of the member that are divided by a factor of safety of 23/12. The allowable bending compressive stresses about major and minor axes are designated by F_{bx} and F_{by}. C_{mx} and C_{my} are the reduction factors, introduced to counterbalance overestimation of the effect of secondary moments by the amplification factors $\left(1 - \frac{f_a}{F'_{ex}}\right)$. For unbraced frame members, these factors are taken as 0.85. For braced frame members without transverse loading between their ends, these are calculated from $C_m = 0.6 - 0.4M_1/M_2$, where M_1/M_2 is the ratio of smaller end moment to the larger end moment. Finally, for braced frame members having transverse loading between their ends, these factors are determined from the formula $C_m = 1 + \psi\left(f_a/F'_e\right)$ based on a rational approximate analysis outlined in ASD-AISC [9] Commentary-H1, where ψ is a parameter that considers maximum deflection and maximum moment in the member. F_a stands for the allowable axial stress under axial compression force alone and is calculated depending on elastic or inelastic bucking failure mode of the member according to the slenderness ratio:

$$F_a = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_C^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_C} - \dfrac{\lambda_i^3}{8C_C^3}\right) & \text{For } \lambda_i < C_C \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & \text{For } \lambda_i \geq C_C \end{cases} \tag{3.34}$$

where $E =$ the modulus of elasticity; $F_y =$ the yield stress of steel; $C_C =$ the slenderness ratio dividing the elastic and inelastic buckling regions $\left(C_C = \sqrt{2\pi^2 E/F_y}\right)$; $\lambda_i =$ the slenderness ratio $(\lambda_i = kL_i/r_i)$; $k =$ the effective length factor; and $r_i =$ the governing radius of gyration. For an axially loaded bracing member whose slenderness ratio exceeds 120, F_a is increased by a factor of $(1.6 - L_i/200r_i)$ considering relative unimportance of the member. Equation (3.23) represents the slenderness limitations imposed on all members such that maximum slenderness ratio is limited to 300 for members under tension and to 200 for members under compression loads.

Geometric constraints:

Geometric constraints are considered between beams and columns framing into each other at a common joint for practicality of an optimum solution generated. For the two beams B1 and B2 and the column shown in Fig. 3.10, the following geometric constraints are written (Saka and Hasançebi [10]):

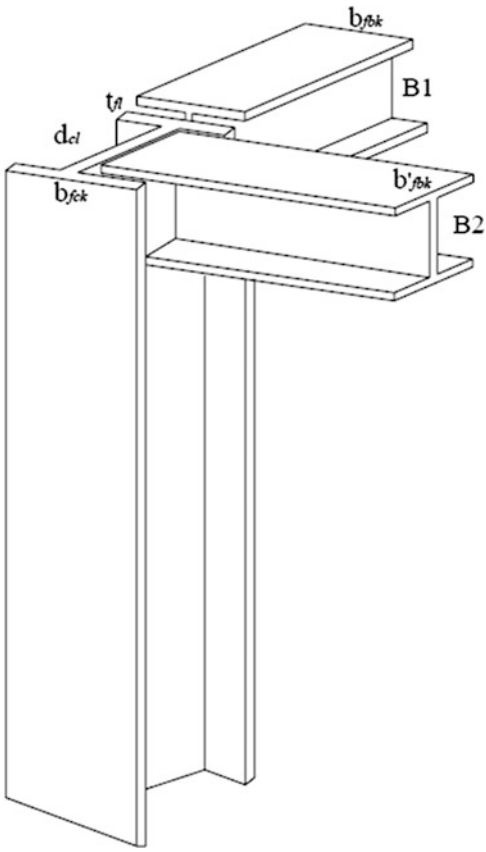


Fig 3.10 Beam-column geometric constraints [2]

$$b_{fb} \leq b_{fc} \tag{3.35}$$

$$b_{fb}^{'} \leq \left(d_c - 2t_f\right) \tag{3.36}$$

where b_{fb}, $b_{fb}^{'}$, and b_{fc} are the flange width of the beam B1, the beam B2, and the column, respectively; d_c is the depth of the column; and t_f is the flange width of the column. Equation (3.35) ensures that the flange width of the beam B1 remains smaller than that of the column. On the other hand, Eq. (3.36) guarantees that flange width of the beam B2 remains smaller than clear distance between the flanges of the column.

Maximum lateral displacement:

$$\frac{\Delta_T}{H} \leq R \tag{3.37}$$

Inter-story displacement constraints:

$$\frac{d_i}{h_i} \leq R_I, \quad i = 1, 2, \ldots, ns \tag{3.38}$$

where Δ_T is the maximum lateral displacement, H is the height of the frame structure, R is the maximum drift index (=1/400), d_i is the inter-story drift, h_i is the story height of the ith floor, ns represents the total number of stories, and R_I is the inter-story drift index permitted by the code of the practice (=1/400).

3.4.1.3 Design Loads for Frame Structures

The frame examples are subjected to various gravity loads in addition to lateral wind forces. The gravity loads acting on floor slabs cover dead (D), live (L), and snow (S) loads. All the floors excluding the roof are subjected to a design dead load of 2.88 kN/m^2 and a design live load of 2.39 kN/m^2. The roof is subjected to a design dead load of 2.88 kN/m^2 plus snow load. The design snow load is computed using Equation (7–1) in ASCE 7–05 [11], resulting in a design snow pressure of 0.75 kN/m^2. The calculated gravity loads are applied as uniformly distributed loads on the beams using distribution formulas developed for slabs. The design wind loads (W) are also computed according to ASCE 7–05 using the following equation:

$$p_w = \left(0.613 K_z K_{zt} K_d V^2 I\right)\left(G C_p\right) \tag{3.39}$$

where p_w is the design wind pressure in kN/m^2; K_z (=1.07) is the velocity exposure coefficient; K_{zt} (=1.0) is the topographic factor; K_d (=0.85) is the wind directionality factor; I (=1.15) is the importance factor; V (=46.94 m/s) is the basic wind; G (=0.85) is the gust factor; and C_p (=0.8 for windward face and −0.5 for leeward face) is the external pressure coefficient. The calculated wind loads are applied as uniformly distributed lateral loads on the external beams of the frames located on windward and leeward facades at every floor level.

The load combination per ASD-AISC specification is considered as:

$$\begin{gathered}(\mathrm{D} + \mathrm{L} + \mathrm{S} + \mathrm{W_x}). \\ \left(\mathrm{D} + \mathrm{L} + \mathrm{S} + \mathrm{W_y}\right).\end{gathered}$$

It should be noted that for wind forces in the above load combinations, two cases are considered. In the first case, the wind loading is acting along x-axis, whereas in the second one it is applied along y-axis.

3.4.2 CSS Algorithm-Based Structural Optimization Procedure

As defined in the previous section, there are some problem-specific constraints in structural optimization problems that must be handled. The penalty function method has been the most popular constraint-handling technique due to its simple principle and ease of implementation. In utilizing the penalty functions, if the constraints are between the allowable limits, the penalty will be zero; otherwise, the amount of penalty is obtained by dividing the violation of allowable limit to the limit itself. Since the CSS is independent of the type of penalty function, one can easily utilize another approach in the application of CSS.

Detailed procedure of the proposed CSS algorithm-based method to determine optimal design of structures is shown in Fig. 3.11. Considering the rules defined for the CSS in Sect. 3.3, and utilizing the penalty functions to handle the problem-specific constraints, the CSS algorithm-based structural optimization procedure can be divided into the following three phases:

Phase 1: Initialization CSS algorithm parameters such as N, CMS, k_v, k_a, and design variable bounds are initialized. An array of N Charged Particles (CPs) with random positions are generated considering the variable bounds together with their associated velocities. The structures associated with the generated CPs are analyzed and the fitness functions values of the CPs are evaluated considering the weight of the structure and the penalty functions. Then, CPs are ranked in the increasing order of their fitness function values. CMS number of the first CPs and their related values of the fitness function are stored in the CM.

Phase 2: Search Each CP moves to the new position considering the probability of motion [Eq. (3.24)], the magnitude of the attracting force vector [Eq. (3.25)], and the motion laws [Eqs. (3.26) and (3.27)]. If each CP exits from the allowable search space, its position is corrected using the harmony-based algorithm. Then, the new CPs are analyzed to evaluate the fitness function values of their corresponding CPs and to sort them increasingly. Then, some of the good new CPs are stored in the CM and the worst ones are excluded from the CM.

Phase 3: Terminating Criterion Controlling Search level is continued until a terminating criterion is satisfied.

3.5 Numerical Examples

In this section, three truss and two frame structures are optimized utilizing the new algorithm. The final results are then compared to the solutions of other advanced metaheuristic methods to demonstrate the efficiency of this work. For the CSS algorithm, a population of 20 CPs is used for the first and the second truss examples, and a population of 50 candidates is selected for the remaining examples. The effect

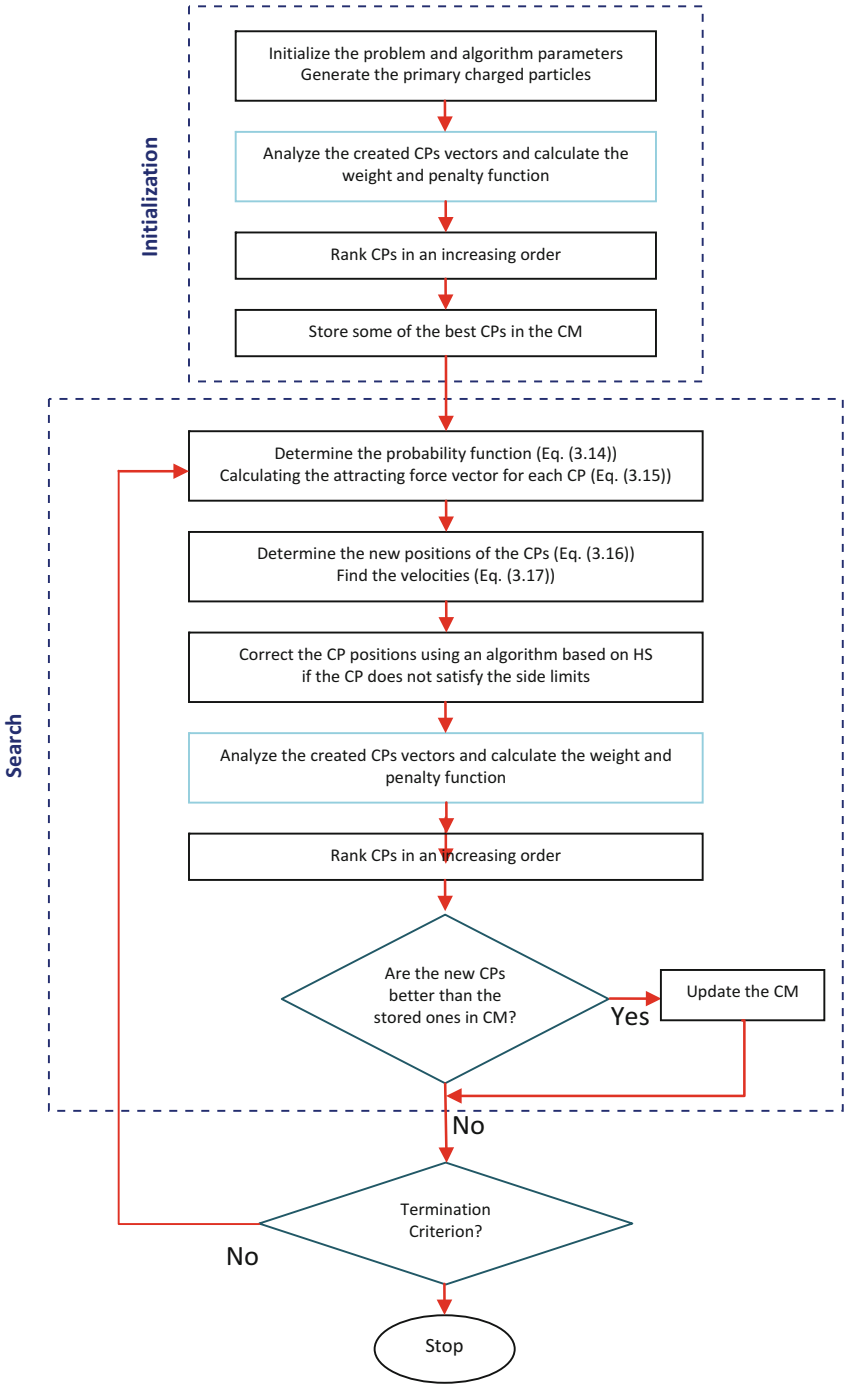


Fig. 3.11 The flowchart of the CSS for the truss structures [2]

of the pervious velocity and the resultant force affecting a CP can be decreased or increased based on the values of the k_v and k_a. Here, k_v and k_a are defined as:

$$\begin{aligned} k_v &= c(1 - iter/iter_{\max}) \\ k_a &= c(1 + iter/iter_{\max}) \end{aligned} \tag{3.40}$$

where *iter* is the iteration number, $iter_{\max}$ is the maximum number of the iterations, and c is set to 0.5 and 0.2 when the population of 20 and 50 CPs are selected, respectively. With this equation, k_v decreases linearly while k_a increases when the number of iterations increases. In this way, the balance between the exploration and the exploitation is saved.

In order to investigate the effect of the initial solution on the final result and because of the stochastic nature of the algorithm, each example is independently solved several times. The initial population in each of these runs is generated in a random manner according to Rule 2. The first two truss examples are optimized by the CSS algorithm for 50 times, while performance comparisons of the CSS method in other examples is based on 20 evaluations. The algorithms are coded in MATLAB and structures are analyzed using the direct stiffness method.

3.5.1 A Benchmark Truss

The topology and nodal numbering of a 25-bar spatial truss structure, shown in Fig. 3.12, are known as a benchmark example in the field of structural optimization. The material density is considered as 0.1 lb/in^3 (2767.990 kg/m^3), and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa). The twenty-five members are categorized into eight groups as follows:

(1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21}, and (8) A_{22}–A_{25}

This spatial truss is subjected to two loading conditions shown in Table 3.3. Maximum displacement limitations of ±0.35 in (±8.89 mm) are imposed on every node in every direction, and the axial stress constraints vary for each group as shown in Table 3.4. The range of cross-sectional areas varies from 0.01 to 3.4 in^2 (0.6452–21.94 cm^2).

The CSS algorithm achieves the best solution after 7000 searches. However, the HBB–BC (Kaveh and Talatahari [12]) and HPSACO (Kaveh and Talatahari [4]) algorithms find the best solution after about 12,500 and 9875 analyses, respectively, which are 50 and 41 % more than the present work. The best weight of the CSS is 545.10 lb. Although the CSS approach has slightly worse performance than the improved methods IACS (Kaveh et al. [13]) and HPSACO (Kaveh and Talatahari [4]), it performs better than other algorithms GA (Rajeev and Krishnamoorthy [14]), PSO (Schutte and Groenwold [15]), and HS (Lee and Geem [8] when the best weight, the average weight, or the standard deviation are compared. Table 3.5 presents a comparison of the performance of the CSS algorithm and other metaheuristic algorithms.

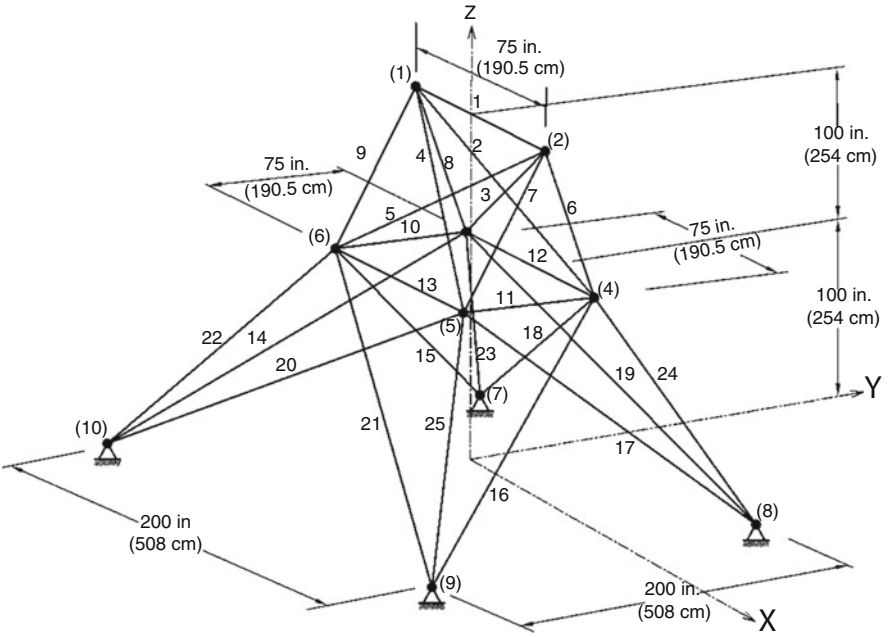


Fig. 3.12 Schematic of a 25-bar spatial truss [2]

Table 3.3 Loading conditions for the 25-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 3.4 Member stress limitation for the 25-bar spatial truss

Element group		Compressive stress limitations ksi (MPa)	Tensile stress limitations ksi (MPa)
1	A_1	35.092 (241.96)	40.0 (275.80)
2	$A_2 \sim A_5$	11.590 (79.913)	40.0 (275.80)
3	$A_6 \sim A_9$	17.305 (119.31)	40.0 (275.80)
4	$A_{10} \sim A_{11}$	35.092 (241.96)	40.0 (275.80)
5	$A_{12} \sim A_{13}$	35.092 (241.96)	40.0 (275.80)
6	$A_{14} \sim A_{17}$	6.759 (46.603)	40.0 (275.80)
7	$A_{18} \sim A_{21}$	6.959 (47.982)	40.0 (275.80)
8	$A_{22} \sim A_{25}$	11.082 (76.410)	40.0 (275.80)

Table 3.5 Performance comparison for the 25-bar spatial truss

Element group		Optimal cross-sectional areas (in^2)								
		Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	Kaveh et al.	Kaveh and Talatahari			Present work [2]	
		GA [14]	PSO [15]	HS [8]	IACS [13]	PSACO [4]	HPSACO [4]	HBB–BC [12]	in^2	cm^2
1	A_1	0.10	0.010	0.047	0.010	0.010	0.010	0.010	0.010	0.065
2	$A_2 \sim A_5$	1.80	2.121	2.022	2.042	2.052	2.054	1.993	2.003	12.923
3	$A_6 \sim A_9$	2.30	2.893	2.950	3.001	3.001	3.008	3.056	3.007	19.400
4	$A_{10} \sim A_{11}$	0.20	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065
5	$A_{12} \sim A_{13}$	0.10	0.010	0.014	0.010	0.010	0.010	0.010	0.010	0.065
6	$A_{14} \sim A_{17}$	0.80	0.671	0.688	0.684	0.684	0.679	0.665	0.687	4.432
7	$A_{18} \sim A_{21}$	1.80	1.611	1.657	1.625	1.616	1.611	1.642	1.655	10.677
8	$A_{22} \sim A_{25}$	3.0	2.717	2.663	2.672	2.673	2.678	2.679	2.660	17.161
Best weight (lb)		546	545.21	544.38	545.03	545.04	544.99	545.16	545.10	2424.7 N
Average weight (lb)		N/A	546.84	N/A	545.74	N/A	545.52	545.66	545.58	2426.8 N
Std dev (lb)		N/A	1.478	N/A	0.620	N/A	0.315	0.367	0.412	1.833 N
No. of analyses		N/A	9596	15,000	3520	28,850	9875	12,500	7000	

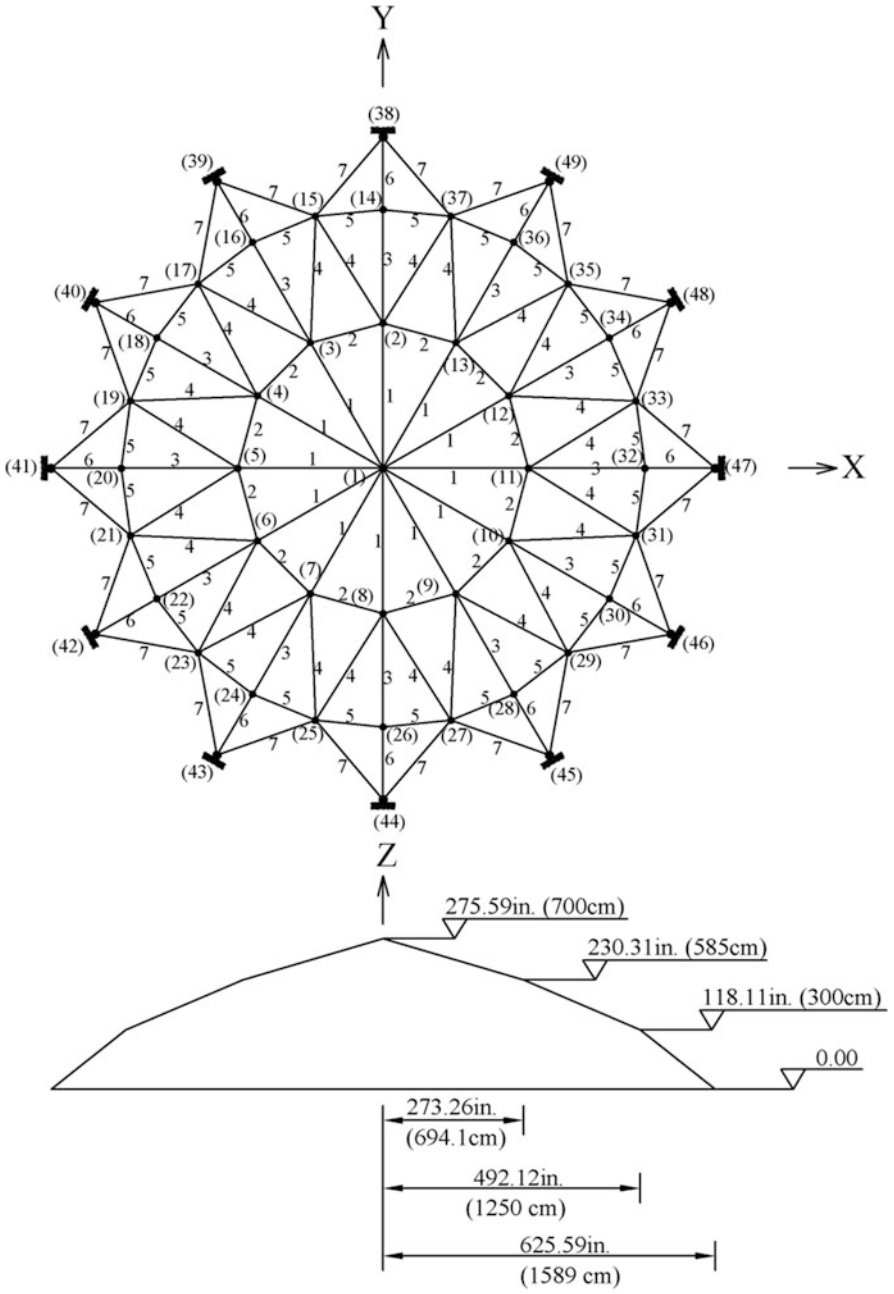


Fig. 3.13 Schematic of a 120-bar dome-shaped truss [2]

3.5.2 A 120-Bar Dome Truss

The topology and group numbers of a 120-bar dome truss are shown in Fig. 3.13. The modulus of elasticity is 30,450 ksi (210,000 MPa), and the material density is 0.288 lb/in^3 (7971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −13.49 kips (−60 kN) at node 1, −6.744 kips (−30 kN) at nodes 2 through 14, and −2.248 kips (−10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (2 cm^2), and the maximum cross-sectional area is taken as 20.0 in^2 (129.03 cm^2). The constraints are considered as follows:

1) Stress constraints (according to the AISC-ASD (1989) code):

$$\begin{cases} \sigma_i^+ = 0.6F_y & for \ \ \sigma_i \geq 0 \\ \sigma_i^- & for \ \ \sigma_i < 0 \end{cases} \tag{3.41}$$

   where σ_i^- is calculated considering the slenderness ratio [Eq. (3.34)].
2) Displacement limitations of ±0.1969 in (±5 mm) are imposed on all nodes in x, y, and z directions.

Table 3.6 illustrates the best solution vectors, the corresponding weights, and the required number of analyses for convergence in the present algorithm and some of other metaheuristic methods. Except IACS which uses two auxiliary mechanisms for searching, the CSS algorithm has the best convergence rates. Figure 3.14 shows the best and average convergence history for the results of the CSS. In addition, CSS and HPSACO find the best result among the other metaheuristics. A comparison of

Table 3.6 Performance comparison for the 120-bar dome truss

Element group		Optimal cross-sectional areas (in^2)						
		Kaveh et al.		Kaveh and Talatahari			Present work [2]	
		IACS [13]	PSOPC [4]	PSACO [4]	HPSACO [4]	HBB–BC [12]	in^2	cm^2
1	A_1	3.026	3.040	3.026	3.095	3.037	3.027	19.529
2	A_2	15.06	13.149	15.222	14.405	14.431	14.606	94.232
3	A_3	4.707	5.646	4.904	5.020	5.130	5.044	32.542
4	A_4	3.100	3.143	3.123	3.352	3.134	3.139	20.252
5	A_5	8.513	8.759	8.341	8.631	8.591	8.543	55.116
6	A_6	3.694	3.758	3.418	3.432	3.377	3.367	21.723
7	A_7	2.503	2.502	2.498	2.499	2.500	2.497	16.110
Best weight (lb)		33,320.52	33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	147,912 N
No. of analyses		3250	150,000	32,600	10,000	10,000	7000	

the allowable and existing stresses and displacements of the 120-bar dome truss structure using CSS is shown in Fig. 3.15. The maximum value for displacement is equal to 0.19689 in (5 mm) and the maximum stress ratio is equal to 99.98 %.

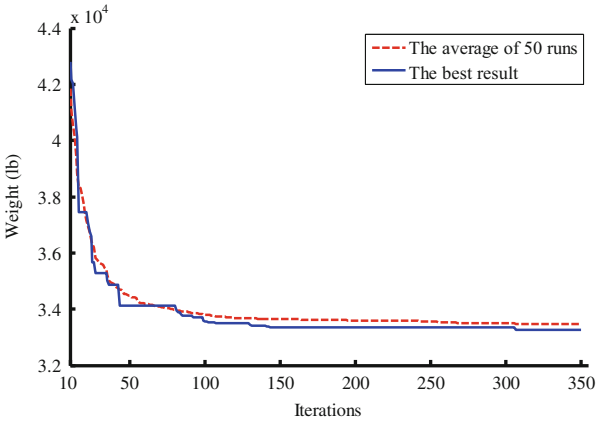


Fig. 3.14 Convergence history of the 120-bar dome-shaped truss for the CSS algorithm [2]

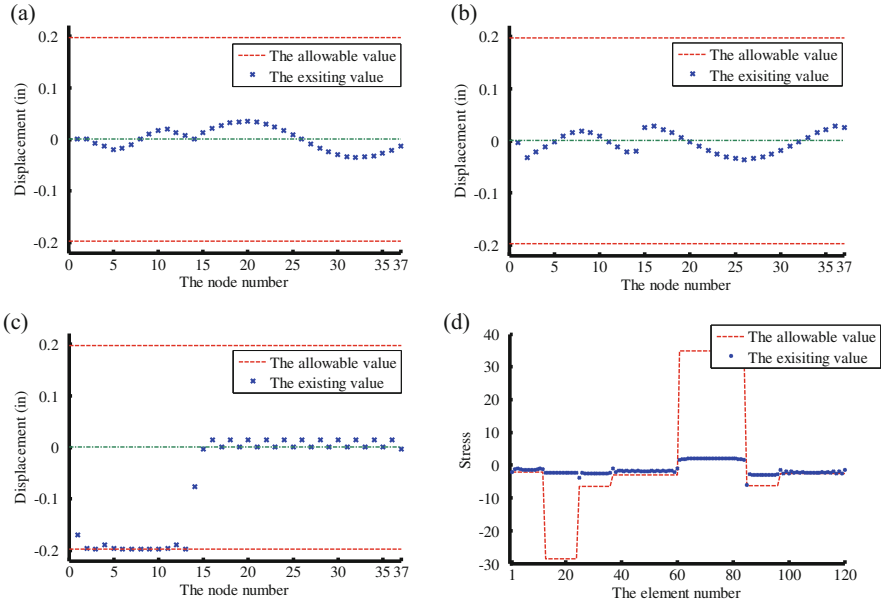


Fig. 3.15 Comparison of the allowable and existing constraints for the 120-bar dome-shaped truss using the CSS [2]. (**a**) Displacement in the *x* direction. (**b**) Displacement in the *y* direction. (**c**) Displacement in the *z* direction. (**d**) Stress

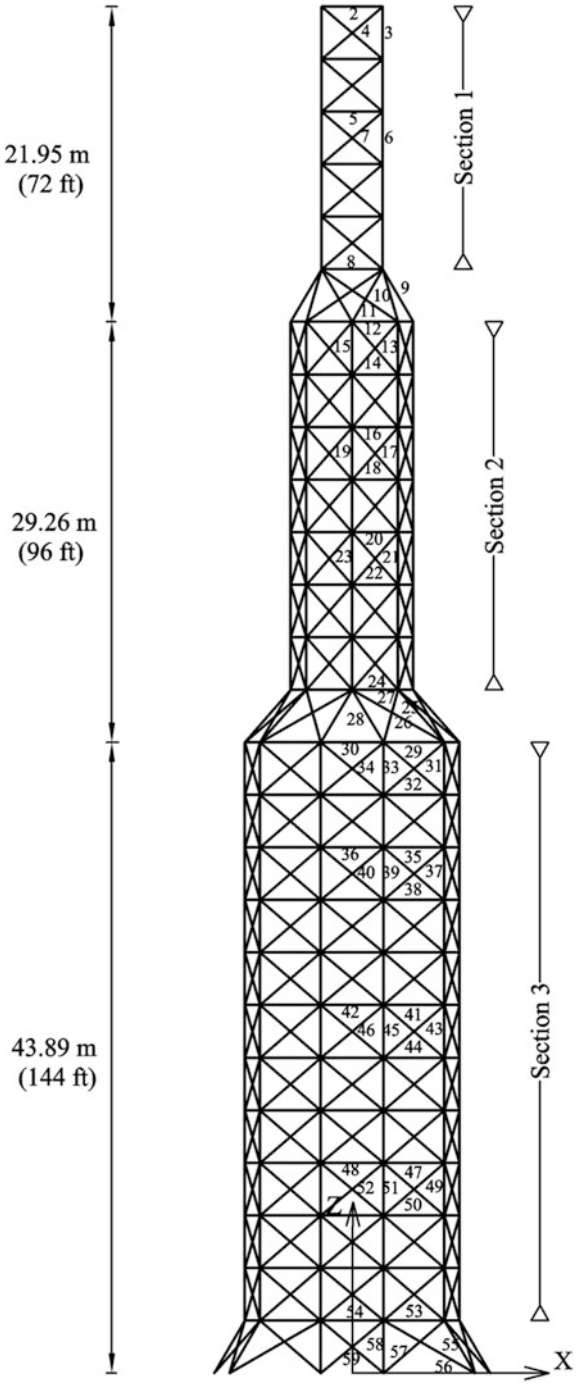


Fig. 3.16 Schematic of a 26-story-truss tower [2]

3.5.3 A 26-Story-Tower Space Truss

The 26-story-tower space truss containing 942 elements and 244 nodes is considered as a large-scale truss example. Fifty-nine design variables are used to represent the cross-sectional areas of 59 element groups in this structure, employing the symmetry of the structure. Figure 3.16 shows the geometry and the 59 element groups. The material density is 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa).

The members are subjected to the stress limits of ±25 ksi (172.375 MPa), and the four nodes of the top level in the x, y, and z directions are subjected to the displacement limits of ±15.0 in (38.10 cm) (about 1/250 of the total height of the tower). The allowable cross-sectional areas in this example are selected from 0.1 to 20.0 in^2 (from 0.6452 to 129.032 cm^2). The loading on the structure consists of:

1) The vertical load at each node in the first section is equal to −3 kips (−13.344 kN).
2) The vertical load at each node in the second section is equal to −6 kips (−26.688 kN).
3) The vertical load at each node in the third section is equal to −9 kips (−40.032 kN).
4) The horizontal load at each node on the right side in the x direction is equal to −1 kips (−4.448 kN).
5) The horizontal load at each node on the left side in the x direction is equal to 1.5 kips (6.672 kN).
6) The horizontal load at each node on the front side in the y direction is equal to −1 kips (−4.448 kN).
7) The horizontal load at each node on the back side in the x direction is equal to 1 kips (4.448 kN).

The CSS method achieved a good solution after 15,000 analyses and found an optimum weight of 47,371 lb (210,716 N). The best weights for the GA, PSO, BB–BC, and HBB–BC are 56,343 lb (250,626 N), 60,385 lb (268,606 N), 53,201 lb (236,650 N), and 52,401 lb (233,091 N), respectively. In addition, CSS has better performance in terms of the optimization time, standard deviation, and the average weight. Table 3.7 provides the statistic information for this example. The stress constraints are dominant in this example. The maximum value of stress ratio is equal to 96.7 %. Figure 3.17 compares the allowable and existing stresses in the elements for the CSS result. The convergence history is shown in Fig. 3.18. The final designs obtained by the CSS technique for this example is given in Table 3.8.

Table 3.7 Performance comparison for the 26-story-tower spatial truss

	Kaveh and Talatahari [12]				Present work [2]
	GA	PSO	BB–BC	HBB–BC	
Best weight (lb)	56,343 (250,626 N)	60,385 (268,606 N)	53,201 (236,650 N)	52,401 (233,091 N)	47,371 (210,716 N)
Average weight (lb)	63,223 (281,230 N)	75,242 (334,693 N)	55,206 (245,568 N)	53,532 (238,122 N)	48,603 (216,197 N)
Std dev (lb)	6640.6 (29,539 N)	9906.6 (44,067 N)	2621.3 (11,660 N)	1420.5 (6318 N)	950.4 (4227 N)
No. of analyses	50,000	50,000	50,000	30,000	15,000
Optimization time (s)	4450	3640	3162	1926	1340

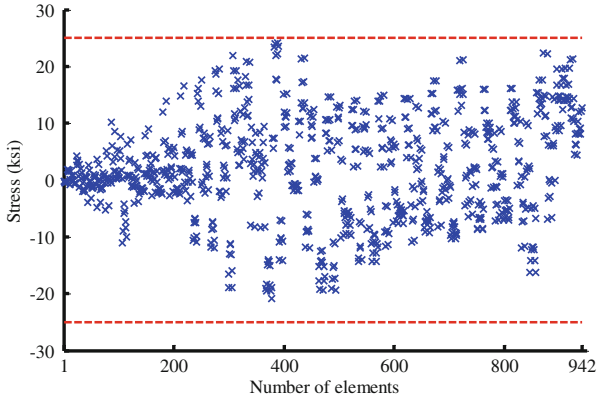


Fig. 3.17 Comparison of the allowable and existing stress constraints for the 26-story-tower truss using the CSS [2]

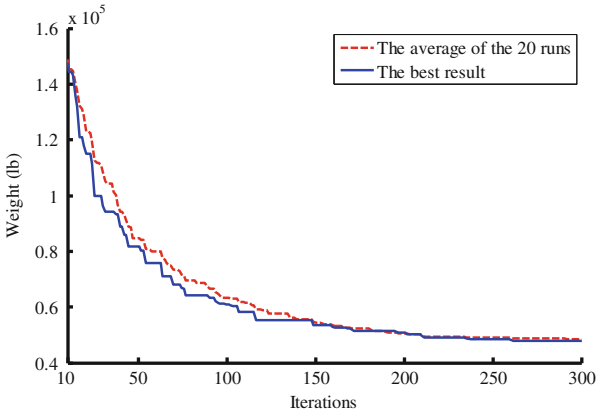


Fig. 3.18 Convergence history of the 26-story-tower truss for the CSS algorithm [2]

Table 3.8 The optimum design of the CSS algorithm for the 26-story-tower spatial truss

	Optimal cross-sectional areas (cm^2)							
	Members	Area		Members	Area		Members	Area
1	A_1	0.962	21	A_{21}	2.780	41	A_{41}	0.417
2	A_2	2.557	22	A_{22}	0.430	42	A_{42}	0.679
3	A_3	1.650	23	A_{23}	3.048	43	A_{43}	19.584
4	A_4	0.402	24	A_{24}	5.112	44	A_{44}	0.533
5	A_5	0.657	25	A_{25}	19.352	45	A_{45}	1.640
6	A_6	18.309	26	A_{26}	0.476	46	A_{46}	0.618
7	A_7	0.346	27	A_{27}	2.887	47	A_{47}	0.531
8	A_8	3.076	28	A_{28}	19.500	48	A_{48}	1.374
9	A_9	2.235	29	A_{29}	4.772	49	A_{49}	19.656
10	A_{10}	3.813	30	A_{30}	5.063	50	A_{50}	0.888
11	A_{11}	0.856	31	A_{31}	15.175	51	A_{51}	4.456
12	A_{12}	1.138	32	A_{32}	1.176	52	A_{52}	0.386
13	A_{13}	3.374	33	A_{33}	0.839	53	A_{53}	10.398
14	A_{14}	0.573	34	A_{34}	1.394	54	A_{54}	18.834
15	A_{15}	19.530	35	A_{35}	0.153	55	A_{55}	18.147
16	A_{16}	1.512	36	A_{36}	0.247	56	A_{56}	3.280
17	A_{17}	2.667	37	A_{37}	18.673	57	A_{57}	2.972
18	A_{18}	0.478	38	A_{38}	0.696	58	A_{58}	4.927
19	A_{19}	17.873	39	A_{39}	1.395	59	A_{59}	0.288
20	A_{20}	0.335	40	A_{40}	0.422			
Weight (N)		210,716						

3.5.4 An Unbraced Space Frame

A 10-story space steel frame consisting of 256 joints and 568 members is shown in Fig. 3.19. This problem has been formerly studied by Saka and Hasançebi [10] to evaluate the performance of an HS-based technique in real-size optimum design of steel frames considering ASD-AISC as the code of the practice.

The columns in a story are collected in three member groups as corner columns, inner columns, and outer columns, whereas beams are divided into two groups as inner beams and outer beams. The corner columns are grouped together as having the same section in the first three stories and then over two adjacent stories thereafter, as are inner columns, outer columns, inner beams, and outer beams. This results in a total of 25 distinct design groups.

The optimum design of the space frame described above is carried out using the CSS and compared with those of the simulated annealing (SA), evolution strategies (ESs), particle swarm optimizer (PSO), tabu search optimization (TSO), simple genetic algorithm (SGA), ant colony optimization (ACO), and harmony search (HS) methods (Saka and Hasançebi [10]). In each optimization technique, the

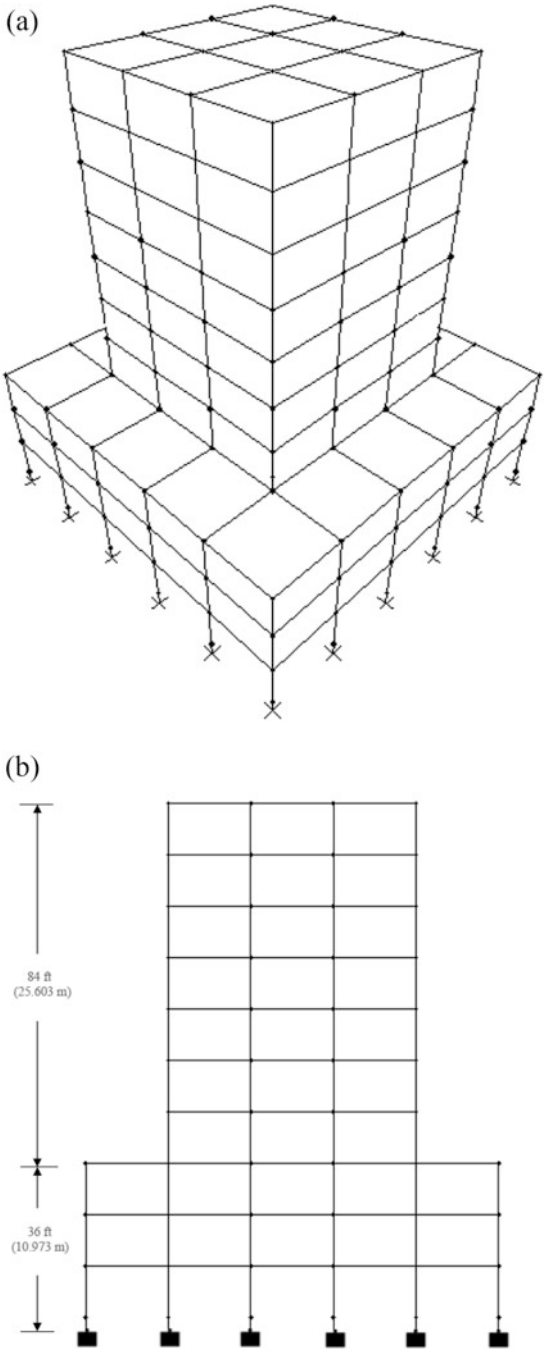
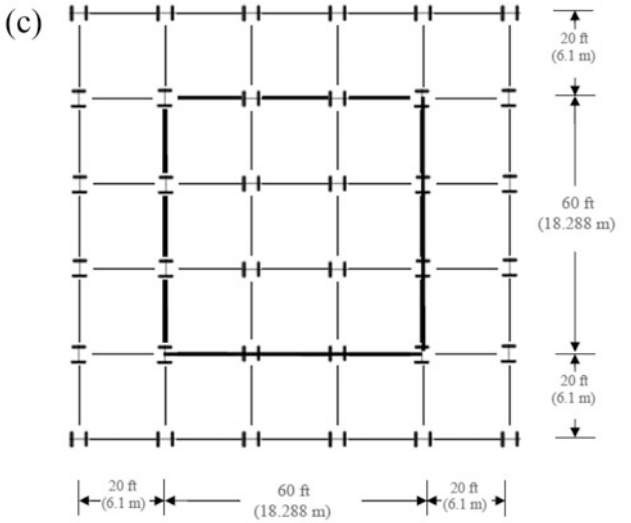


Fig. 3.19 Schematic of an unbraced space frame [2]. (**a**) Three-dimensional view, (**b**) elevation, (**c**) plan, (**d**) member grouping



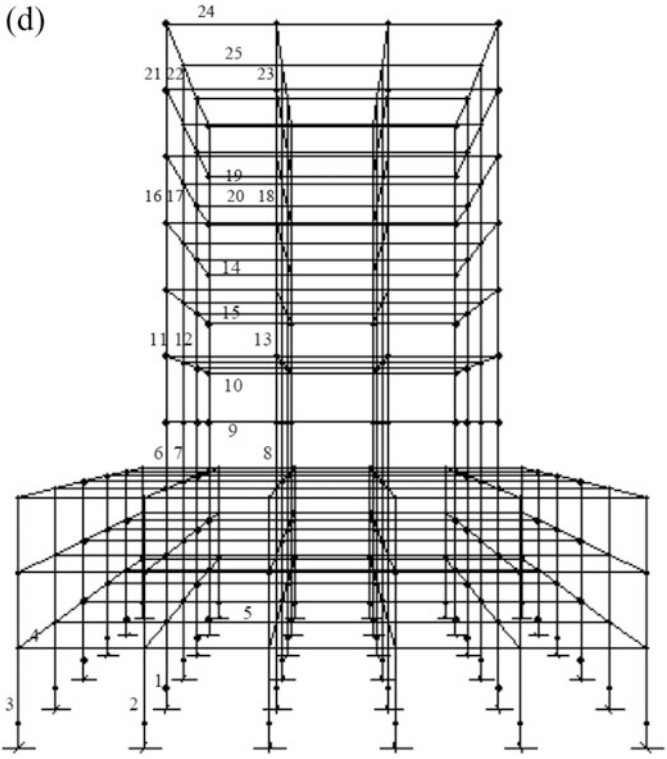


Fig. 3.19 (continued)

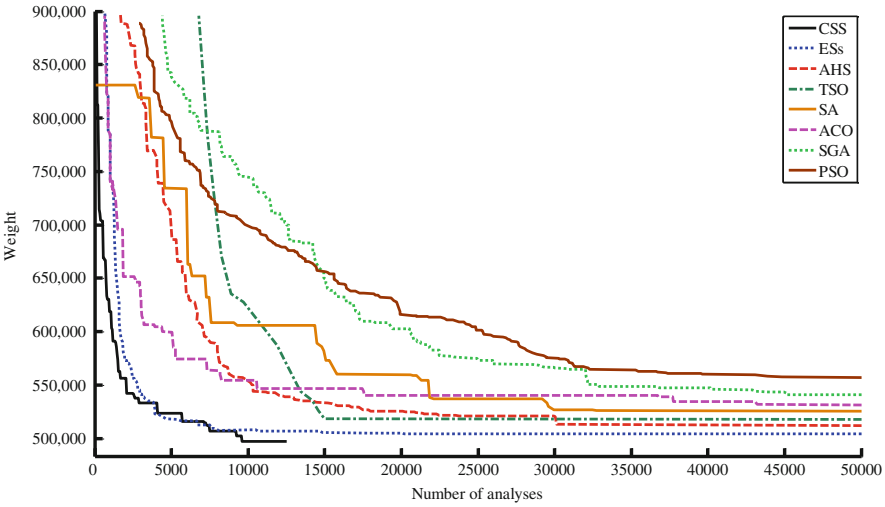


Fig. 3.20 Comparison of the convergence history for the unbraced space frame [2]

number of iterations was taken as 50,000, when ASD-AISC is used as the code of the practice. Our investigation shows that 12,500 analyses are sufficient as the maximum number of analyses for the CSS. This shows that the CSS can reach a similar result as the other methods with smaller number of analyses. The design history of each run by each technique is shown in Fig. 3.20.

The optimum design attained by the CSS method for this example is 225,654.0 kg, while it is 228,588.3 kg for the ESs. Among the metaheuristic algorithms, the adaptive harmony search algorithm is the third best which is 1.6 % heavier than the one obtained by evolutionary strategy algorithm. The optimum result for the TSO, SA, ACO, SGA, and PSO is 235,167.5 kg, 238,756.5 kg, 241,470.31 kg, 245,564.80 kg, and 253,260.23 kg, respectively. The minimum weights as well as W-section designations obtained by some of the best algorithms are provided in Table 3.9.

3.5.5 A Braced Space Frame

The second frame example considered in this chapter is a 36-story braced space steel frame consisting of 814 joints and 1860 members, as shown in Fig. 3.21 (Saka and Hasançebi [10]). An economical and effective stiffening of the frame against lateral forces is achieved through exterior diagonal bracing members located on the perimeter of the building, which also participate in transmitting the gravity forces.

The 1860 frame members are collected in 72 different member groups, considering the symmetry of the structure and the practical fabrication requirements. That

Table 3.9 Optimal design for the unbraced space frame

	Optimal W-shaped sections				
	Saka and Hasançebi [10]				
Element group	SA	TSO	AHS	ESs	Present work [2]
1	W14 × 193	W14 × 193	W14 × 176	W14 × 193	W14 × 132
2	W8 × 48	W8 × 48	W14 × 48	W8 × 48	W21 × 55
3	W8 × 40	W8 × 40	W10 × 39	W10 × 39	W12 × 40
4	W10 × 22	W10 × 22	W10 × 22	W10 × 22	W10 × 33
5	W21 × 44	W21 × 50	W24 × 55	W21 × 50	W21 × 50
6	W12 × 65	W10 × 54	W12 × 65	W10 × 54	W12 × 65
7	W14 × 145	W14 × 120	W14 × 145	W14 × 109	W14 × 99
8	W14 × 145	W14 × 159	W14 × 159	W14 × 176	W14 × 120
9	W24 × 65	W21 × 44	W14 × 30	W18 × 40	W21 × 44
10	W24 × 55	W18 × 40	W18 × 40	W18 × 40	W21 × 44
11	W10 × 49	W10 × 45	W10 × 54	W10 × 49	W14 × 61
12	W14 × 90	W14 × 90	W14 × 90	W14 × 90	W10 × 88
13	W14 × 120	W12 × 120	W14 × 120	W14 × 109	W14 × 99
14	W16 × 36	W12 × 44	W14 × 34	W14 × 30	W18 × 35
15	W16 × 40	W16 × 36	W18 × 40	W16 × 36	W12 × 50
16	W12 × 40	W10 × 33	W8 × 31	W12 × 45	W21 × 68
17	W12 × 65	W12 × 65	W12 × 65	W12 × 65	W16 × 57
18	W12 × 26	W14 × 34	W18 × 35	W10 × 22	W24 × 68
19	W12 × 72	W12 × 79	W12 × 79	W12 × 79	W16 × 36
20	W16 × 36	W14 × 30	W14 × 30	W14 × 30	W16 × 31
21	W8 × 24	W10 × 39	W10 × 22	W8 × 35	W10 × 33
22	W10 × 49	W12 × 45	W10 × 45	W10 × 39	W16 × 31
23	W8 × 24	W12 × 35	W8 × 31	W8 × 31	W8 × 28
24	W12 × 26	W6 × 20	W10 × 22	W8 × 18	W8 × 18
25	W12 × 26	W12 × 26	W12 × 26	W14 × 30	W10 × 26
Weight (kg)	238,756.5	235,167.5	232,301.2	228,588.3	225,654.0

is, the columns in a story are collected in three member groups as corner columns, inner columns, and outer columns, whereas beams are divided into two groups as inner beams and outer beams. The corner columns are grouped together as having the same section over three adjacent stories, as are inner columns, outer columns, inner beams, and outer beams. Bracing members on each facade are designed as 3-story deep members, and two bracing groups are specified in every six stories.

The minimum weight design of the frame is equal to 2301.69 t for the CSS algorithm, while it is 2383.60 and 4438.17 t for the adaptive harmony search and the simple harmony search algorithms, respectively. Figure 3.22 shows the design history graph obtained for this example. In the optimum design process, CSS finds the optimum design after 12,500 analyses, while ES needs 50,000 searches to determine the optimum solution.

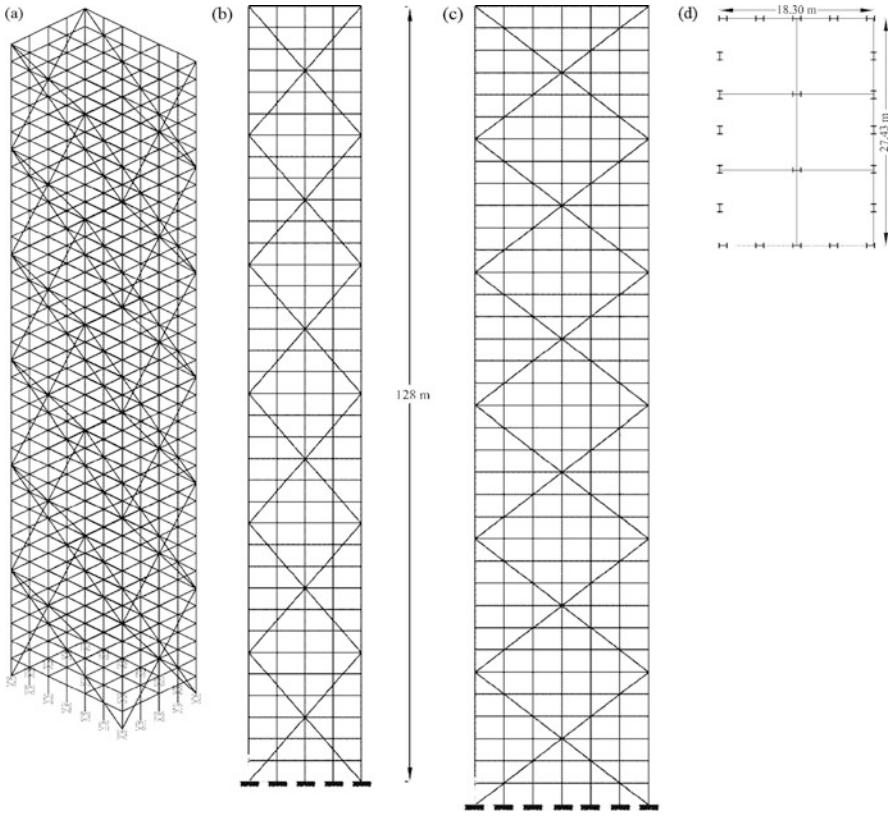


Fig. 3.21 Schematic of a braced space frame [2]. (**a**) 3D view of the frame, (**b**) Front view, (**c**) side view, (**d**) plan

3.6 Discussion

3.6.1 Efficiency of the CSS Rules

Solution of a number of design examples shows the superiority of the CSS algorithm to the other existing metaheuristics. To investigate the effect of some utilized rules, a number of the CSS-based algorithms are defined as follows:

Case 1: Rule 3 is changed as:

The kind of the electric forces between two charged particles is selected randomly.

Case 2: Rule 4 is changed as:

Any CP can act on another one; i.e., a bad CP can affect a good one and vice versa ($p_{ij} = 1$).

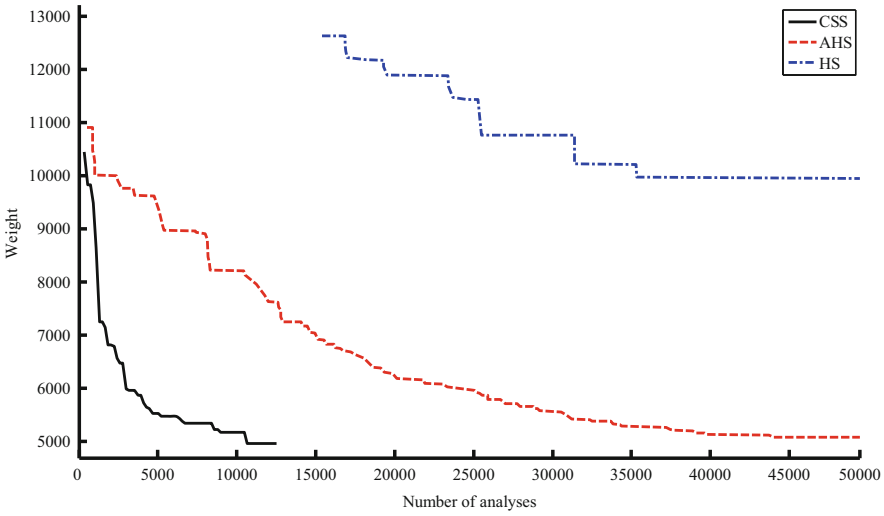


Fig. 3.22 Comparison of the convergence history for the braced space frame [2]

Table 3.10 Investigation on the performance of various CSS-based algorithms for the 25-bar truss in 50 runs

	Case 1	Case 2	Case 3	Case 4	Case 5
Best weight (lb)	551.31	551.10	545.99	546.28	550.55
Average weight (lb)	554.75	552.39	549.42	547.06	550.90
Std dev (lb)	1.210	0.885	1.467	0.707	0.686

Case 3: Rule 4 is changed as:
Only good CPs can attract bad CPs.
Case 4: Rule 5 is changed as:
Always $i_1 = 0$ and $i_2 = 1$.
Case 5: Rule 5 is changed as:
Always $i_1 = 1$ and $i_2 = 0$.

Table 3.10 shows the results of the 50 runs of the first example for each case. Comparing the result of Case 1 with the result of the original CSS (Table 3.5) confirms that considering repulsive forces between CPs reduces the power of the algorithm. Although when a good agent attracts a bad one, the exploitation ability for the algorithm is provided, and vice versa if a bad CP attracts a good CP, the exploration is provided, differences between the results of the Cases 2 and 3 with the original CSS indicated that when all bad agents attract good ones, a disorder will be created and when only good CPs attract bad ones the convergence will occur very soon and a complete search will not be performed. As a result, at least the computational cost to reach a good solution may increase for the condition of the Cases 2 and 3.

3.6.2 Comparison of the PSO and CSS

Both the CSS and the PSO are multi-agent algorithms in which the position of each agent is obtained by adding the agent's movement to its previous position; however, the movement strategies are different. In the PSO algorithm, each particle continuously focuses and refocuses on the effort of its search according to both local best and global best, while the CSS approach uses the governing laws from electrical physics and the governing laws of motion from the Newtonian mechanics to determine the amount and the direction of a charged particle's movement. The focus of the PSO is placed upon finding the direction of an agent movement, while the CSS method can determine not only the directions but also the amount of movements. In the PSO, the direction of an agent is calculated using only two best positions containing local best and global best. However, in the CSS the agent's direction is calculated based on the overall forces resulted by the best agents stored in the CM and some of the best current CPs. CSS can recognize the end of the global phase and change the movement updating equation for the local phase to have a better balance between the exploration and exploitation. One of the greatest disadvantages of the PSO approach is the existence of some difficulties in controlling the balance between the exploration and exploitation due to ignoring the effect of other agents (Kaveh and Talatahari [4]).

3.6.3 Efficiency of the CSS

CSS utilizes the Coulomb and Gauss laws to determine the direction and the amount of the movement of each agent and uses some laws of the Newtonian mechanics to complete the movement process. Compared to other metaheuristics, CSS has less computing cost and can determine the optimum result with a smaller number of analyses. Due to having a good balance between the exploration and exploitation, the performance of the CSS in both global search stage (initial iterations) and the local search stage (last iterations) is good. The comparison of the CSS results with those of the other metaheuristic shows the robustness of the present algorithm and demonstrates the efficiency of the algorithm to find optimum design of structures.

References

1. Kaveh A, Talatahari S (2010) A novel metaheuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
2. Kaveh A, Talatahari S (2010) Optimal design of truss structures via the charged system search algorithm. Struct Multidiscip Optim 37(6):893–911
3. Halliday D, Resnick R, Walker J (2008) Fundamentals of physics, 8th edn. Wiley, New York

4. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(5–6):267–283
5. Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Meth Appl Mech Eng 191 (11–12):1CA.245–287
6. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variable. J Constr Steel Res 65(8–9):1558–1568
7. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
8. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
9. American Institute of Steel Construction (AISC) (1989) Manual of steel construction—allowable stress design, 9th edn. AISC, Chicago, IL
10. Saka MP, Hasançebi O (2009) Design code optimization of steel structures using adaptive harmony search algorithm, Chapter 3. In Geem ZW (ed) Harmony search algorithms for structural design. Springer, Berlin. SCI 239:79–120
11. ASCE 7–05. Minimum design loads for building and other structures. Standards ASCE/SEI 7–05
12. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput Struct 87:1129–1140
13. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181
14. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. J Struct Eng ASCE 118(5):1233–1250
15. Schutte JJ, Groenwold AA (2003) Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 25:261–269