Chapter 4
Magnetic Charged System Search

4.1 Introduction

This chapter consists of two parts. In the first part, the standard magnetic charged system search (MCSS) is presented and applied to different numerical examples to examine the efficiency of this algorithm. The results are compared to those of the original charged system search method [1].

In the second part, an improved form of the MCSS algorithm, denoted by IMCSS, is presented and also its discrete version is described. The IMCSS algorithm is applied to optimization of truss structures with continuous and discrete variables to demonstrate the performance of this algorithm in the field of structural optimization [2].

4.2 Magnetic Charged System Search Method

One of the most recent metaheuristic algorithms is the charged system search (CSS) presented in Chap. 3, which uses the Coulomb and Gauss laws from physics and Newtonian laws from mechanics to guide the charged particles (CPs) to explore the locations of the optimum [3].

In this chapter, an improved CSS algorithm which is called magnetic charged system search (MCSS) is presented. The new algorithm utilizes the governing laws for magnetic forces and includes magnetic forces in addition to electrical forces. The movements of CPs due to the total force (Lorentz force) are determined using the governing laws of motion from the Newtonian mechanics.

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-46173-1_4

4.2.1 Magnetic Laws

4.2.1.1 Magnetic Fields

There is a relation between electric and magnetic forces, and these forces are called electromagnetic forces. The region surrounding any stationary or moving charged particle contains electric fields. In addition to electric field, the region surrounding any moving charged particle also contains magnetic fields. The existence of the magnetic field near the moving charged particles was Oersted's discovery in 1819. He has shown that a compass needle was deflected by a current-carrying conductor. Shortly after this discovery, Biot and Savar proposed a mathematical expression so-called Biot–Savar law that provides the magnitude of magnetic field at any point of the space in terms of the electric current that produces the field (Fig. 4.1). Biot–Savar law is expressed [4] as:

$$d\mathbf{B} = \frac{\mu_0}{4\pi}\frac{Id\mathbf{s} \times \hat{\mathbf{r}}}{r^2} \tag{4.1}$$

Here, $d\mathbf{B}$ is the magnetic field at point P, μ_0 is a constant called the permeability of free space, and r is the distance between $d\mathbf{s}$ to P.

Consider a straight wire with radius of R carrying electric current of magnitude I which is uniformly distributed through the cross section of the wire (Fig. 4.2a). By utilizing the Biot–Savar law, the magnetic field produced by wire at a point like P outside the wire can be determined as:

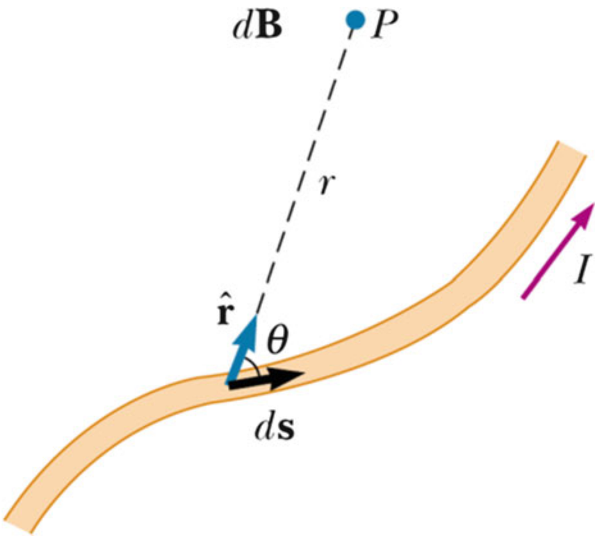


Fig. 4.1 The magnitude of the magnetic field $d\mathbf{B}$ at point P due to current I through a length element $d\mathbf{s}$ given by the Biot–Savar law [1]

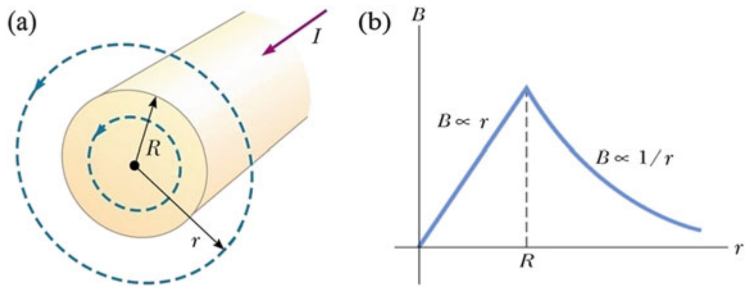


Fig. 4.2 (**a**) A wire carrying electric current *I* that is uniformly distributed in its cross section. (**b**) A plot of distribution of magnetic field produced by a wire in the space [1]

$$B = \frac{\mu_0 I}{2\pi r} \quad when: \; r \geq R \tag{4.2}$$

The magnitude of the magnetic field inside the wire can be obtained using Ampère's law:

$$B = \left(\frac{\mu_0}{2\pi}\frac{I}{R^2}\right) \times r \quad when: \; r < R \tag{4.3}$$

With this formulation for magnetic field, the magnitude of the field inside the wire increases linearly from $r = 0$ to $r = R$ ($B \propto r$), and outside of the wire, it is inversely proportional to the distance ($B \propto 1/r$) and decreases by increasing the distance. When $r = R$, Eqs. (4.2) and (4.3) have an overlap, and both give identical magnitude for the magnetic field. A plot of these two equations from Ref. [4] is shown in Fig. 4.2b.

If there are many wires in a space, in order to calculate the total magnitude of the magnetic field in a specified point, the equivalent magnetic field should be calculated by considering the principle of superposition and summing the magnetic fields produced by each wire. Therefore, the total magnetic field at a specified point *P*, due to a group of wires, can be obtained as:

$$B_P = \sum_{i=1}^{n} B_{ip} \tag{4.4}$$

where B_P is the total magnetic field at point *P*, *n* is the number of wires in the space, and B_{ip} is the magnetic field created by the *i*th wire at point *P* which can be expressed as:

$$B_{ip} = \begin{cases} \dfrac{\mu_0 I}{2\pi r} & for \quad r \geq R \\ \left(\dfrac{\mu_0}{2\pi}\dfrac{I}{R^2}\right) \times r & for \quad r < R \end{cases} \tag{4.5}$$

4.2.1.2 Magnetic Forces

When a charged particle moves in a magnetic field, a magnetic force $\mathbf{F}_B$ will be imposed on it. Experiments on charged particles moving in a magnetic field result in the following:

- The magnitude of the magnetic force $\mathbf{F}_B$ exerted on the charged particle is proportional to the charge q and to the speed v of the particle.
- The magnitude and direction of the magnetic force $\mathbf{F}_B$ depend on the velocity of the particle and magnitude and direction of magnetic field $\mathbf{B}$.

By summarizing these observations, an expression for calculating the magnetic force is obtained [4] as:

$$\mathbf{F}_B = q\mathbf{v} \times \mathbf{B} \tag{4.6}$$

where $\mathbf{B}$ is the magnetic field exerted on the particle. Here, the only source of the magnetic field is the magnetic field produced by the wires. Thus, the magnitude of the $\mathbf{B}$ can be calculated using Eq. (4.5).

4.2.2 A Brief Introduction to Charged System Search Algorithm

The charged system search (CSS) algorithm, as explained in Chap. 3, takes its inspiration from the physical laws governing a group of charged particles (CPs). These charge particles are sources of the electric fields, and each CP can exert electric force on other CPs. Using the governing laws of motion from the Newtonian mechanics, the movement of each CP due to the electric force can be determined. The CSS algorithm is summarized in a step-by-step form as follows:

Step 1. Initialization

The initial positions of the CPs are randomly determined using a uniform source, and the initial velocities of the particles are set to zero. A memory is used to save a number of best results. This memory is called the Charged Memory (CM).

Step 2. Determination of electric forces and the corresponding movements

- Force Determination. Each charged particle imposes electric forces on the other CPs according to the magnitude of its charge. The charge of each CP is:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst} \tag{4.7}$$

where fit(i) is the objective function value of the ith CP and *fitbest* and *fitworst* are the so far best and worst fitness among all of the CPs, respectively.

In addition to electric charge, the magnitude of the electric forces exerted on the CPs is depended on their separation distance that is,

$$r_{ij} = \frac{\|\mathbf{X}_i - \mathbf{X}_j\|}{\|(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best}\| + \varepsilon} \tag{4.8}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the *i*th and *j*th CPs and r_{ij} is the separation distance of these CPs. $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number to prevent singularity.

The probability of the attraction of the *i*th CP by the *j*th CP is expressed as:

$$p_{ij} = \begin{cases} 1 \Leftrightarrow \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand, or, fit(j) > fit(i) \\ 0 \Leftrightarrow else. \end{cases} \tag{4.9}$$

The electric resultant force $\mathbf{F}_{E,j}$, acting on the *j*th CP, can be calculated by the following equation:

$$\mathbf{F}_{E,j} = q_j \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{r_{ij}^{\,2}} \cdot w_2 \right) \cdot p_{ji} \cdot (\mathbf{X}_i - \mathbf{X}_j), \left\langle \begin{array}{l} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{array} \right. \tag{4.10}$$

- Movements Calculations. According to the determined forces, each CP moves to its new position, and attain velocity as:

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old}, \tag{4.11}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{4.12}$$

where $rand_{j1}$ and $rand_{j2}$ are two random numbers that are uniformly distributed in the range (0,1). k_a is the acceleration coefficient, k_v is the velocity coefficient, and m_j is the mass of particle that is considered equal to q_j. The magnitudes of the k_a and k_v are set to 0.5 which are linearly functions:

$$k_a = 0.5(1 + iter/iter_{\max}), \quad k_v = 0.5(1 - iter/iter_{\max}) \tag{4.13}$$

where *iter* is the current iteration number, and $iter_{max}$ is the maximum number of iterations.

Step 3. Charged Memory (CM) Updating

If among all of the new CPs, there are CPs with better objective function values than the worst ones in the CM, these should be included in the CM, and the worst ones in the CM are excluded from the CM.

Step 4. Checking the Termination Criteria

Steps 2 and 3 are reiterated until one of the specified terminating criteria is satisfied.

4.2.3 Magnetic Charged System Search Algorithm

4.2.3.1 Combination of Magnetic and Electric forces

The inspiration of the standard CSS algorithm is based on a group of charged particles that exert electric forces on each other based on their charges and their separation distances. After computing the electric forces, each particle moves, and its movement is calculated by using the governing laws of motion from the Newtonian mechanics. Therefore, we have charged particles that move in the search space. In physics, it has been shown that when a charged particle moves, it produces magnetic field. This magnetic field can exert a magnetic force on other charged particles. Thus, in addition to the electric forces, we should consider magnetic forces. In physics, when a charged particle moves with velocity $\mathbf{v}$ in the presence of both an electric field $\mathbf{E}$ and a magnetic field $\mathbf{B}$, it experiences both an electric force $q\mathbf{E}$ and a magnetic force $q\mathbf{v} \times \mathbf{B}$. The total force, known as the Lorentz force [4], exerted on the charged particle is:

$$\sum \mathbf{F} = \mathbf{F}_B + \mathbf{F}_E = q\mathbf{v} \times \mathbf{B} + q\mathbf{E} = q \cdot (\mathbf{v} \times \mathbf{B} + \mathbf{E}) \tag{4.14}$$

where $\mathbf{F}$ is the Lorentz force. Thus, MCSS considers the magnetic force as an additional force with the purpose of making the new algorithm closer to the nature of the movement of charged particles. From optimization point of view, this new force records additional information about the movement of the CPs, and it improves the performance of the standard CSS.

4.2.3.2 MCSS Algorithm

The MCSS algorithm is based on its original version, standard CSS. The difference between these two algorithms is that CSS only considers the electric force, but MCSS includes magnetic forces besides electric forces. The main structure of the algorithm is the same as the standard CSS, but in MCSS changes are made in part of the algorithm where the forces are computed. By using the aforementioned physical laws about magnetic fields and forces, the magnetic forces are determined. Each solution candidate $\mathbf{X}_i$ known as charged particle (CP) contains electrical charge. These CPs produce electric fields and exert electric forces on each other. When a

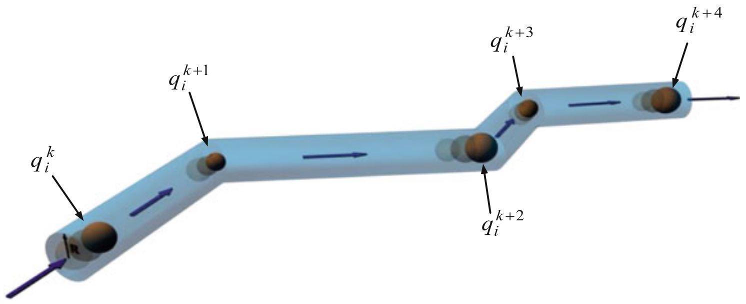


Fig. 4.3 The schematic view of a virtual wire (movement path of a CP), q_i^k is the charge of the *i*th CP at end of the *k*th movement (*k*th iteration) [1]

CP moves, it creates a magnetic field in the space, and this magnetic field imposes magnetic forces on other CPs.

As explained previously, the source of the magnetic fields is the movement of the CPs. For computing these fields, we assumed that CPs move in virtual straight wires with radius of R. Thus, the path of movement of each particle consists of straight wires. These straight wires change their directions by each movement of the CPs, but during the movement, each wire remains straight (Fig. 4.3). The place that a wire changes its direction is the position of the CP at the end of its movement. When the CP starts a new movement, the direction of its movement may differ from its previous one, so the direction of the wire which includes the CP during its movement also changes. According to magnetic laws presented in Sect. 4.2.1, a conducting wire carrying electric current can create magnetic fields in the space. Now our virtual wires contain charged particles that move on them. By each movement of the CPs, their charges are altered, so during the movement the magnitude of the charge is not constant and changes. This movement of CPs can be comprehended as an electric current in the virtual wire. The current of a wire is the rate at which charge flows through one specified cross section of the wire. If Δq is the amount of charge that passes through this area in a time interval Δt, the average current I_{avg} will be equal to the charge that passes through the cross section per unit time:

$$I_{avg} = \frac{\Delta q}{\Delta t} \tag{4.15}$$

Since the time intervals of each movement are set to unity, the average current will be equal to the variation of the charge. For computing the variation of the charges, we consider the start and the end points of the movement of CPs. By taking these assumptions into account, Eq. (4.13) can be written as:

$$\left(I_{avg}\right)_{ik} = q_i^k - q_i^{k-1} \tag{4.16}$$

where $(I_{avg})_{ik}$ is the average current in the *i*th wire (corresponding to the *i*th CP) in the *k*th movement (iteration), and q_i^{k-1} and q_i^k are the charges of the *i*th CP at the start and end of its *k*th movement, respectively. Equation (4.14) shows that by this definition for the electric current, the concept of quantity represents the variation of the objective function of each CP in each movement. By this definition, the electric current can be both positive and negative values. A positive one indicates that the movement produced an improvement in the charge of the CP. In other words, since the charge of a CP is a quantity of its quality or objective function value, a positive electric current means an improvement, and a negative electric current means a deterioration in the quality of the CP.

The charges of the CPs are defined by Eq. (4.6). This expression for computing electric charges results in values between 0 and 1. This is due to normalization of the objective function of each CP in that expression. Therefore, the charges of the worst and best CP are always zero and unity, respectively. Now, consider the situation that the worst CP moves in the search space, at the end of the movement, it may attain a better objective function value, but it may still be the worst CP, so its charge will still be zero. This means that there may be some situations that the objective function of a CP improves but its charge does not change because charge is a relative quantity. It seems necessary to modify the electric current expression in a way that the concept of electric current is saved and the aforementioned problem is solved. In relation with this problem, two alternative expressions for computing electric current are proposed. The first one is:

$$\left(I_{avg}\right)_{ik} = \frac{q_{i,k} - q_{i,k-1}}{q_{i,k}} \tag{4.17}$$

where $q_{i,k}$ and $q_{i,k-1}$ are the charge of the *i*th CP at the start of the *k*th and $k-1$th iterations, respectively. This equation gives a normalized value for the variation of the *i*th *CP*. The second proposed relation is expressed as:

$$\left(I_{avg}\right)_{ik} = sign\left(df_{i,k}\right) \times \frac{\left|df_{i,k}\right| - df_{\min,k}}{df_{\max,k} - df_{\min,k}} \tag{4.18}$$

$$df_{i,k} = fit_k(i) - fit_{k-1}(i) \tag{4.19}$$

where $df_{i,k}$ is the variation of the objective function in the *k*th movement (iteration). $fit_k(i)$ and $fit_{k-1}(i)$ are the values of the objective function of the *i*th CP at the start of the *k*th and $k-1$th iterations, respectively. The quantity $df_{i,k}$ can attain both positive and negative values. If we consider absolute values of *df* for all of the current CPs, $df_{\max,k}$ and $df_{\min,k}$ will be the maximum and minimum values among these absolute values of *df*, respectively. Therefore, $df_{\max,k}$ and $df_{\min,k}$ are always positive quantities. It should be noted that here the second expression [Eqs. (4.16) and (4.17)] is utilized for the computation of the electric current.

For computing the magnetic field in place of each particle, one must compute the distance of that particle from the virtual wire. This distance is assumed to be the same as Eq. (4.7). Thus, r_{ij} now means the distance between the ith wire and ith virtual CP to the jth charged particle.

In the expression for computing the magnetic force (Eq. (4.5)), we should consider the velocity of the movement of CPs. In this case, due to the movements of both CPs (CP in the virtual wire and CP in the space) the relative velocity, $\mathbf{v}_{rel}$, is considered as:

$$\mathbf{v}_{rel} = \frac{\mathbf{X}_i - \mathbf{X}_j}{\Delta t} \tag{4.20}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and jth CPs and the Δt is the time step that is set to unity. Therefore the relative velocity can be rewritten as:

$$\mathbf{v}_{rel} = \mathbf{X}_i - \mathbf{X}_j \tag{4.21}$$

By considering these assumptions, the magnetic force $\mathbf{F}_{\mathbf{B},ji}$ exerted on the jth CP due to the magnetic field produced by the ith virtual wire (ith CP) can be expressed as:

$$\mathbf{F}_{\mathbf{B},ji} = q_j \cdot \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2 \right) \cdot pm_{ji} \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \left\langle \begin{array}{l} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R \end{array} \right. \tag{4.22}$$

where q_i is the charge of the ith CP, R is the radius of the virtual wires, I_i is the average electric current in each wire, and pm_{ji} is the probability of the magnetic influence (attracting or repelling) of the ith wire (CP) on the jth CP. This term can be computed by the following expression:

$$pm_{ji} = \begin{cases} 1 \Leftrightarrow fit(i) > fit(j) \\ 0 \Leftrightarrow else \end{cases} \tag{4.23}$$

where $fit(i)$ and $fit(j)$ are the objective values of the ith and jth CPs, respectively. This probability determines that only a good CP can affect a bad CP by the magnetic force. This magnetic probability is slightly different from the electric probability expressed by Eq. (4.8). The electric probability considers a chance for both good and bad CPs to attract each other, but the magnetic probability has allocated this chance only to good CPs. The purpose of this definition of magnetic probability is to reduce the parasite magnetic fields and reinforce the efficiency of the magnetic forces.

Investigating different terms of the magnetic force shows how this force can help the standard CSS algorithm. If I_i, electric current in the ith virtual wire, is negative, according to the concept of the electric current, a negative value means that the ith CP did not experience an improvement in the value of its objective function. Thus, a negative value will be multiplied by $\left(\mathbf{X}_i - \mathbf{X}_j\right)$, so this produces a repelling force. In this case, it is an ideal force. On the other hand, if the ith CP experiences an improvement in its movement, it will attract the jth CP. From optimization point of view, this kind of force can help the algorithm. It stores and applies the information of the movement of each CP. This information is lost in the standard CSS, but MCSS utilizes this information and increases the efficiency of algorithm.

Now by considering the group of the charged particles, the resultant magnetic force acting on each CP can be calculated using the following expression:

$$\mathbf{F}_{\mathbf{B},j} = q_j \cdot \sum_{i,i\neq j} \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2 \right) \cdot pm_{ji} \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \left\langle \begin{array}{l} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{array} \right. \tag{4.24}$$

where $\mathbf{F}_{\mathbf{B},j}$ is the resultant magnetic force exerted on the jth charged particle.

The quantity R is the radius of the virtual wires, and if a charged particle is placed outside or inside of a virtual wire, the magnetic force that is exerted on it is computed differently. With this formulation for magnetic force, in the early iterations where the agents are far from each other, their distances will be large values, and the magnetic force in this case will be inversely proportional to the distances. As a result, the magnitude of the magnetic force is relatively small, and this feature of the algorithm provides a good situation for search ability of the CPs in the early iterations which is ideal for optimization problems. After a number of iterations, CPs search the search space and most of them will be gathered in a small space. Now, the distances between CPs are decreased and a local search starts. In this case, if the magnetic force computed based on the inverse relation between distances, the magnitude of the forces will be increased due to decrease of the distances. These large forces may prevent the convergence of the algorithm in the local search. One of the solutions that can be proposed is that when the distances are relatively small, the magnetic force should be computed using the linear formulation of magnetic fields (Eq. (4.3)). This means that the formulation of the magnetic force for global and local phases should be separated (Eq. (4.24)). A suitable value for R in Eq. (4.24) can be unity. However, by more investigating in the magnetic force formulation, it could be understood that the aforementioned problem can be solved automatically. If the value of the R is taken as zero, all of the magnetic fields produced by virtual wires can be calculated based on Eq. (4.2). Using this equation for small distances gives large values for the magnetic field, but when the values of distances are small, it means that the CPs are collected in a small space and their movements are small (local search). Thus, both $\mathbf{X}_i - \mathbf{X}_j$ and I_i are small values. By considering Eq. (4.24) for calculating the magnetic forces, it can be noted that a large value is multiplied by two small values, so the final value (magnetic force) is a normal value which helps the algorithm. Due to the ease of implementation and better convergence rate, the second solution is selected in this part and the magnetic force is revised in Eq. (4.25).

The term pm_{ji}, in the expression for calculating the magnetic force, provides competition ability for the CPs. According to the concept of the magnetic force in this algorithm, when a CP experiences an improvement in its value of the objective function, it should attract other CPs, regardless of its previous and current charge. However, by considering the term pm_{ji}, CPs with larger charges have more tendency to attract other CPs. The reason is that by considering this term, the redundant and parasite magnetic fields made by bad CPs are eliminated and it helps the efficiency of the algorithm.

It should be noted that in implementing the MCSS, the part of CSS algorithm related to computing forces should be changed. Both magnetic and electric forces should be computed and superposed. The Lorentz force (total force) will be expressed as:

$$\sum \mathbf{F}_j = \mathbf{F}_{B,j} + \mathbf{F}_{E,j} = q_j \sum_{i,i \neq j} \left(\frac{I_i}{r_{ij}} \cdot pm_{ji} + \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{{r_{ij}}^2} \cdot w_2 \right) \cdot p_{ji} \right) \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \left\langle \begin{array}{l} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{array} \right. \tag{4.25}$$

where $\mathbf{F}_j$ is the resultant Lorentz force (total force) acting on the jth CP.

Consider the ith CP among all of the CPs; this CP has a charge which is larger than the charges of a number of CPs. Considering the rules of the CSS, the ith CP attracts all other CPs that have smaller charges. After computing the electric forces, all of the CPs move around the search space. Now, the ith CPs also moved to a new position. In this movement, the ith particle may experience deterioration in its objective function value. Due to this decrease, the new charge of the ith particle will be decreased, but its charge may still be larger than a number of CPs. According to the CSS algorithm, the ith particle still attracts all other CPs with smaller charges regardless of the failure of the ith CP in its last movement. From one perspective, this is logical that a good CP can attract bad CPs. This feature ensures the competition ability of the algorithm. However, from another point of view, if no attention is paid to the success or failure of the CPs in their last movement, a lot of useful information in optimization process will be lost. Thus, in the MCSS algorithm, magnetic forces are included to prevent the loss of this kind of information from which the algorithm benefits. By this concept, the ith particle which has experienced a failure in its last movement exerts repelling magnetic forces on the other CPs. In this situation, the direction of the magnetic forces and electrical ones that are exerted on CPs by the ith CP are opposite.

That was a special case that the magnetic and electric forces were against each other. Most of the times, the magnetic and electric forces are in the same direction, and they reinforce the effect of each other. Consequently, the exploitation ability of the algorithm is mostly reinforced. Because of this increase in exploitation ability, we can slightly modify k_v in Eq. (4.11) to increase the exploration ability of the algorithm. In fact, the MCSS algorithm guides the CPs with more information, and the efficiency of the algorithm including a fast convergence is improved, and in comparison with the standard CSS, a better exploitation and exploration are provided.

4.2.4 Numerical Examples

In order to ensure the efficient performance of the MCSS algorithm, some numerical examples are solved, and the results are compared to those of the standard CSS algorithm. The examples consist of 18 mathematical functions. The numerical examples are presented in Sect. 5.1. In Sect. 5.2 the results of the MCSS are presented and compared to those of the CSS and other optimization algorithms in the literature. Finally, in Sect. 5.3 three well-studied engineering design problems are solved by MCSS, and the results are compared to those of the CSS.

4.2.4.1 Mathematical Benchmark Functions

Comparison Between MCSS, CSS, and a Set of Genetic Algorithms

In this section, some mathematical benchmarks are chosen from Ref. [5] and optimized using the MCSS algorithm. The description of these mathematical benchmarks is provided in Table 4.1.

Numerical Results

In this section, the numerical results of optimization for the mathematical benchmarks are presented. In this investigation, some parameters of the algorithm such as HMCR, PAR, CM size (CMS), the number of CPs, and the maximum number of iteration are modified. For eliminating the effect of such parameters in studying the performance of the algorithm, these parameters are considered the same as those of Ref. [6]. It should be noted that the number of CPs is set to 20, and the maximum number of iterations is considered as 200 for both CSS and MCSS algorithm. In Table 4.2, the results of the MCSS are compared to the results obtained by the CSS from Ref. [6], and GA and some of its variants are derived from [5]. For a fair comparison between MCSS and CSS, the random initial solutions of each runs are the same. The numbers in Table 4.2 indicate the average number of function evaluation from 50 independent runs. The numbers in parenthesis demonstrate the fraction of the unsuccessful to successful runs. The absence of a parenthesis means that the algorithm was successful in all of the runs. Each run of the algorithm is successful when that run determines a local minimum with predefined accuracy, i.e., $\varepsilon = \left|f_{\min} - f_{final}\right| = 10^{-4}$. The results verify the efficiency of the MCSS algorithm compared to the CSS and other genetic algorithms. The existence of the magnetic forces in the MCSS provides a better exploration and exploitation for the algorithm. Thus, the convergence is speeded up. One of the important features of the MCSS algorithm is its ability to converge to the desired optimum with a few number of CPs and a small value for maximum number of iterations. The difference between the CSS algorithm and MCSS algorithm becomes more obvious when the number of CPs and the number of iterations are set to small values. Thus, another

Table 4.1 Description of the mathematical benchmarks

Function name	Interval	Function	Global minimum
Aluffi-Pentini	$\mathbf{X} \in [-10, \quad 10]^2$	$f(\mathbf{X}) = \frac{1}{4}{x_1}^4 - \frac{1}{2}{x_1}^2 + \frac{1}{10}x_1 + \frac{1}{2}{x_2}^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in [-100, \quad 100]^2$	$f(\mathbf{X}) = {x_1}^2 + 2{x_2}^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in [-50, \quad 50]^2$	$f(\mathbf{X}) = {x_1}^2 + 2{x_2}^2 - \frac{3}{10}\cos(3\pi x_1) \cdot \cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10, \quad 10]^2$	$f(\mathbf{X}) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	0.0
Branin	$0 \le x_2 \le 15,$ $-5 \le x_1 \le 10$	$f(\mathbf{X}) = \left(x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{\pi}x_1\right)^2 + 10 \cdot \left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	0.397887
Camel	$\mathbf{X} \in [-5, \quad 5]^2$	$f(\mathbf{X}) = 4{x_1}^2 - 2.1{x_1}^4 + \frac{1}{3}{x_1}^6 + x_1x_2 - 4{x_2}^2 + 4{x_2}^4$	−1.0316
Cb3	$\mathbf{X} \in [-5, \quad 5]^2$	$f(\mathbf{X}) = 2{x_1}^2 - 1.05{x_1}^4 + \frac{1}{6}{x_1}^6 + x_1x_2 + {x_2}^2$	0.0
Cosine mixture	$\mathbf{X} \in [-1, \quad 1]^n, n = 4$	$f(\mathbf{X}) = \sum_{i=1}^{n} {x_i}^2 - \frac{1}{10}\sum_{i=1}^{n} \cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in [-5.12, \quad 5.12]^3$	$f(\mathbf{X}) = {x_1}^2 + {x_2}^2 + {x_3}^2$	0.0
Exponential	$\mathbf{X} \in [-1, \quad 1]^n,$ $n = 2, 4, 8$	$f(\mathbf{X}) = -\exp\left(-0.5 \cdot \sum_{i=1}^{n} {x_i}^2\right)$	−1.0
Goldstein and Price	$\mathbf{X} \in [-2, \quad 2]^2$	$f(\mathbf{X}) = \left[1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3{x_2}^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 - 12x_1 + 48x_2 - 36x_1x_2 + 27{x_2}^2)\right]$	3.0
Griewank	$\mathbf{X} \in [-100, \quad 100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} {x_i}^2 - \prod_{i=1}^{2} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Hartman 3	$\mathbf{X} \in [-30, \quad 30]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right), \quad a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ $p = \begin{bmatrix} 0.3689 & .117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$	−3.862782

(continued)

Table 4.1 (continued)

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

Table 4.2 Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS	MCSS
AP	1360 (0.99)	1360	1277	1253	804	437
Bf1	3992	3356	1640	1615	1187	542
Bf2	20,234	3373	1676	1636	742	556
BL	19,596	2412	2439	1436	423	481
Branin	1442	1418	1404	1257	852	351
Camel	1358	1358	1336	1300	575	384
Cb3	9771	2045	1163	1118	436	288
CM	2105	2105	1743	1539	1563	538
Dejoung	9900	3040	1462	1281	630	387
Exp2	938	936	817	807	132	183
Exp4	3237	3237	2054	1496	867	317
Exp8	3237	3237	2054	1496	1426	659
Goldstein and Price	1478	1478	1408	1325	682	450
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1551	1272
Hartman 3	1350	1350	1332	1274	860	344
Hartman 6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1783	908
Rastrigin	1533 (0.97)	1523 (0.97)	1392	1381	1402	1252
Rosenbrock	9380	3739	1675	1462	1452	1424
Total	112,311 (96.7)	41,640 (96.7)	29,166 (98.16)	25,193 (98.16)	17,367	10,773

comparison is performed to show the difference between the CSS and MCSS algorithm in unsuitable situations, i.e., small number of CPs and maximum number of permitted iterations. Therefore, the number of CPs is set to 10, and the maximum number of permitted iterations is considered as 100. This means that the computational cost is one quarter of the previous comparison. The results of this comparison are presented in Table 4.3. The numbers in Table 4.3 are the optimum found by each algorithm. These are the average of 100 independent runs. The accuracy of the solutions in some cases may be unsatisfactory, but it should be noted that the number of CPs and maximum number of iterations are small. The purpose of this comparison is to magnify the difference between the CSS and MCSS algorithm and verify the better performance of the MCSS in this situation. For more detailed presentation, Fig. 4.4 illustrates the optimization process and convergence.

Statistical Test

Now in the following we want to ensure that the results of MCSS in Table 4.3 are better than CSS algorithm. For this purpose, we apply a multiproblem analysis using statistical tests. We apply the test on the obtained errors by each algorithm. If we have the normality condition for our sample of results, a parametric pair *t*-test can be suitable. We first analyze a safe usage of parametric tests. We utilized two

Table 4.3 Numerical comparison of CSS and MCSS algorithms. Number of CPs = 10, maximum number of iterations = 100

Function	Global minimum	CSS	MCSS	CSS's error	MCSS's error
AP	−0.352386	−0.198721	−0.308396	0.153665	0.04399
Bf1	0.0	28.809183	0.088327	28.80918	0.088327
Bf2	0.0	8.938997	0.034876	8.938997	0.034876
BL	0.0	0.106252	6.781E−05	0.106252	6.78E−05
Branin	0.397887	3.960884	0.537231	3.562997	0.139344
Camel	−1.0316	−0.866765	−1.031591	0.164835	9E−06
Cb3	0.0	0.125161	6.517E−05	0.125161	6.52E−05
CM	−0.4	−0.230142	−0.352661	0.169858	0.047339
Dejoung	0.0	0.166451	6.891E−05	0.166451	6.89E−05
Exp2	−1.0	−0.999366	−0.999947	0.000634	5.3E−05
Exp4	−1.0	−0.990884	−0.999818	0.009116	0.000182
Exp8	−1.0	−0.949659	−0.999686	0.050341	0.000314
Goldstein and Price	3.0	15.729613	4.620501	12.72961	1.620501
Griewank	0.0	0.342795	0.105112	0.342795	0.105112
Hartman 3	−3.862782	−3.491627	−3.816318	0.371155	0.046464
Hartman 6	−3.322368	−2.054548	−3.292364	1.26782	0.030004
Rastrigin	−2.0	−1.875735	−1.917121	0.124265	0.082879
Rosenbrock	0.0	19.476846	3.117751	19.47685	3.117751

normality tests including Kolmogorov–Smirnov and Shapiro–Wilk test. The p-values of the normality tests over the sample results obtained by CSS and MCSS are shown in Table 4.4. If we consider a significance level $\alpha = 0.05$, all of the p-values in Table 4.4 will be <0.05. Thus the sample results do not follow a normal distribution. The Q–Q plot for sample results is illustrated in Fig. 4.5, and it can be understood that the normality conditions are not satisfied in both CSS and MCSS algorithms. This result was predictable because the sample size (the number of problems) is small. Therefore, a parametric test such as pair t-test is not appropriate in this case. Therefore we use the Wilcoxon test that is a nonparametric test for pairwise comparisons. The method of this test is described in Ref. [7]. The result of this test can be summarized as:

- The p-value obtained by the Wilcoxon test is 0.00. Consequently, the Wilcoxon test considers a difference between the performance of these two algorithms assuming a significance level $\alpha = 0.05$. Therefore, because of better mean value of the MCSS algorithm results, MCSS outperforms its predecessor, CSS algorithm.

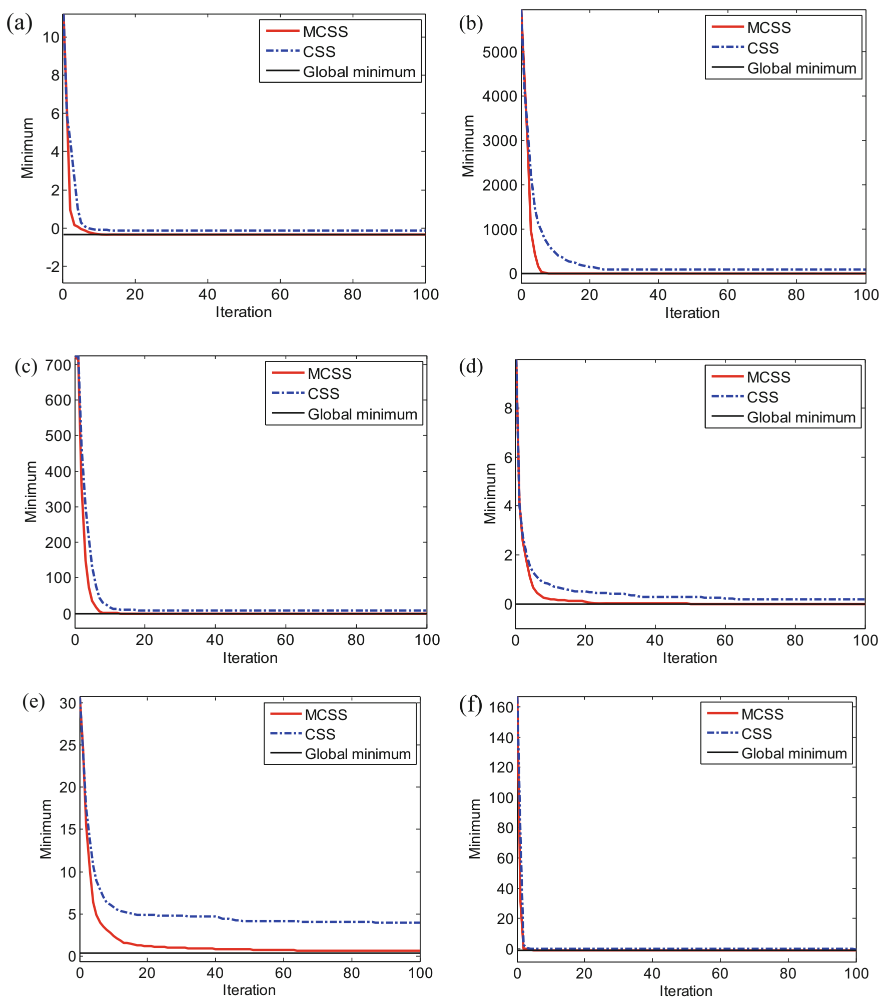


Fig. 4.4 Comparison of the convergence rate of optimizing mathematical benchmarks [1]: (**a**) AP, (**b**) Bf1, (**c**) Bf2, (**d**) BL, (**e**) Branin, (**f**) Camel, (**g**) Cb3, (**h**) CM, (**i**) Dejoung, (**j**) Exp2, (**k**) Exp4, (**l**) Exp8, (**m**) Goldstein and price, (**n**) Griewank, (**o**) Hartman3, (**p**) Hartman6, (**q**) Rastrigin, (**r**) Rosenbrock

4.2.4.2 Comparison Between MCSS and Other State-of-the-Art Algorithms

Description of Test Functions and Algorithms

In the following section, the set of test functions designed for Special Session on Real-Parameter Optimization organized in the 2005 I.E. Congress on Evolutionary Computation (CEC 2005) are solved by the MCSS algorithm. The detailed

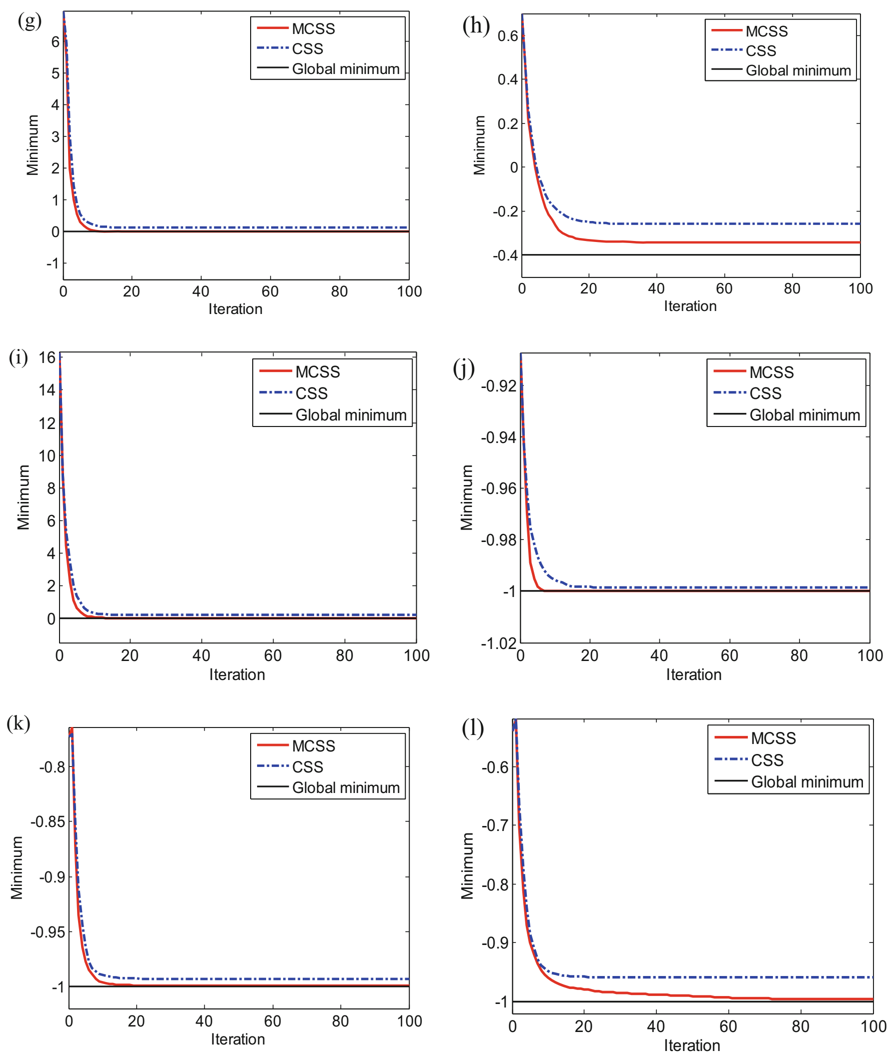


Fig. 4.4 (continued)

Table 4.4 Normality tests and their *p*-values over multiproblem analysis

Algorithm	Kolmogorov–Smirnov	Shapiro–Wilk
CSS	0.00	0.00
MCSS	0.00	0.00

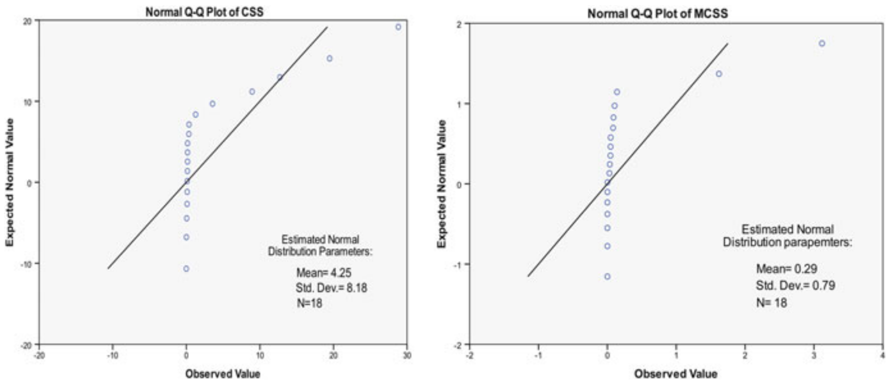


Fig. 4.5 Normal Q–Q plots of the sample results of the CSS and MCSS algorithms [1]

description of test functions is presented by Suganthan et al. [8]. The set of these test functions consists of the following functions:

– Five displaced unimodal functions (f1–f5).
  - Sphere function d.
  - Schewefel's problem 1.2 displaced.
  - Elliptical function rotated widely conditioned.
  - Schwefel's problem 1.2 displaced with noise in the fitness.
  - Schwefel's problem 2.6 with global optimum in the frontier.

– 20 multimodal functions (f6–f7)
  - Seven basic functions.
    – Rosenbrock function displaced.
    – Griewank function displaced and rotated without frontiers.
    – Ackley function displaced and rotated with the global optimum in the frontier.
    – Rastrigin function displaced.
    – Rastrigin function displaced and rotated.
    – Weierstrass function displaced and rotated.
    – Schewefel's problem 2.13.
  - Two expanded functions.
  - 11 hybrid functions. Each one of these has been defined through compositions of 10 out of 14 previous functions (different in each case).

The characteristics of this experiment are the same as what has been suggested by Suganthan et al. [8]. Each function is solved by MCSS in 25 independent runs, and the average error of the best CP is recorded. The number of CPs is set to 25. The dimension of the test functions is set to 10 ($D = 10$), and algorithm performs 10,000 function evaluation. The termination criterion is either reaching the maximum number of function evaluation or achieving error $<10^{-8}$. Table 4.5 shows the

Table 4.5 Average error rates of the algorithms in CEC 2005 and MCSS algorithm

Function	BLX-GL50	BLX-MA	CoEVO	DE	DMS-L-PSO	EDA	G-CMA-ES	K-PCX	L-CMA-ES	L-SaDE	SPC-PNX	MCSS
f1	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09
f2	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09	1E−09
f3	570.5	47710	1E−09	1.94E−06	1E−09	21.21	1E−09	0.415	1E−09	0.01672	1.08E5	0.101
f4	1E−09	2E−08	1E−09	1E−09	0.001885	1E−09	1E−09	7.94E−07	1.76E6	1.42E−05	1E−09	1E−09
f5	1E−09	0.02124	2.133	1E−09	1.14E−06	1E−09	1E−09	48.5	1E−09	0.012	1E−09	1E−09
f6	1E−09	1.49	12.46	0.159	6.89E−08	0.04182	1E−09	0.478	1E−09	1.2E−08	18.91	0.014
f7	0.01172	0.1971	0.03705	0.146	0.04519	0.4205	1E−09	0.231	1E−09	0.02	0.08261	1. E−09
f8	20.35	20.19	20.27	20.4	20	20.34	20	20	20	20	20.99	20
f9	1.154	0.4379	19.19	0.955	1E−09	5.418	0.239	0.119	44.9	1E−09	4.02	0.012
f10	4.975	5.643	26.77	12.5	3.622	5.289	0.0796	0.239	40.8	4.969	7.304	2.152
f11	2.334	4.557	9.029	0.847	4.623	3.944	0.934	6.65	3.65	4.891	1.91	3.823
f12	406.9	74.3	604.6	31.7	2.4001	442.3	29.3	149	209	4.5E−07	259.5	2.503
f13	0.7498	0.7736	1.137	0.977	0.3689	1.841	0.696	0.653	0.494	0.22	0.8379	0.552
f14	2.172	2.03	3.706	3.45	2.36	2.63	3.01	2.35	4.01	2.915	3.046	2.432
f15	400	269.6	293.8	259	4.854	365	228	510	211	32	253.8	153.46
f16	93.49	101.6	177.2	113	94.76	143.9	91.3	95.9	105	101.2	109.6	90.567
f17	109	127	211.8	115	110.1	156.8	123	97.3	549	114.1	119	102.12
f18	420	803.3	901.4	400	760.7	483.2	332	752	497	719.4	439.6	741.73
f19	449	762.8	844.5	420	714.3	564.4	326	751	516	704.9	380	317.27
f20	446	800	862.9	460	822	651.9	300	813	442	713	440	502.31
f21	689.3	721.8	634.9	492	536	484	500	1050	404	464	680.1	436.61
f22	758.6	670.9	778.9	718	692.4	770.9	729	659	740	734.9	749.3	642.49
f23	638.9	926.7	834.6	572	730.3	640.5	559	1060	791	664.1	575.9	630.38
f24	200	224	313.8	200	224	200	200	406	865	200	200	194.17
f25	403.6	395.7	257.3	923	365.7	373	374	406	442	375.9	406	356.51
Mean	224.6815	2144.922	272.4173	189.7254	203.5414	213.4814	152.6623	273.1534	70635.3	194.261	191.12	167.97

Table 4.6 The Wilcoxon test results

MCSS versus	R^+	R^-	*p*-value
BLX-GL50	46	185	0.016
BLX-MA	6	270	0.000
CoEVO	14	239	0.000
DE	59	172	0.050
DMS-L-PSO	57	196	0.024
EDA	20	211	0.314
G-CMA-ES	70	120	.001
K-PCX	20	233	0.025
L-CMA-ES	45	165	0.048
L-SaDE	65.5	187	0.027
SPC-PNX	52	179	0.001

official results of the participated algorithms obtained from Garcia et al. [9]. The description of each algorithm is given in Ref. [19]. The results of the MCSS algorithm are added to Table 4.5. The values of Table 4.5 indicate the average error rate of each algorithm. This value can be considered as a means for measuring the performance of each algorithm.

Numerical Results and Statistical Test

As the results in Table 4.5 show, MCSS has a good performance and its average error rates are good; however, there are some cases that MCSS performs slightly weaker than some other algorithms. For a fair comparison, we have to use statistical test to judge about the performance of MCSS in comparison with other algorithms. We want to find out whether the results of MCSS have a significant difference in comparison with the other algorithms. This is a multiproblem analysis; therefore, a nonparametric test is more suitable in this case. We utilized Wilcoxon's test. This test performs pairwise comparisons between two algorithms. In this test, MCSS is compared to some other algorithms.

Table 4.6 summarizes the results of applying the Wilcoxon test. Table 4.6 includes sum of ranking and *p*-value of each comparison. The method of this test is simply described in Ref. [7]. The significance level α is considered as 0.05. In each comparison when the corresponding *p*-value is <0.05, it means that two compared algorithms behave differently, and the one with smaller mean value of error rate has a better performance.

The *p*-value in pairwise comparison is independent from another one. If we draw a conclusion involving more than one pairwise comparison in Wilcoxon's analysis, an accumulated error which is merged up by combination of pairwise comparisons will be obtained. In statistical terms, the family-wise error rate (FWER) will be lost. FWER is defined as the probability of making one or more false discoveries among all the hypotheses when performing multiple pairwise tests (Garcia et al. [9]). The true statistical significance for combining pairwise comparisons is given by:

$$p = 1 - \prod_{i=1}^{i=k-1} (1 - pH_i) \tag{4.26}$$

where k is the number of pairwise comparisons considered and pH_i is the p-value of each comparison. For more information, the reader may refer to Ref. [9].

Considering the values of Table 4.6, the p-value of all of the comparisons except MCSS versus G-CMA-ES is less than significance level $\alpha = 0.05$, and it cannot be concluded that MCSS is better than all of algorithms except G-CMA-ES because we have to consider FWER in making a conclusion in multiple pairwise comparisons. The MCSS outperforms all of the algorithms except G-CMA-ES considering independent pairwise comparisons due to the fact that the achieved p-values are less than $\alpha = 0.05$. The true p-value for multiple pairwise comparisons can be computed using Eq. (4.22):

$$\begin{gathered} p = 1 - \big((1-0.16)\cdot(1-0.0)\cdot(1-0.0)\cdot(1-0.05)\cdot(1-0.24)\cdot(1-0.001) \\ \cdot(1-0.025)\cdot(1-0.048)\cdot(1-0.027)\cdot(1-0.001)\big) = 0.17765 \end{gathered} \tag{4.27}$$

Based on this analysis it can be claimed that the MCSS algorithm has a better performance in relation with all of the algorithms except G-CMA-ES with a p-value of 0.17765. As a result, if we consider a significance level $\alpha = 0.17765$, the confidence interval for the mentioned claim will be $100(1-\alpha) = 82.23\,\%$.

4.2.5 Engineering Examples

Three well-studied engineering design problems that have been solved by various optimization methods in the literature are used to examine the efficiency of the MCSS algorithm and compare the results with those obtained by the CSS. For handling constraints, a simple penalty function is utilized to prevent adding the effect of a robust constraint handling method on the performance of the algorithm.

Example 1 A tension/compression spring design problem

This is a well-known optimization problem which has been used to evaluate the efficiency of different optimization methods [6]. This problem is defined by Belegundu [10] and Arora [11] as depicted in Fig. 4.6. The objective of this optimization problem is to minimize the weight of tension/compression spring. This minimization involves some constrains, i.e., shear stress, frequency, and minimum deflection.

The design variables are the mean coil diameter $D(=x_1)$, the wire diameter $d(=x_2)$, and the number active coils $N(=x_3)$. By considering these decision variables, the cost function can be formulated as:

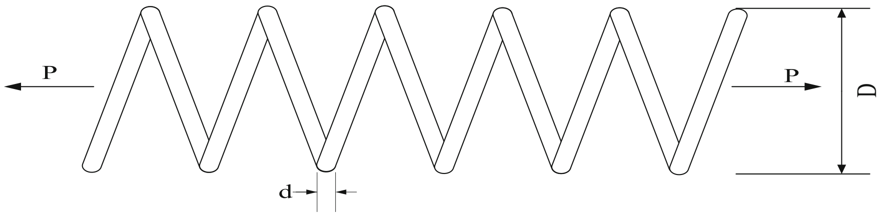


Fig. 4.6 Schematic of the tension/compression spring with indication of design variables

Table 4.7 Optimum results for the tension/compression spring design

	Optimal design variables			
Methods	$x_1(d)$	$x_2(D)$	$x_3(N)$	f_{cost}
Belegundu [10]	0.050000	0.315900	14.250000	0.0128334
Arora [11]	0.053396	0.399180	9.1854000	0.0127303
Coello [12]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [13]	0.051989	0.363965	10.890522	0.0126810
He and Wang [14]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [15]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [16]	0.051865	0.361500	11.000000	0.0126432
Kaveh and Talatahari (CSS) [6]	0.051744	0.358532	11.165704	0.0126384
Present work [1]	0.051645	0.356496	11.271529	0.0126192

$$f_{\cos t}(\mathbf{X}) = (x_3 + 2)x_2x_1^2 \tag{4.28}$$

$$\begin{aligned} g_1(\mathbf{X}) &= 1 - \frac{x_2^3x_3}{71785 \cdot x_1^4} \le 0, \\ g_2(\mathbf{X}) &= \frac{4x_2^2 - x_1x_2}{12566 \cdot (x_2x_1^3 - x_1^4)} + \frac{1}{5108 \cdot x_1^2} - 1 \le 0, \\ g_3(\mathbf{X}) &= 1 - \frac{140.45x_1}{x_2^2x_3} \le 0, \\ g_4(\mathbf{X}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0. \end{aligned} \tag{4.29}$$

The decision variables are limited as:

$$\begin{aligned} 0.05 &\le x_1 \le 2.0, \\ 0.25 &\le x_2 \le 1.3, \\ 2.0 &\le x_3 \le 15. \end{aligned} \tag{4.30}$$

This problem has been solved with various methods by different researchers, Belegundu [10], Arora [11], Coello [12], Coello and Montes [13], He and Wang [14], Montes and Coello [15], and Kaveh and Talatahari [14, 26]. The results of the best solutions found by different methods are presented in Table 4.7. From Table 4.7 it can be understood that the best solution found by MCSS is better than other

Table 4.8 Statistical results of different methods for the tension/compression spring

Methods	Best	Mean	Worst	Standard deviation
Belegundu [10]	0.0128334	N/A	N/A	N/A
Arora [11]	0.0127303	N/A	N/A	N/A
Coello [12]	0.0127048	0.012769	0.012822	3.9390e−5
Coello and Montes [13]	0.0126810	0.012742	0.012973	5.9000e−5
He and Wang [14]	0.0126747	0.012730	0.012924	5.1985e−5
Montes and Coello [15]	0.012698	0.013461	0.16485	9.6600e−4
Kaveh and Talatahari [16]	0.0126432	0.012720	0.012884	3.4888e−5
Kaveh and Talatahari (CSS) [6]	0.0126384	0.012852	0.013626	8.3564e−5
Present work [1]	0.0126192	0.012794	0.013962	5.3491e−5

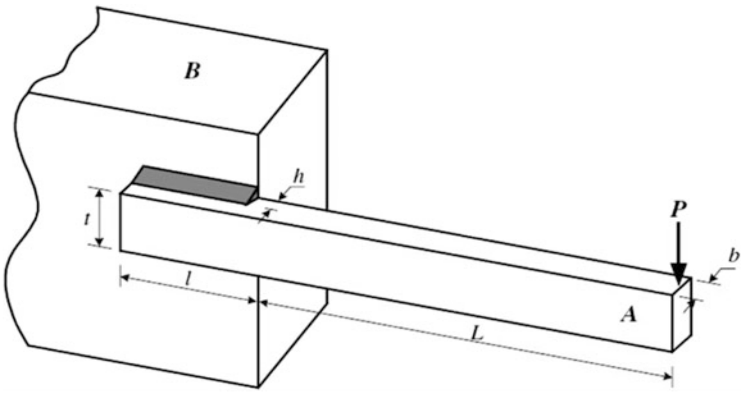


Fig. 4.7 Schematic of the welded beam system

methods. The statistical simulation results of 30 independent runs for MCSS are illustrated in Table 4.8 and compared to other methods.

Example 2 A welded beam design

One of the practical design problems which has been widely used as a benchmark to test the performance of different optimization methods is the welded beam design problem as illustrated in Fig. 4.7. The goal of this optimization problem is to minimize the constructing cost of a welded beam that is subjected to different constrains, such as shear (τ) and bending (σ) stresses, buckling load (P_c), end deflection (δ), and end side constraint. Design variables are $h(=x_1)$, $l(=x_2)$, t $(=x_3)$, and $b(=x_4)$. By considering the setup, welding labor, and the materials costs, the cost function can be expressed as:

$$f_{\cos t}(\mathbf{X}) = 1.1047 x_1{}^2 x_2 + 0.04811 x_3 x_4 \cdot (14.0 + x_2) \tag{4.31}$$

Subjected to the following constrains:

$$\begin{gathered}
g_1(\mathbf{X}) = \tau(\{x\}) - \tau_{\max} \leq 0, \\
g_2(\mathbf{X}) = \sigma(\{x\}) - \delta_{\max} \leq 0, \\
g_3(\mathbf{X}) = x_1 - x_4 \leq 0, \\
g_4(\mathbf{X}) = 0.10471{x_1}^2 + 0.04811 x_3 x_4 \cdot (14.0 + x_2) - 5.0 \leq 0, \\
g_5(\mathbf{X}) = 0.125 - x_1 \leq 0, \\
g_6(\mathbf{X}) = \delta(\{x\}) - \delta_{\max} \leq 0, \\
g_7(\mathbf{X}) = P - P_c(\{x\}) \leq 0.
\end{gathered} \tag{4.32}$$

where

$$\begin{gathered}
\tau(\mathbf{X}) = \sqrt{(\tau')^2 + 2\tau' \cdot \tau'' \frac{x_2}{2R} + \left(\tau''\right)^2}, \\
\tau' = \frac{P}{\sqrt{2}x_1 \cdot x_2}, \tau'' = \frac{MR}{J}, \\
M = P \cdot \left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_2}{2}\right)^2}, \\
J = 2\left\{\sqrt{2}x_1 x_2 \left[\frac{{x_2}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}, \\
\sigma(\mathbf{X}) = \frac{6PL}{x_4 \cdot {x_3}^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^2 x_4}, \\
P_c(\mathbf{X}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \\
P = 6{,}000\,\text{lb}, \quad L = 14\,\text{in}, \\
E = 30 \times 10^6\,\text{psi}, \quad G = 12 \times 10^6\,\text{psi}
\end{gathered} \tag{4.33}$$

and variable boundaries are:

$$\begin{gathered}
0.1 \leq x_1 \leq 2.0, \\
0.1 \leq x_2 \leq 10, \\
0.1 \leq x_3 \leq 10, \\
0.1 \leq x_3 \leq 2.0.
\end{gathered} \tag{4.34}$$

This is a well-studied problem that is solved by different researchers using different approaches. Regsdell and Phillips [17] solved it using mathematical-based methods. Deb [18], Coello [12], and Coello and Montes [13] solved it using GA-based algorithms. Also, He and Wang [14] solved it by CPSO, Montes and Coello [15] by evolutionary strategies, and Kaveh and Talatahari [16] by ACO. This problem is also solved by Kaveh and Talatahari [6] utilizing the CSS algorithm. The results of the best solution found by each method are listed in Table 4.9. The best solution found by MCSS is better than other results in literature. The result of the MCSS is slightly better than that of the CSS, but the speed of the convergence is much higher compared to the CSS. The results of statistical simulation are

Table 4.9 Optimum results for the design of welded beam

Methods	Optimal design variables				
	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	f_{cost}
Regsdell and Phillips [17]					
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815
DAVID	0.2434	6.2552	8.2915	0.2444	2.3841
SIMPLEX	0.2792	5.6256	7.7512	0.2796	2.5307
RANDOM	0.4575	4.7313	5.0853	0.6600	4.1185
Deb [18]	0.248900	6.173000	8.178900	0.253300	2.433116
Coello [12]	0.248900	3.420500	8.997500	0.210000	1.748309
Coello and Montes [13]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang [14]	0.202369	3.544214	9.048210	0.205723	1.728024
Montes and Coello [15]	0.199742	3.612060	9.037500	0.206082	1.737300
Kaveh and Talatahari [16]	0.205700	3.471131	9.036683	0.205731	1.724918
Kaveh and Talatahari (CSS) [6]	0.205820	3.468109	9.038024	0.205723	1.724866
Present work [1]	0.205729	3.470493	9.036623	0.205729	1.724853

Table 4.10 Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Regsdell and Phillips [17]	2.3815	N/A	N/A	N/A
Deb [18]	2.433116	N/A	N/A	N/A
Coello [12]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [13]	1.728226	1.792654	1.993408	0.074713
He and Wang [14]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [15]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [16]	1.724918	1.729752	1.775961	0.009200
Kaveh and Talatahari (CSS) [6]	1.724866	1.739654	1.759479	0.008064
Present work [1]	1.724853	1.735438	1.753681	0.009527

presented in Table 4.10. Similar to the CSS algorithm, MCSS has a small value for the standard deviation.

Example 3 A pressure vessel design problem

The objective of this optimization is to minimize the cost of fabricating a pressure vessel which is clapped at both ends by hemispherical heads as depicted in Fig. 4.8. The construction cost consists of the cost of materials, forming, and welding [19]. The design variables are the thickness of the shell T_s $(=x_1)$, the thickness of the head T_h $(=x_2)$, the inner radius R $(=x_3)$, and the length of cylindrical section of the vessel L $(=x_4)$. T_s and T_h are integer multiples of 0.0625 in the available thickness of the rolled steel plates, but R and L are continuous variables. The mathematical expression of the cost function is:

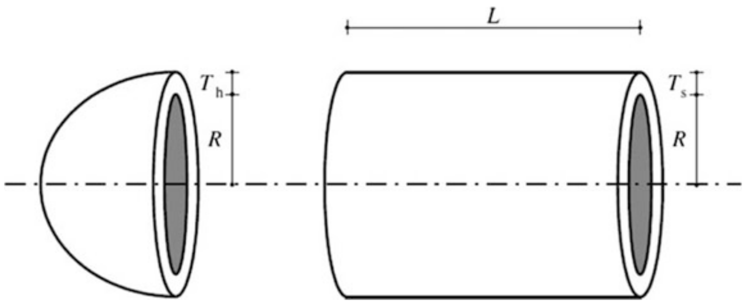


Fig. 4.8 Schematic of the pressure vessel and its design variables

$$f_{\cos t}(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2 + 19.84x_1^2x_3, \tag{4.35}$$

The constraint areas are as follows:

$$\begin{gathered} g_1(\mathbf{X}) = -x_1 + 0.0193x_3 \le 0, \\ g_2(\mathbf{X}) = -x_2 + 0.00954x_3 \le 0, \\ g_3(\mathbf{X}) = -\pi \cdot x_3^2x_4 - \frac{4}{3}\pi \cdot x_3^3 + 1,296,000 \le 0, \\ g_4(\mathbf{X}) = x_4 - 240 \le 0. \end{gathered} \tag{4.36}$$

The search space is defined as:

$$\begin{gathered} 0 \le x_1 \le 99, \\ 0 \le x_2 \le 99, \\ 10 \le x_3 \le 200, \\ 10 \le x_3 \le 200. \end{gathered} \tag{4.37}$$

Various types of methods have been used to solve this problem. Some of these approaches are a branch and bound method [19], an augmented Lagrangian multiplier approach [20], a genetic adaptive search [21], a GA-based algorithm [12], a feasibility-based tournament selection scheme [13], a coevolutionary particle swarm method [14], an evolution strategy [15], an improved ant colony optimization [16], and the CSS algorithm [6]. The results of the best solution found by different methods are presented in Table 4.11. MCSS algorithm found better solution compared to other techniques and the standard CSS. In Table 4.12 the results of statistical simulations are listed. The mean value of the 30 independent runs for MCSS is slightly weaker than that of the CSS; however, the best solution and speed of the convergence for MCSS is much higher.

Table 4.11 Optimum results for the design of welded beam

Methods	Optimal design variables				
	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	f_{cost}
Sandgren [19]	1.125000	0.625000	47.700000	117.701000	8129.1036
Kannan and Kramer [20]	1.125000	0.625000	58.291000	43.690000	7198.0428
Deb and Gene [21]	0.937500	0.500000	48.329000	112.679000	6410.3811
Coello [12]	0.812500	0.437500	40.323900	200.000000	6288.7445
Coello and Montes [13]	0.812500	0.437500	42.097398	176.654050	6059.9463
He and Wang [14]	0.812500	0.437500	42.091266	176.746500	6061.0777
Montes and Coello [15]	0.812500	0.437500	42.098087	176.640518	6059.7456
Kaveh and Talatahari [16]	0.812500	0.437500	42.098353	176.637751	6059.7258
Kaveh and Talatahari (CSS) [6]	0.812500	0.437500	42.103624	176.572656	6059.0888
Present work [1]	0.812500	0.437500	42.107406	176.525589	6058.6233

Table 4.12 Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Sandgren [19]	8129.1036	N/A	N/A	N/A
Kannan and Kramer [20]	7198.0428	N/A	N/A	N/A
Deb and Gene [21]	6410.3811	N/A	N/A	N/A
Coello [12]	6288.7445	6293.8432	6308.1497	7.4133
Coello and Montes [13]	6059.9463	6177.2533	6469.3220	130.9297
He and Wang [14]	6061.0777	6147.1332	6363.8041	86.4545
Montes and Coello [15]	6059.7456	6850.0049	7332.8798	426.0000
Kaveh and Talatahari [16]	6059.7258	6081.7812	6150.1289	67.2418
Kaveh and Talatahari (CSS) [6]	6059.0888	6067.9062	6085.4765	10.2564
Present work [1]	6058.6233	6073.5931	6108.5479	24.6712

4.3 Improved Magnetic Charged System Search

In this part, the improved version of magnetic charged system search (IMCSS) is presented and also utilized for optimization of truss structures. As mentioned earlier, the standard CSS and MCSS algorithms use harmony search-based approach for process of position correction of CPs. In this process, the CMCR and PAR parameters help the algorithm to find globally and locally improved solutions, respectively [22]. PAR and bw in HS scheme are very important parameters in fine-tuning of optimized solution vectors and can be potentially useful in adjusting convergence rate of algorithm to optimal solution.

The traditional HS scheme uses fixed value for both PAR and bw. Small PAR values with large bw values can lead to poor performance of the algorithm and increase the iterations needed to find optimum solution; also on the other hand small bw values in final iterations increase the fine-tuning of solution vectors, but in the

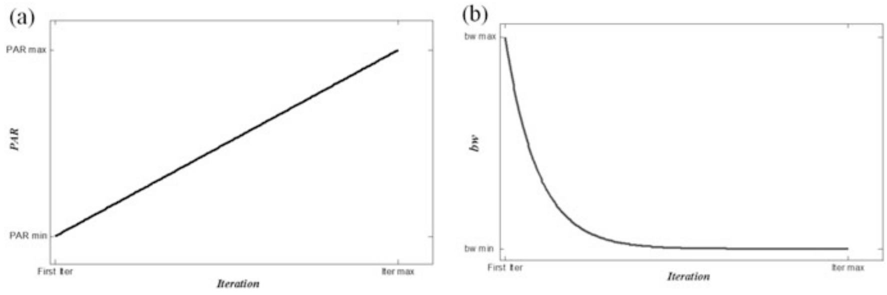


Fig. 4.9 Variation of (**a**) PAR and (**b**) bw versus iteration number [2]

first iterations bw must take a bigger value to enforce the algorithm to increase the diversity of solution vectors. Furthermore, large PAR values with small bw values usually lead to the improvement of best solutions in final iterations and convergence of the algorithm to optimal solution vector. To improve the performance of the HS scheme and eliminate the drawbacks which correspond with fixed values of PAR and bw; IMCSS algorithm uses an improved form of HS algorithm with varied PAR and bw for the step of position correction. PAR and bw change dynamically with iteration number as shown in Fig. 4.9 and expressed as follows: [22]:

$$PAR(iter) = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{iter_{\max}} \cdot iter \tag{4.38}$$

and

$$bw(iter) = bw_{\max}\exp(c \cdot iter), \tag{4.39}$$

$$c = \frac{Ln\left({}^{bw_{\min}}\!/_{bw_{\max}}\right)}{iter_{\max}}, \tag{4.40}$$

where $PAR(iter)$ and $bw(iter)$ are the values of the PAR and bandwidth for each iteration, respectively, subscripts *min* and *max* denote the minimum and maximum values for each parameter, respectively, and *iter* is the current iteration number.

4.3.1 A Discrete IMCSS

The IMCSS algorithm can be also applied to optimal design problem with discrete variables. One way to solve discrete problems using a continuous algorithm is to utilize a rounding function which changes the magnitude of a result to the nearest discrete value [23], as follows:

$$X_{j,new} = Fix\left(rand_{j1} \cdot k_a \cdot \frac{F_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot V_{j,old} \cdot \Delta t + X_{j,old}\right), \tag{4.41}$$

where *Fix(X)* is a function which rounds each element of vector X to the nearest allowable discrete value. Using this position updating formula, the agents will be permitted to select discrete values.

4.3.2 An Improved Magnetic Charged System Search for Optimization of Truss Structures with Continuous and Discrete Variables

4.3.2.1 Statement of the Optimization Problem

The aim of size optimization of truss structures is to find the optimum values for cross-sectional area of members A_i, in order to minimize the structural weight W, satisfying the constraints corresponding to the response of the structure. Thus, the optimal design problem can be expressed as:

$$\begin{array}{lll} Find & X = [x_1, x_2, x_3, \ldots, x_n] & \\ to\ minimize & Mer(X) = f_{penalty}(X) \times W(X) & \\ subject\ to & \sigma_{\min} < \sigma_i < \sigma_{\max} & i = 1, 2, \ldots, nm \\ & \delta_{\min} < \delta_i < \delta_{\max} & i = 1, 2, \ldots, nn \end{array} \tag{4.42}$$

where X is the vector containing the design variables; for a discrete optimum design problem, the variables x_i are selected from an allowable set of discrete values; n is the number of member groups; $Mer(X)$ is the merit function; $W(X)$ is the cost function, which is taken as the weight of the structure; $f_{penalty}(X)$ is the penalty function which results from the violations of the constraints; nm is the number of members forming the structure; nn is the number of nodes; σ_i and δ_{i} are the stress of members and nodal displacements, respectively; and *min* and *max* mean the lower and upper bounds of constraints, respectively. The cost function can be expressed as:

$$W(X) = \sum_{i=1}^{nm} \rho_i \cdot A_i \cdot L_i \tag{4.43}$$

where ρ_i is the material density of the member i, L_i is the length of the ith member, and A_i is the cross-sectional area of the member i.

The penalty function can be defined as:

$$f_{penalty}(X) = \left(1 + \varepsilon_1 \cdot \sum_{i=1}^{np} \left(\varphi_{\sigma(i)}^k + \varphi_{\delta(i)}^k\right)\right)^{\varepsilon_2}, \tag{4.44}$$

where *np* is the number of multiple loadings. Here ε_1 is taken as unity, and ε_2 is set to 1.5 in the first iterations of the search process, but gradually it is increased to 3 [24]. φ_σ^k and φ_δ^k are the summation of stress penalties and nodal displacement penalties for *k*th charged particle which are mathematically expressed as:

$$\varphi_\sigma = \sum_{i=1}^{nm} \max\left(\left|\frac{\sigma_i}{\overline{\sigma}_i}\right| - 1, 0\right), \tag{4.45}$$

$$\varphi_\delta = \sum_{i=1}^{nn} \max\left(\left|\frac{\delta_i}{\overline{\delta}_i}\right| - 1, 0\right), \tag{4.46}$$

where $\sigma_i$and $\overline{\sigma}_i$ are the stress and allowable stress in member *i*, respectively, and δ_i and $\overline{\delta}_i$ are the displacement of the joints and the allowable displacement, respectively.

4.3.2.2 Numerical Examples

In this section, common truss optimization examples as benchmark problems are used for optimization using the proposed algorithm. This algorithm is applied to problems with both continuous and discrete variables. The final results are compared to those of previous studies to demonstrate the efficiency of the present method. The discrete variables are selected from American Institute of Steel Construction (AISC) code [25], listed in Table 4.13.

In the proposed algorithm, for all of examples a population of 25 CPs is used, and the value of CMCR is set to 0.95.

Example 1 A 10-bar planar truss structure

The 10-bar truss structure is a common problem in the field of structural optimization to verify the efficiency of a proposed optimization algorithm. The geometry and support conditions for this planar, cantilevered truss with loading condition, are shown in Fig. 4.10.

There are ten design variables in this example and a set of pseudo-variables ranging from 0.1 to 35.0 in^2 (0.6452–225.806 cm^2).

In this problem two cases are considered:

Case 1, $P_1 = 100$ kips (444.8 kN) and $P_2 = 0$, and Case 2, $P_1 = 150$ kips (667.2 kN) and $P_2 = 50$ kips (222.4 kN).

Table 4.13 The allowable steel pipe sections taken from AISC code

No.	Area (in^2)	Area (mm^2)	No.	Area (in^2)	Area (mm^2)
1	0.111	71.613	33	3.84	2477.414
2	0.141	90.968	34	3.87	2496.769
3	0.196	126.451	35	3.88	2503.221
4	0.25	161.29	36	4.18	2696.769
5	0.307	198.064	37	4.22	2722.575
6	0.391	252.258	38	4.49	2896.768
7	0.442	285.161	39	4.59	2961.284
8	0.563	363.225	40	4.8	3096.768
9	0.602	388.386	41	4.97	3206.445
10	0.766	494.193	42	5.12	3303.219
11	0.785	506.451	43	5.74	3703.218
12	0.994	641.289	44	7.22	4658.055
13	1	645.16	45	7.97	5141.925
14	1.228	792.256	46	8.53	5503.215
15	1.266	816.773	47	9.3	5999.988
16	1.457	939.998	48	10.85	6999.986
17	1.563	1008.385	49	11.5	7419.43
18	1.62	1045.159	50	13.5	8709.66
19	1.8	1161.288	51	13.9	8967.724
20	1.99	1283.868	52	14.2	9161.272
21	2.13	1374.191	53	15.5	9999.98
22	2.38	1535.481	54	16	10,322.56
23	2.62	1690.319	55	16.9	10,903.2
24	2.63	1696.771	56	18.8	12,129.01
25	2.88	1858.061	57	19.9	12,838.68
26	2.93	1890.319	58	22	14,193.52
27	3.09	1993.544	59	22.9	14,774.16
28	1.13	729.031	60	24.5	15,806.42
29	3.38	2180.641	61	26.5	17,096.74
30	3.47	2238.705	62	28	18,064.48
31	3.55	2290.318	63	30	19,354.8
32	3.63	2341.931	64	33.5	21,612.86

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 $\pm$25 ksi (172.375 MPa), and all nodes in both vertical and horizontal directions are subjected to the displacement limits of $\pm$2.0 in (5.08 cm). Figure 4.11 shows a comparison of the convergence history of both cases for MCSS and IMCSS algorithms.

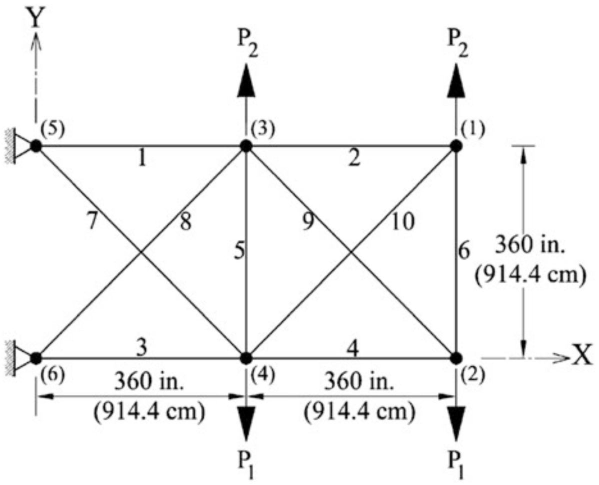


Fig. 4.10 Schematic of a 10-bar planar truss structure

Tables 4.14 and 4.15 are provided for comparison of the optimal design results with those of the previous studies for both cases. In both cases the HS algorithm reaches its best solutions after 20,000 analyses and the PSO and PSOPC algorithms after 3000 iterations (150,000 analyses). The HPSACO algorithm finds the best solution after 10,650 and 9925 analyses, for Case 1 and Case 2, respectively.

The MCSS and IMCSS algorithms achieve the best solutions after 355 iterations (8875 analyses) and 339 iterations (8475 analyses), respectively. The best weights of IMCSS are 5064.6 lb for Case 1 and 4679.15 for Case 2.

As seen in both tables, although the best weights of IMCSS in both cases are a little bigger than the HPSACO, it has lower penalty values compared to HPSACO, and therefore IMCSS has a lower merit function than HPSACO.

Example 2 A 52-bar planar truss

The 52-bar planar truss structure shown in Fig. 4.12 has been analyzed by Lee and Geem [27], Li et al. [28], Wu and Chow [30], and Kaveh and Talatahari [31].

The members of this structure are divided into 12 groups: (1) A1–A4, (2) A5–A10, (3) A11–A13, (4) A14–A17, (5) A18–A23, (6) A24–A26, (7) A27–A30, (8) A31–A36, (9) A37–A39, (10) A40–A43, (11) A44–A49, and (12) A50–A52.

The material density is 7860.0 kg/m^3 and the modulus of elasticity is 2.07×10^5 MPa. The members are subjected to stress limitations of $\pm$180 MPa. Both of the loads, $P_x = 100$ kN and $P_y = 200$ kN, are considered.

Table 4.16 and Fig. 4.13 are provided for comparison of the optimal design results with the previous studies and convergence rates for the 52-bar planar truss structure, respectively.

Table 4.16 shows that the best weights of MCSS and IMCSS algorithms are 1904.05 lb and 1902.61 lb, respectively, while for DHPSACO is 1904.83 lb.

The MCSS and IMCSS algorithms find the best solutions after 4225 and 4075 analyses, respectively, but the DHPSACO reaches a good solution in 5300 analyses. As it can be seen in the results of Table 4.16, the IMCSS algorithm achieves better

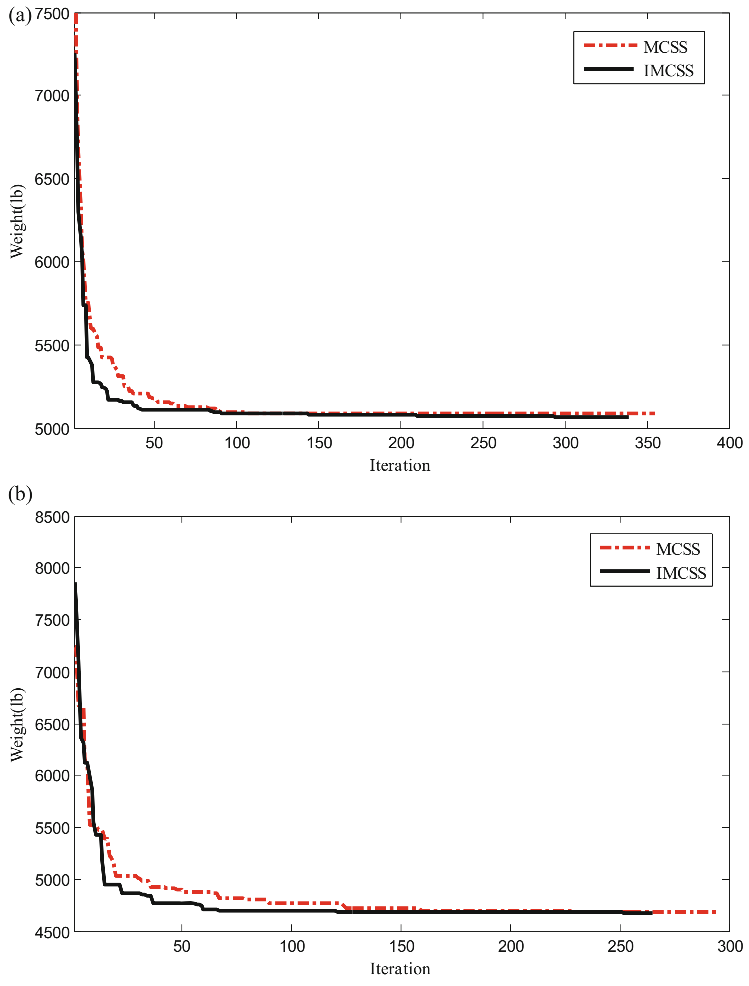


Fig. 4.11 Convergence curves for the 10-bar planar truss structure using MCSS and IMCSS [2]. (**a**) Case 1 and (**b**) Case 2

optimal results than previous methods like MCSS, PSO, PSOPC, HPSO, and DHPSACO algorithms.

Example 3 A 72-bar spatial truss

In the 72-bar spatial truss structure which is shown in Fig. 4.14, 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 nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm), and the members are subjected to the stress limits of ±25 ksi (±172.375 MPa).

Table 4.14 Optimal design comparison for the 10-bar planner truss (Case 1)

Element group		Camp et al. [26]	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [29]	Present work [2]	
		GA	HS	PSO	PSOPC	HPSO	HPSACO	MCSS	IMCSS
1	A1	28.92	30.15	33.469	30.569	30.704	30.307	29.5766	30.0258
2	A2	0.1	0.102	0.11	0.1	0.1	0.1	0.1142	0.1
3	A3	24.07	22.71	23.177	22.974	23.167	23.434	23.806	23.627
4	A4	13.96	15.27	15.475	15.148	15.183	15.505	15.887	15.973
5	A5	0.1	0.102	3.649	0.1	0.1	0.1	0.1137	0.1
6	A6	0.56	0.544	0.116	0.547	0.551	0.5241	0.1003	0.5167
7	A7	7.69	7.541	8.328	7.493	7.46	7.4365	8.6049	7.4567
8	A8	21.95	21.56	23.34	21.159	20.978	21.079	21.682	21.437
9	A9	22.09	21.45	23.014	21.556	21.508	21.229	20.303	20.744
10	A10	0.1	0.1	0.19	0.1	0.1	0.1	0.1117	0.1
Weight (lb)		5076.31	5057.88	5529.5	5061	5060.92	5056.56	5086.9	5064.6
Displacement constraint		–	–	–	–	5.53E−07	9.92E−04	1.49E−05	5.85E−08
No. of analyses		N/A	20,000	150,000	150,000	N/A	10,650	8875	8475

Table 4.15 Optimal design comparison for the 10-bar planner truss (Case 2)

Element group		Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [29]	Present work [2]	
		HS	PSO	PSOPC	HPSO	HPSACO	MCSS	IMCSS
1	A1	23.25	22.935	23.473	23.353	23.194	22.863	23.299
2	A2	0.102	0.113	0.101	0.1	0.1	0.120	0.1
3	A3	25.73	25.355	25.287	25.502	24.585	25.719	25.682
4	A4	14.51	14.373	14.413	14.25	14.221	15.312	14.510
5	A5	0.1	0.1	0.1	0.1	0.1	0.101	0.1
6	A6	1.977	1.99	1.969	1.972	1.969	1.968	1.969
7	A7	12.21	12.346	12.362	12.363	12.489	12.310	12.149
8	A8	12.61	12.923	12.694	12.894	12.925	12.934	12.360
9	A9	20.36	20.678	20.323	20.356	20.952	19.906	20.869
10	A10	0.1	0.1	0.103	0.101	0.101	0.100	0.1
Weight (lb)		4668.81	4679.47	4677.7	4677.29	4675.78	4686.47	4679.15
Displacement constraint		–	–	–	0	7.92E−04	0	0
Stress constraint		–	–	–	2.49E−05	7.97E−05	0	0
No. of analyses		N/A	150,000	150,000	N/A	9625	7350	6625

All members of this spatial truss are categorized into 16 groups using symmetry: (1) A1–A4, (2) A5–A12, (3) A13–A16, (4) A17–A18, (5) A19–A22, (6) A23–A30, (7) A31–A34, (8) A35–A36, (9) A37–A40, (10) A41–A48, (11) A49–A52, (12) A53–A54, (13) A55–A58, (14) A59–A66 (15), A67–A70, and (16) A71–A72.

Two optimization cases are implemented:

Case 1: The discrete variables are selected from the set D = {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2} (in^2) or {0.65, 1.29, 1.94, 2.58, 3.23, 3.87, 4.52, 5.16, 5.81, 6.45, 7.10, 7.74, 8.39, 9.03, 9.68, 10.32, 10.97, 12.26, 12.90, 13.55, 14.19, 14.84, 15.48, 16.13, 16.77, 17.42, 18.06, 18.71, 19.36, 20.00, 20.65} (cm^2).

Case 2: The discrete variables are selected from AISC code in Table 4.13. Table 4.17 lists the values and directions of the two load cases applied to the 72-bar spatial truss.

Tables 4.18 and 4.19 are provided for comparison of the results of MCSS and IMCSS algorithms with the results of the previous studies for both cases. The convergence history for both algorithms is shown in Fig. 4.15.

In Case 1, the best weight of the IMCSS and DHPSACO algorithm is 385.54 lb (174.88 kg), while it is 389.49 lb, 388.94 lb, 387.94 lb, and 400.66 lb for the MCSS, HPSO, HS, and GA, respectively. For the PSO and PSOPC algorithms, these algorithms do not get optimal results when the maximum number of iterations is

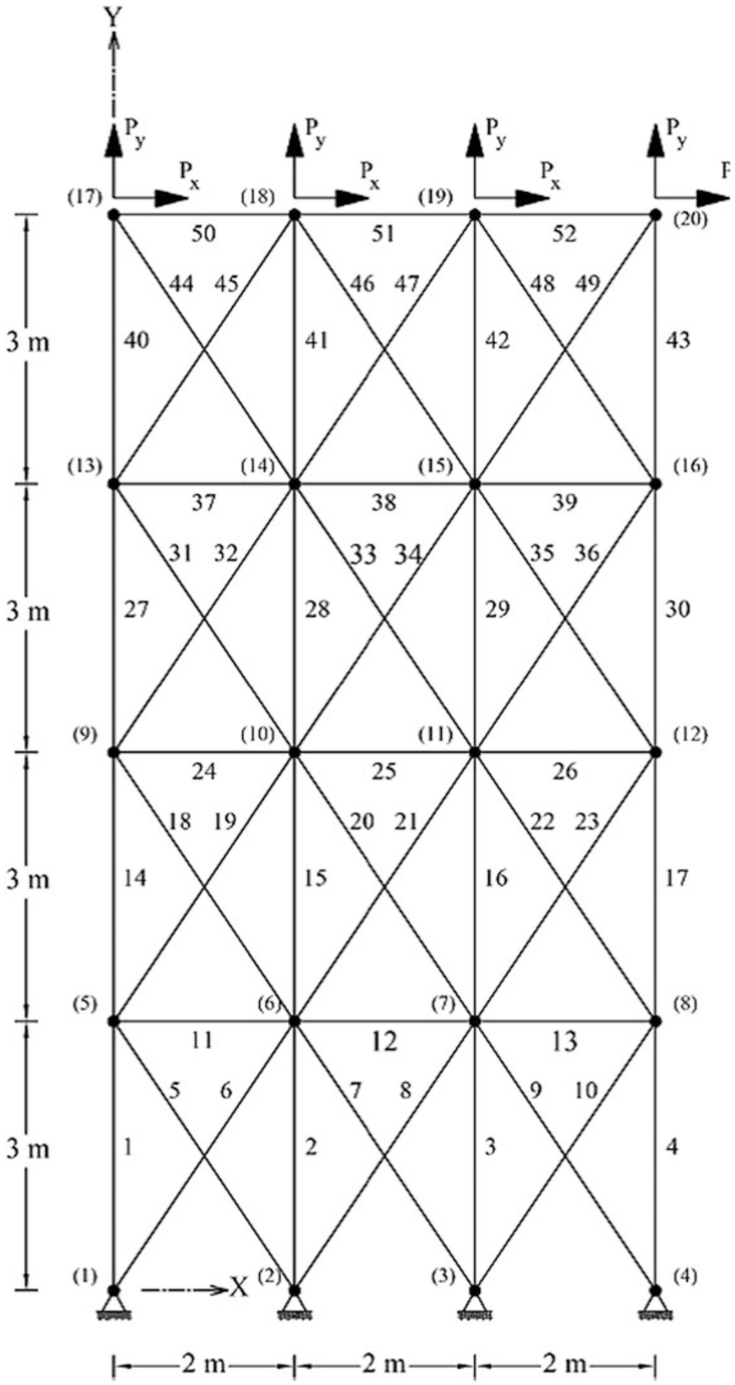


Fig. 4.12 Schematic of a 52-bar planar truss

Table 4.16 Optimal design comparison for the 52-bar planar truss

Element group	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [31]	Present work [2]	
	HS	PSO	PSOPC	HPSO	DHPSACO	MCSS	IMCSS
1	4658.055	4658.055	5999.988	4658.055	4658.055	4658.055	4658.055
2	1161.288	1374.19	1008.38	1161.288	1161.288	1161.288	1161.288
3	506.451	1858.06	2696.38	363.225	494.193	363.225	494.193
4	3303.219	3206.44	3206.44	3303.219	3303.219	3303.219	3303.219
5	940	1283.87	1161.29	940	1008.385	939.998	939.998
6	494.193	252.26	729.03	494.193	285.161	506.451	494.193
7	2290.318	3303.22	2238.71	2238.705	2290.318	2238.705	2238.705
8	1008.385	1045.16	1008.38	1008.385	1008.385	1008.385	1008.385
9	2290.318	126.45	494.19	388.386	388.386	388.386	494.193
10	1535.481	2341.93	1283.87	1283.868	1283.868	1283.868	1283.868
11	1045.159	1008.38	1161.29	1161.288	1161.288	1161.288	1161.288
12	506.451	1045.16	494.19	792.256	506.451	729.031	494.193
Weight (kg)	1906.76	2230.16	2146.63	1905.49	1904.83	1904.05	1902.61
No. of analyses	N/A	N/A	N/A	50,000	5300	4225	4075

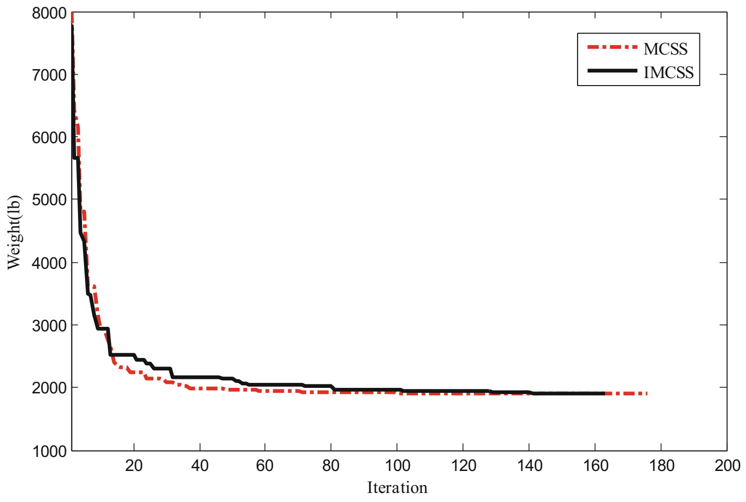


Fig. 4.13 Convergence curves for the 52-bar planar truss structure using MCSS and IMCSS [2]

reached. The IMCSS algorithm gets the best solution after 145 iterations (3625 analyses), while it takes for MCSS and DHPSACO 216 iterations (5400 analyses) and 213 iterations (5330 analyses), respectively.

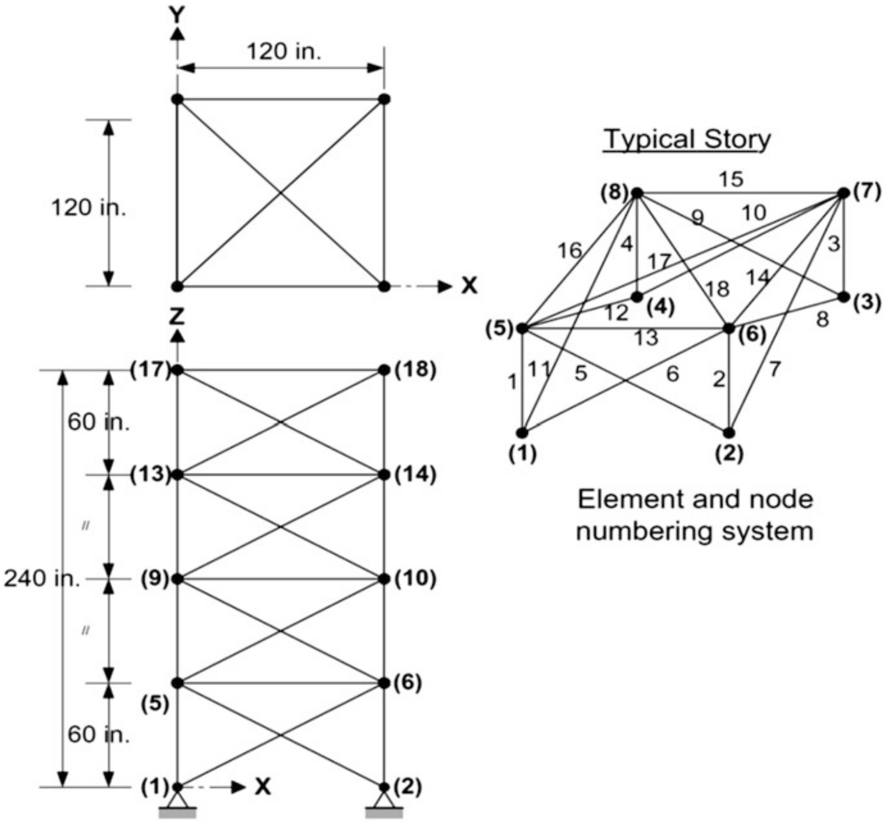


Fig. 4.14 Schematic of a 72-bar spatial truss

Table 4.17 Loading conditions for the 72-bar spatial truss

Node	Case 1			Case 2		
	P_X kips (kN)	P_y kips (kN)	P_z kips (kN)	P_X kips (kN)	P_y kips (kN)	P_z kips (kN)
17	5.0 (22.25)	5.0 (22.25)	−5.0 (−22.25)	0.0	0.0	−5.0 (−22.25)
18	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)
19	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)
20	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)

In Case 2, the best obtained weight from IMCSS is 389.60 lb, but it is 393.13 lb, 389.87 lb, 392.84 lb, 393.06 lb, and 393.38 lb for MCSS, CS, ICA, CSS, and HPSACO algorithms, respectively. IMCSS algorithm finds the best solutions after 173 iterations (4325 analyses), while MCSS, CS, ICA, CSS, and HPSACO algorithms need 4775, 4840, 4500, 7000, and 5330 analyses to find the best solutions.

Table 4.18 Optimal design comparison for the 72-bar truss (Case 1)

Element group		Wu and Chow [30]	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [31]	Present work [2]	
		GA	HS	PSO	PSOPC	HPSO	DHPSACO	MCSS	IMCSS
A1	A1 ~ A4	1.5	1.9	2.6	3	2.1	1.9	1.8	2
A2	A5 ~ A12	0.7	0.5	1.5	1.4	0.6	0.5	0.5	0.5
A3	A13 ~ A16	0.1	0.1	0.3	0.2	0.1	0.1	0.1	0.1
A4	A17 ~ A18	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
A5	A19 ~ A22	1.3	1.4	2.1	2.7	1.4	1.3	1.3	1.3
A6	A23 ~ A30	0.5	0.6	1.5	1.9	0.5	0.5	0.5	0.5
A7	A31 ~ A34	0.2	0.1	0.6	0.7	0.1	0.1	0.1	0.1
A8	A35 ~ A36	0.1	0.1	0.3	0.8	0.1	0.1	0.1	0.1
A9	A37 ~ A40	0.5	0.6	2.2	1.4	0.5	0.6	0.7	0.5
A10	A41 ~ A48	0.5	0.5	1.9	1.2	0.5	0.5	0.6	0.5
A11	A49 ~ A52	0.1	0.1	0.2	0.8	0.1	0.1	0.1	0.1
A12	A53 ~ A54	0.2	0.1	0.9	0.1	0.1	0.1	0.1	0.1
A13	A55 ~ A58	0.2	0.2	0.4	0.4	0.2	0.2	0.2	0.2
A14	A59 ~ A66	0.5	0.5	1.9	1.9	0.5	0.6	0.6	0.6
A15	A67 ~ A70	0.5	0.4	0.7	0.9	0.3	0.4	0.4	0.4
A16	A71 ~ A72	0.7	0.6	1.6	1.3	0.7	0.6	0.4	0.6
Weight (kg)		400.6	387.94	1089.88	1069.79	388.94	385.54	389.49	385.54
No. of analyses		N/A	N/A	N/A	150,000	50,000	5330	5400	3625

Table 4.19 Optimal design comparison for the 72-bar truss (Case 2)

		Wu and Chow [30]	Li et al. [28]			Kaveh and Talatahari			Kaveh and Bakhshpoori [33]	Present work [2]	
Element group		GA	PSO	PSOPC	HPSO	DHPSACO [31]	CSS [23]	ICA [32]	CS	MCSS	IMCSS
1	AA4	0.196	7.22	4.49	4.97	1.8	1.99	1.99	1.8	1.8	1.8
2	A5 ~ A12	0.602	1.8	1.457	1.228	0.442	0.442	0.442	0.563	0.563	0.563
3	A13 ~ A16	0.307	1.13	0.111	0.111	0.141	0.111	0.111	0.111	0.111	0.111
4	A17 ~ A18	0.766	0.2	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
5	A19 ~ A22	0.391	3.09	2.62	2.88	1.228	0.994	1.228	1.266	1.457	1.228
6	A23 ~ A30	0.391	0.79	1.13	1.457	0.563	0.563	0.602	0.563	0.563	0.563
7	A31 ~ A34	0.141	0.56	0.196	0.141	0.111	0.111	0.111	0.111	0.111	0.111
8	A35 ~ A36	0.111	0.79	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
9	A37 ~ A40	1.8	3.09	1.266	1.563	0.563	0.563	0.563	0.563	0.563	0.391
10	A41 ~ A48	0.602	1.23	1.457	1.228	0.563	0.563	0.563	0.442	0.442	0.563
11	A49 ~ A52	0.141	0.11	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
12	A53 ~ A54	0.307	0.56	0.111	0.196	0.25	0.111	0.111	0.111	0.111	0.111
13	A55 ~ A58	1.563	0.99	0.442	0.391	0.196	0.196	0.196	0.196	0.196	0.196
14	A59 ~ A66	0.766	1.62	1.457	1.457	0.563	0.563	0.563	0.602	0.563	0.563
15	A67 ~ A70	0.141	1.56	1.228	0.766	0.442	0.442	0.307	0.391	0.307	0.307
16	A71 ~ A72	0.111	1.27	1.457	1.563	0.563	0.766	0.602	0.563	0.766	0.563
Weight (lb)		427.20	1209	941.82	933.09	393.38	393.06	392.84	389.87	393.13	389.6
No. of analyses		N/A	N/A	150,000	50,000	5330	7000	4500	4840	4775	4325

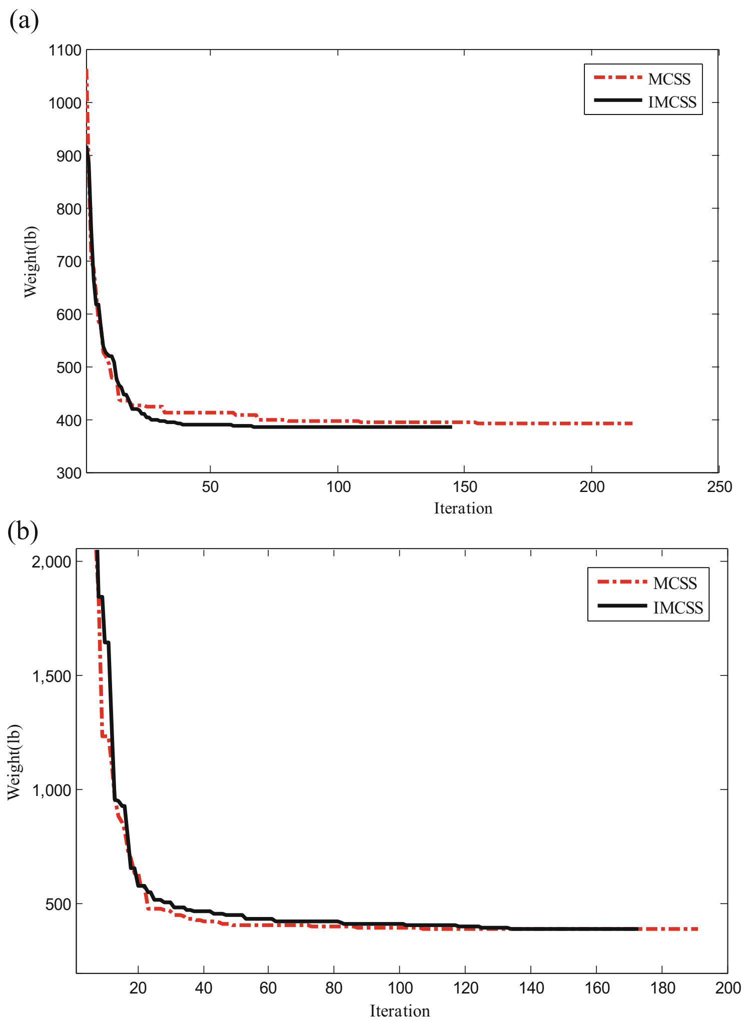


Fig. 4.15 Convergence curves for the 72-bar truss structure using MCSS and IMCSS [2]. (**a**) Case 1 and (**b**) Case 2

Example 4 A 120-bar dome-shaped truss

The 120-bar dome truss was first analyzed by Soh and Yang [34] to obtain the optimal sizing and configuration variables, but for this study only sizing variables are considered to minimize the structural weight in this example, similar to Lee and Geem [27] and Keleşoğlu and Ülker [35].

The geometry of this structure is shown in Fig. 4.16. 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).

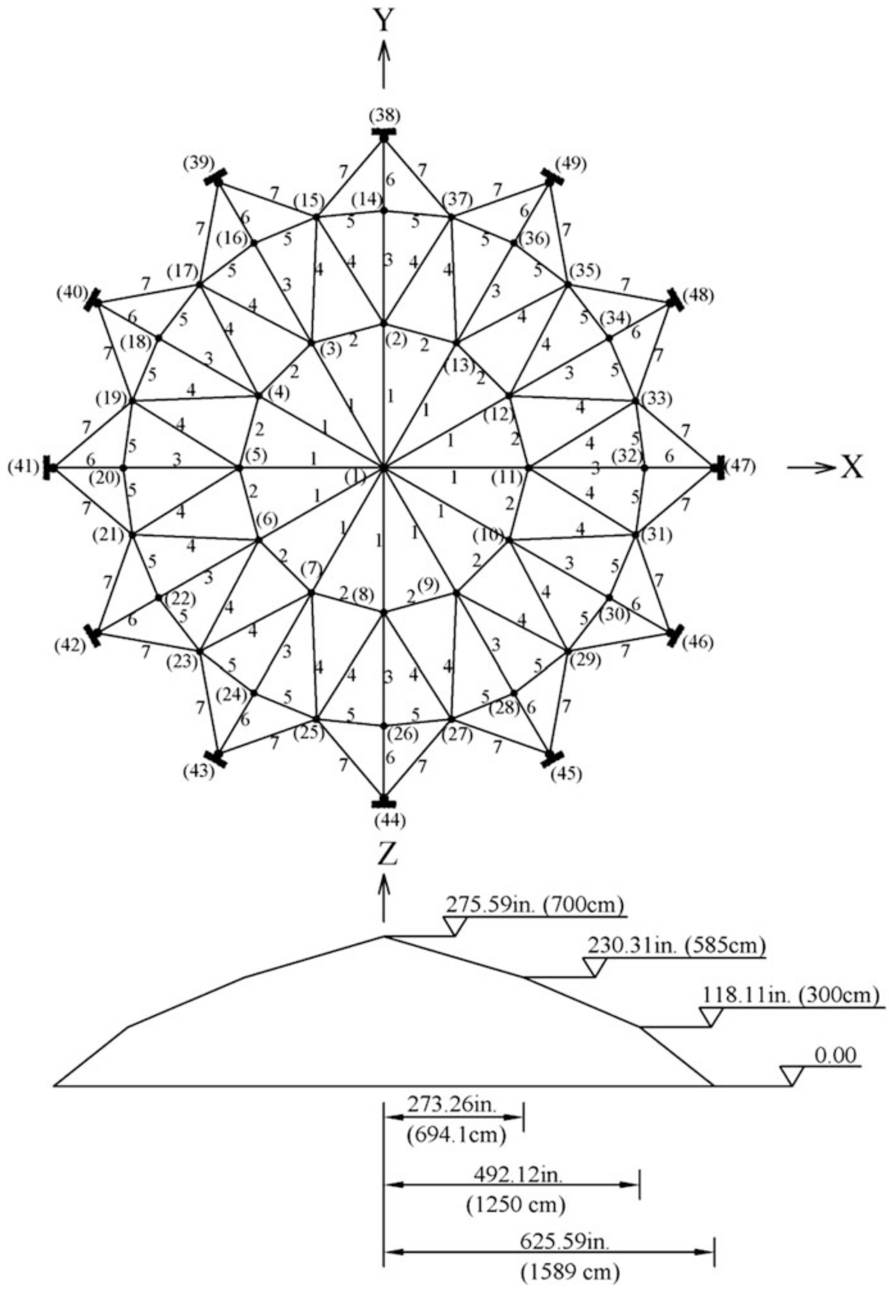


Fig. 4.16 Schematic of a 120-bar dome-shaped truss

The allowable tensile and compressive stresses are used according to the ASD-AISC code [25], as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y \ for \ \sigma_i \geq 0 \\ \sigma_i^- \qquad\quad for \ \sigma_i < 0 \end{cases} \tag{4.47}$$

where σ_i^- is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_c} - \dfrac{\lambda_i^3}{8C_c^3}\right) & \text{for } \lambda_i < C_c \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \tag{4.48}$$

where E is the modulus of elasticity, F_y is the yield stress of steel, C_c is the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions $\left(C_c = \sqrt{2\pi^2 E/F_y}\right)$, λ_i is the slenderness ratio ($\lambda i = kL_i/r_i$), k is the effective length factor, L_i is the member length, and r_i is the radius of gyration. The radius of gyration (r_i) can be expressed in terms of cross-sectional areas, i.e., $r_i = aA_i^b$. Here, a and b are the constants depending on the types of sections adopted for the members such as pipes, angles, and tees. In this chapter, pipe sections (a = 0.4993 and b = 0.6777) were adopted for bars [36].

All members of the dome are categorized into seven groups, as shown in Fig. 4.16. The dome is considered to be subjected to vertical loading at all the unsupported joints. These were 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).

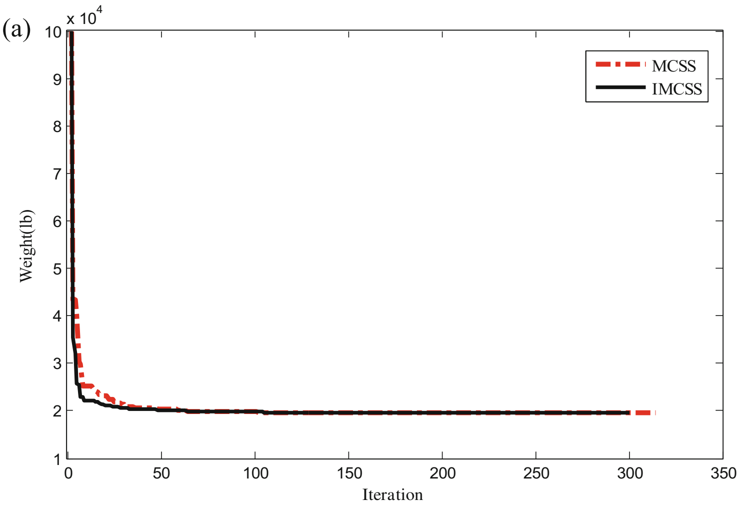
In this example, two cases of constraints are considered:

Case 1, with stress constraints and no displacement constraints, and Case 2, with stress constraints and displacement limitations of ±0.1969 in (5 mm) imposed on all nodes in x and y directions. For two cases, the maximum cross-sectional area is 5.0 in^2 (32.26 cm^2).

Figure 4.17 shows the convergence history for both cases, and Table 4.20 gives the best solution vectors and weights for both cases.

In Case 1, the best weights of MCSS and IMCSS are 19,607.39 lb and 19,476.92 lb, respectively, while for the Ray, HPSACO, and PSOPC are 19,476.19 lb, 19,491.30 lb, and 19,618.7 lb. The MCSS and IMCSS find the best solutions in 314 iterations (7850 analyses) and 299 iterations (7475 analyses), respectively, but for Ray and HPSACO algorithms, it takes 19,950 and 10,025 analyses to reach the best solutions, respectively.

In Case 2, the MCSS and IMCSS algorithms need 386 iterations (9650 analyses) and 324 iterations (8100 analyses) to find the best solutions, respectively, while for Ray and HPSACO algorithms, 19,950 and 10,075 analyses are required. The best weights obtained from MCSS and IMCSS algorithms are 19,928 lb and

(a)


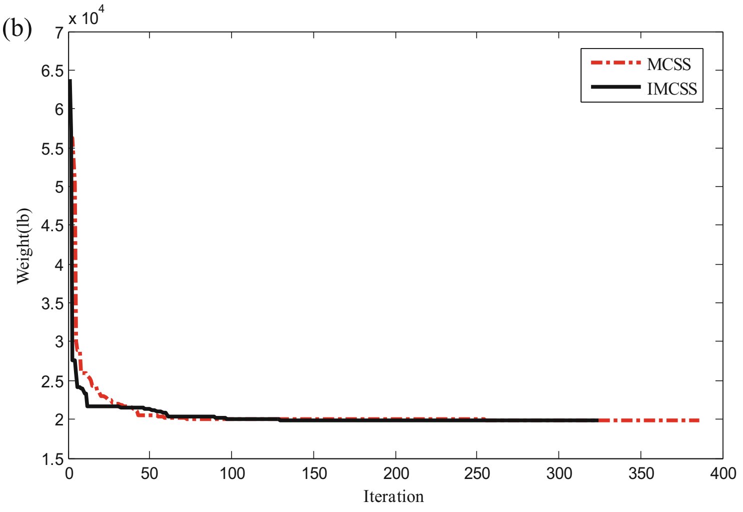
(b)


Fig. 4.17 Comparison of the convergence curves between the MCSS and IMCSS for the 120-bar dome truss structure [2], (**a**) Case 1 and (**b**) Case 2

19,796.71 lb, respectively, but from the Ray, HPSACO, and PSOPC are 20,071.9 lb, 20,078, and 20,681.7 lb, respectively.

Some design examples as benchmark problems are optimized using the IMCSS algorithm for both continuous and discrete design variables. The aim of this study is to find the best merit function, i.e., considering both penalty and cost functions. In comparing the results with those of the previous studies for all examples, the IMCSS has the better merit function than all of previous algorithms; however for few examples the best weight obtained from IMCSS algorithm is not the best in the

Table 4.20 Optimal design comparison for the 120-bar dome truss (two cases) optimal cross-sectional areas (in^2)

	Case 1						
	Lee and Geem [27]			Kaveh and Talatahari [29]	Kaveh and Khayatazad [37]	Present work [2]	
Element group	HS	PSO	PSOPC	HPSACO	Ray	MCSS	IMCSS
1	3.295	3.147	3.235	3.311	3.128	3.1108	3.1208
2	3.396	6.376	3.37	3.438	3.357	3.3903	3.3566
3	3.874	5.957	4.116	4.147	4.114	4.106	4.111
4	2.571	4.806	2.784	2.831	2.783	2.7757	2.7811
5	1.15	0.775	0.777	0.775	0.775	0.9674	0.8055
6	3.331	13.798	3.343	3.474	3.302	3.2981	3.3001
7	2.784	2.452	2.454	2.551	2.453	2.4417	2.4451
Weight (lb)	19,707.77	32,432.9	19,618.7	19,491.3	19,476.19	19,607.39	19,476.92
No. of analyses	35,000	N/A	125,000	10,025	19,950	7850	7475
	Case 2						
	Lee and Geem [27]			Kaveh and Talatahari [29]	Kaveh and Khayatazad [37]	Present work	
Element group	HS	PSO	PSOPC	HPSACO	Ray	MCSS	IMCSS
1	3.296	15.978	3.083	3.779	3.084	3.309	3.3187
2	2.789	9.599	3.639	3.377	3.360	2.6316	2.4746
3	3.872	7.467	4.095	4.125	4.093	4.2768	4.2882
4	2.57	2.79	2.765	2.734	2.762	2.7918	2.8103
5	1.149	4.324	1.776	1.609	1.593	0.9108	0.7753
6	3.331	3.294	3.779	3.533	3.294	3.5168	3.523
7	2.781	2.479	2.438	2.539	2.434	2.3769	2.3826
Weight (lb)	19,893.34	41,052.7	20,681.7	20,078	20,071.9	19,928	19,796.71
No. of analyses	35,000	N/A	125,000	10,075	19,950	9650	8100

results. Also, the results demonstrate the effectiveness of improvement process for MCSS algorithm to achieve a better convergence and find better solutions especially in final iterations of the improved algorithm.

References

1. Kaveh A, Motie Share MA, Moslehi M (2013) A new meta-heuristic algorithm for optimization: magnetic charged system search. Acta Mech 224(1):85–107
2. Kaveh A, Jafarvand A, Mirzaei B (2014) An improved magnetic charged system search for optimization of truss structures with continuous and discrete variables. Asian J Civil Eng 15 (1):95–105
3. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
4. Halliday D, Resnick R, Walker J (2008) Fundamentals of physics, 8th edn. Wiley, New York
5. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
6. Kaveh A, Talatahari S (2010) Optimal design of truss structures via the charged system search algorithm. Struct Multidiscip Optim 37(6):893–911
7. Hines W, Montgomery D (1990) Probability and statistics in engineering and management science, 3rd edn. Wiley, New York
8. Suganthan PN, Hansen N, Liang, JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005
9. Garcia S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms behavior: a case study on the CEC 2005 special session on real parameter optimization. J Heuristics 15:617–644
10. Belegundu AD (1982) A study of mathematical programming methods for structural optimization. Ph.D. thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa, USA
11. Arora JS (2000) Introduction to optimum design. McGraw-Hill, New York (1989)
12. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
13. Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 16:193–203
14. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intel 20:89–99
15. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37(4):443–473
16. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182
17. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind Ser B 98(3):1021–1025
18. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29 (11):2013–2015
19. Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, Kissimine, FL, pp 95–105
20. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Trans ASME J Mech Des 116:318–320

21. Deb K, Gene AS (1997) A robust optimal design technique for mechanical component design. In: Dasgupta D, Michalewicz Z (eds) Evolutionary algorithms in engineering applications. Springer, Berlin, pp 497–514
22. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579
23. Kaveh A, Talatahari S (2010) A charged system search with a fly to boundary method for discrete optimum design of truss structures. Asian J Civil Eng 11(3):277–293
24. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181
25. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. AISC, Chicago, IL
26. Camp C, Pezeshk S, Cao G (1998) Optimized design of two dimensional structures using a genetic algorithm. J Struct Eng ASCE 124(5):551–559
27. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
28. Li LJ, Huang ZB, Liu F, Wu QH (2007) A heuristic particle swarm optimizer for optimization of pin connected structures. Comput Struct 85:340–349
29. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283
30. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56(6):979–991
31. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Constr Steel Res 65(8–9):1558–1568
32. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
33. Kaveh A, Bakhshpoori T (2013) Optimum design of space trusses using cuckoo search algorithm with lévy flights. Iran J Sci Technol Trans Civil Eng 37(C1):1–15
34. Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. J Comput Civil Eng ASCE 10(2):143–150
35. Keleşoğlu O, Ülker M (2005) Fuzzy optimization geometrical nonlinear space truss design. Turkish J Eng Environ Sci 29:321–329
36. Saka MP (1990) Optimum design of pin-jointed steel structures with practical applications. J Struct Eng ASCE 116:2599–2620
37. Kaveh A, Khayatazad M (2013) Ray optimization for size and shape optimization of truss structures. Comput Struct 117:82–94