Acta Mech
DOI 10.1007/s00707-012-0745-6

A. Kaveh · Mohammad A. Motie Share · M. Moslehi

Magnetic charged system search: a new meta-heuristic algorithm for optimization

Received: 12 July 2012 / Revised: 12 September 2012

Abstract An improved version of the charged system search (CSS) algorithm is introduced which is called magnetic charged system search (MCSS). In the new algorithm, magnetic forces are considered in addition to electrical forces, using the Biot–Savart law. Each charged particle (CP), as a search agent, exerts magnetic forces on other CPs based on the variation of its objective function value during its last movement and its distance between other CPs. This additional force provides useful information for the optimization process and enhances the performance of the CSS algorithm. The efficiency of the MCSS is examined by application of this algorithm to well-known mathematical benchmarks and three well-studied engineering design problems. The results are compared to those of the CSS, and the improvements are highlighted.

1 Introduction

Optimization algorithms can be divided into two groups consisting of mathematical programming and meta-heuristic methods. Earlier optimization methods or classical optimization methods were all based on the mathematical programming. Many different mathematical programming methods have been proposed and developed since past decades. Linear programming, convex programming, integer programming, quadratic programming, and dynamic programming are some of these approaches that have been utilized for optimization problems. These methods usually provide accurate solutions; however, most of these need the gradient information of the objective function and a suitable initial point. These methods require the problem to be defined in a continuous space, while many optimization problems, such as engineering problems, are defined in discrete spaces.

The second generation of the optimization methods are meta-heuristic that are proposed to solve more complex problems. Every meta-heuristic method consists of a group of search agents that explore the feasible region based on both randomization and some specified rules. The rules are usually inspired by natural phenomena laws. Genetic algorithms (GA) proposed by Holland [1] and Goldberg [2] are inspired by Darwin's theory about biological evolutions. Particle swarm optimization (PSO) proposed by Eberhart and Kennedy [3] simulates social behavior, and it is inspired by the movement of organisms in a bird flock or fish school. Ant colony optimization (ACO) formulated by Dorigo [4] imitates foraging behavior of ant colonies. Many other natural-inspired algorithms such as simulated annealing (SA) proposed by Kirkpatrick et al. [5], Harmony Search (SA) presented by Geem et al. [6], Gravitational Search Algorithm (GSA) proposed by Rashedi et al.

A. Kaveh (✉)
Centre of Excellence for Fundamental Studies in Structural Engineering,
Iran University of Science and Technology, Narmak, Tehran-16, Iran
E-mail: alikaveh@iust.ac.ir

M. A. Motie Share · M. Moslehi
School of Civil Engineering, College of Engineering, University of Tehran, Tehran, Iran

Published online: 19 October 2012

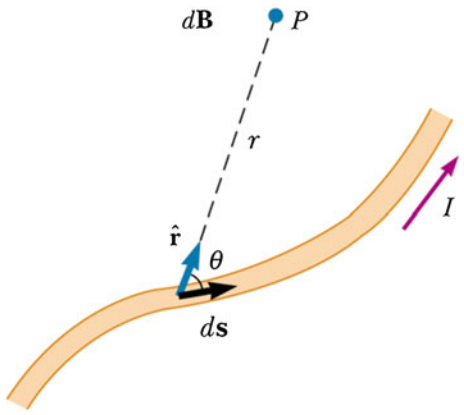

Fig. 1 The magnitude of the magnetic field d**B** at point P due to current I through a length element d**s** given by Biot–Savart law

[7], Big Bang–Big Crunch algorithm (BB–BC) proposed by Erol and Eksin [8] and improved by Kaveh and Talathari [9] have been proposed in recent years. Due to their good performance and ease of implementation, these methods have been widely applied to various problems in different fields of science and engineering. Structural optimization [10–13] is one of the active branches of the applications of optimization algorithms. One of the most recent meta-heuristic algorithms is the charged system search (CSS) proposed by Kaveh and Talathari [14] that 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.

In this paper, an improved CSS algorithm which is called magnetic charged system search (MCSS) is proposed. 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 Newtonian mechanical laws. The present paper is organized as follows: in Sect. 2, magnetic laws from physics are presented. Section 3 reviews the CSS. In Sect. 4, the MCSS algorithm is introduced. In Sect. 5, the MCSS is applied to different numerical examples to examine the efficiency of the new algorithm, and in Sect. 6, some concluding remarks are provided.

2 Magnetic laws

2.1 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 the electric field, the region surrounding any moving charged particle also contains magnetic fields. The existence of the magnetic field near the moving CPs was Oersted's discovery in 1819. He has shown that a compass needle is deflected by a current-carrying conductor. Shortly after this discovery, Biot and Savart proposed a mathematical expression so-called Biot–Savart 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. 1. Biot–Savart law is expressed [15] as:

$$\mathrm{d}\mathbf{B} = \frac{\mu_0}{4\pi}\frac{I\mathrm{d}\mathbf{s} \times \hat{\mathbf{r}}}{r^2}. \tag{1}$$

Here, d**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**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. 2a. By utilizing Biot–Savart law, the magnetic field produced by a wire at a point like P outside the wire can be determined as:

$$B = \frac{\mu_0}{2\pi}\frac{I}{r} \qquad \text{when: } r \geq R. \tag{2}$$

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

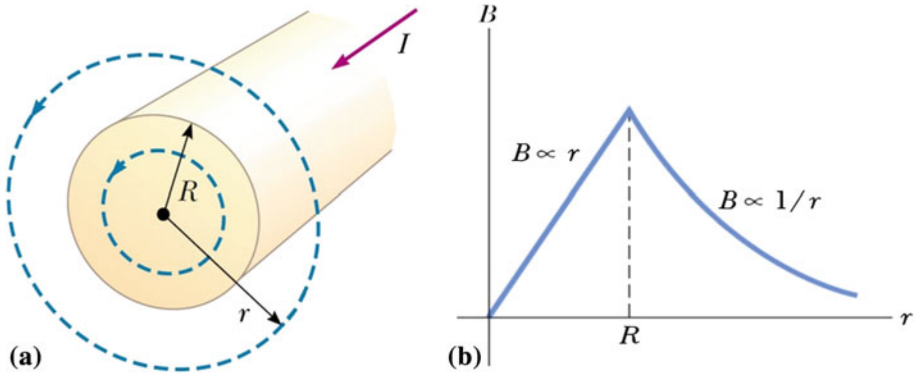

Fig. 2 **a** A wire carrying electric current I that is uniformly distributed in its cross-section. **b** A plot of the distribution of the magnetic field produced by a wire in the space

$$B = \left(\frac{\mu_0}{2\pi}\frac{I}{R^2}\right) \times r \qquad \text{when: } r < R. \tag{3}$$

With this formulation for the 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. (2) and (3) have an overlap, and both give identical magnitude for the magnetic field. A plot of these two equations from Ref. [15] is shown in Fig. 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:

$$B_P = \sum_{i=1}^{n} B_{ip}, \tag{4}$$

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 ith wire at point P which can be expressed as:

$$B_{ip} = \begin{cases} \frac{\mu_0}{2\pi}\frac{I}{r} & \text{for } r \geq R, \\ \left(\frac{\mu_0}{2\pi}\frac{I}{R^2}\right) \times r & \text{for } r < R. \end{cases} \tag{5}$$

2.2 Magnetic forces

When a charged particle moves in a magnetic field, a magnetic force $\mathbf{F}_B$ will be imposed on that moving charged particle. Experiments on CPs 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 the magnetic field $\mathbf{B}$.

By summarizing these observations, an expression for calculating the magnetic force is obtained [15] as:

$$\mathbf{F}_B = q\mathbf{v} \times \mathbf{B}, \tag{6}$$

where $\mathbf{B}$ is the magnetic field exerted on the moving charged particle. Here, the only source of the magnetic field is produced by the wires. Thus, the magnitude of $\mathbf{B}$ can be calculated using Eq. (5).

3 A review on charged system search algorithm

The CSS algorithm proposed by Kaveh and Talathari [14] is an evolutionary algorithm for optimization problems. This meta-heuristic optimization algorithm takes its inspiration from the physic laws governing a group of CPs. These charge particles are sources of the electric fields, and each CP can exert electric force on other CPs. Using the Newtonian mechanic laws, 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:

$$q_i = \frac{\text{fit}(i) - \text{fitworst}}{\text{fitbest} - \text{fitworst}}, \tag{7}$$

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

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

$$r_{ij} = \frac{||\mathbf{X}_i - \mathbf{X}_j||}{||(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{\text{best}}|| + \varepsilon}, \tag{8}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the position of the ith and jth CPs, and r_{ij} is the separation distance of these CPs. $\mathbf{X}_{\text{best}}$ 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 ith CP by the jth CP is expressed as:

$$p_{ij} = \begin{cases} 1 \Leftrightarrow \frac{\text{fit}(i) - \text{fitbest}}{\text{fit}(j) - \text{fit}(i)} > r \text{ and,} \quad \text{or,} \quad \text{fit}(j) > \text{fit}(i). \\ 0 \Leftrightarrow \text{else.} \end{cases} \tag{9}$$

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

$$\mathbf{F}_{E,j} = q_j \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{r_{ij}^2} \cdot w_2 \right) \cdot p_{ji} \cdot (\mathbf{X}_i - \mathbf{X}_j), \begin{cases} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R, \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R, \\ j = 1, 2, \ldots, N. \end{cases} \tag{10}$$

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

$$\mathbf{X}_{j,\text{new}} = \text{rand}_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + \text{rand}_{j2} \cdot k_v \cdot \mathbf{V}_{j,\text{old}} \cdot \Delta t + \mathbf{X}_{j,\text{old}}, \tag{11}$$

$$\mathbf{V}_{j,\text{new}} = \frac{\mathbf{X}_{j,\text{new}} - \mathbf{X}_{j,\text{old}}}{\Delta t}, \tag{12}$$

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 velocity coefficient controls the influence of the previous velocity of the particles. In other words, this coefficient is related to the exploration ability of the algorithm. The acceleration coefficient affects the force acting on each CP, or it influences the exploitation ability of the algorithm. An efficient optimization algorithm should perform good exploration in early iterations and good exploitation in last iterations. Thus, the magnitude of the k_a and k_v is set to 0.5 which are linearly increased and decreased, respectively. Thus, k_a and k_v are expressed as:

$$k_a = 0.5(1 + \text{iter}/\text{iter}_{\text{max}}), \quad k_v = 0.5(1 - \text{iter}/\text{iter}_{\text{max}}), \tag{13}$$

where iter is the current iteration number, and iter$_{\text{max}}$ is the maximum number of iterations.

Step 3. Boundaries Constraints Handling

For handling boundary constraints, a harmony search-based approach is used. According to this method, any variable of each solution ($x_{i,j}$) that violates its corresponding boundary can be regenerated from CM as

- w.p. CMCR
  - select a new value for a variable from CM,
  - w.p. (1–PAR) do nothing,
  - w.p. PAR choose a neighboring value,
- w.p. (1–CMCR)
  - select a new value randomly,

where “w.p.” is the abbreviation of “with the probability”, CMCR (the Charge Memory Considering Rate) varying between 0 and 1 sets the rate of selecting a value in the new vector from historic values stored in CM, and (1–CMCR) sets the rate of randomly choosing one value from a possible range of values. The value (1–PAR) sets the rate of doing nothing, and PAR sets the rate of choosing a value from neighboring the best CP. For further details, the reader may refer to Ref. [14].

Step 4. CM Updating

If among all of the new CPs, there are better CP or CPs that have a better objective function value 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 5. Checking the Termination Criteria

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

4 Magnetic charged system search algorithm

4.1 Combination of magnetic and electric forces

The inspiration of the standard CSS algorithm is based on a group of CPs 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 Newtonian mechanics laws. Therefore, we have CPs that move in the search space. In physics, it has been shown that when a charged particle moves, it produces a magnetic field. This magnetic field can exert a magnetic force on other CPs. Thus, in addition to the electric forces, we should have magnetic forces. When a charged particle moves with velocity $\mathbf{v}$ in the presence of both electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$, it experiences both electric force $q\mathbf{E}$ and magnetic force $q\mathbf{v}\times\mathbf{B}$. The total force, known as the Lorentz force [15], exerted on the charged particle is:

$$\sum \mathbf{F} = \mathbf{F}_B + \mathbf{F}_E = q\mathbf{v} \times \mathbf{B} + q\mathbf{E} = q \cdot (\mathbf{v} \times \mathbf{B} + \mathbf{E}), \tag{14}$$

where $\mathbf{F}$ is the Lorentz force. Thus, the new algorithm, 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 CPs. From the 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 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 we have made changes in part of the algorithm where the forces are computed. By using the aforementioned physical laws about magnetic fields and forces in Sect. 2, magnetic forces can be determined. Each solution candidate $\mathbf{X}_i$ called CP (charged particle) contains electrical charge. These CPs produce electric fields and exert electric forces on each other. When a 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 have 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

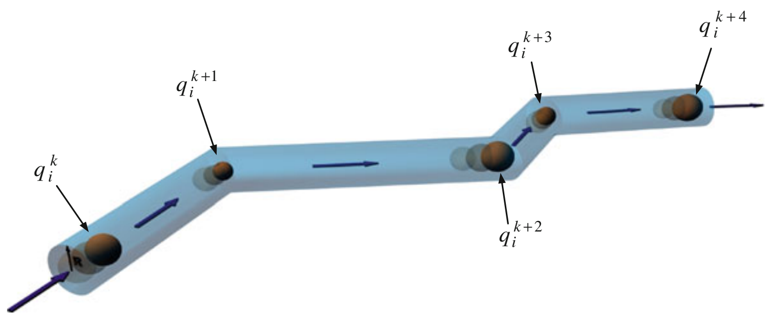

Fig. 3 A schematic view of the virtual wire (movement path of a CP), q_i^k is the charge of ith CP at end of the kth movement (kth iteration)

CPs, but during the movement, each wire remains straight, Fig. 3. The places that a wire changes its direction are 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. 2, a conducting wire carrying electric current can create magnetic fields in the space. Now virtual wires contain CPs 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 during the movement. 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:

$$I_{\text{avg}} = \frac{\Delta q}{\Delta t}. \tag{15}$$

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. (15) can be written as:

$$(I_{\text{avg}})_{ik} = q_i^k - q_i^{k-1}, \tag{16}$$

where $(I_{\text{avg}})_{ik}$is the average current in the ith wire where ith CP performs its kth movement (iteration) in it, and q_i^{k-1} and q_i^k are the charges of the ith CP at the start and end of its kth movement, respectively. Equation (16) 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 take 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.

Charge of the CPs is defined by Eq. (7). This expression for computing electric charges results in values between 0 to 1. This is due to normalization of the objective function of each CP. Therefore, the charges of the worst and best CP are always zero and unity, respectively. Now, consider a 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:

$$(I_{\text{avg}})_{ik} = \frac{q_{i,k} - q_{i,k-1}}{q_{i,k} + \varepsilon}, \tag{17}$$

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

$$(I_{\text{avg}})_{ik} = \text{sign}(\mathrm{d}f_{i,k}) \times \frac{\left|\mathrm{d}f_{i,k}\right| - \mathrm{d}f_{\min,k}}{\mathrm{d}f_{\max,k} - \mathrm{d}f_{\min,k}}, \tag{18}$$

$$\mathrm{d}f_{i,k} = \text{fit}_k(i) - \text{fit}_{k-1}(i), \tag{19}$$

where $\mathrm{d}f_{i,k}$ is the variation of the objective function of the ith CP in the kth movement (iteration). $\text{fit}_k(i)$ and $\text{fit}_{k-1}(i)$ are the values of the objective function of the ith CP at the start of the kth and $k-1$th iterations, respectively. The quantity $\mathrm{d}f_{i,k}$ can attain both positive and negative values. If we consider absolute values of $\mathrm{d}f$ for all of the current CPs, $\mathrm{d}f_{\max,k}$ and $\mathrm{d}f_{\min,k}$ will be the maximum and minimum values among these absolute values of $\mathrm{d}f$, respectively. Therefore, $\mathrm{d}f_{\max,k}$ and $\mathrm{d}f_{\min,k}$ are always positive quantities. It should be noted that in this paper the second expression (Eqs. (18), and (19)) is utilized for the computation of the electric current.

For computing the magnetic field in the place of each particle, we must compute the distance of that particle from the virtual wire. This distance is assumed to be the same as Eq. (8). Thus, r_{ij} now means the distance between ith wire and ith virtual CP to the jth charged particle.

In the expression for computing the magnetic force, Eq. (6), 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, $\mathbf{v}_{\text{rel}}$, is considered as:

$$\mathbf{v}_{\text{rel}} = \frac{\mathbf{X}_i - \mathbf{X}_j}{\Delta t}, \tag{20}$$

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

$$\mathbf{v}_{\text{rel}} = \mathbf{X}_i - \mathbf{X}_j. \tag{21}$$

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:

$$\mathbf{F}_{\mathbf{B},ji} = q_j \cdot \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2\right) \cdot \text{pm}_{ji} \cdot (\mathbf{X}_i - \mathbf{X}_j), \begin{cases} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R, \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R, \end{cases} \tag{22}$$

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:

$$\text{pm}_{ji} = \begin{cases} 1 \Leftrightarrow \ \text{fit}(i) > \text{fit}(j), \\ 0 \Leftrightarrow \ \text{else}, \end{cases} \tag{23}$$

where $\text{fit}(i)$ and $\text{fit}(j)$ are the objective values of the ith and jth CP, 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. (9). The electric probability considers a chance for both good and bad CPs to attract each other, but the magnetic probability 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, the electric current in virtual 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 $(\mathbf{X}_i - \mathbf{X}_j)$, 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 the 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 will be lost in the standard CSS, but MCSS considers this information and increases the efficiency of the algorithm.

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

$$\mathbf{F}_{\mathbf{B},j} = q_j \cdot \sum_{i,i \neq j} \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2\right) \cdot \text{pm}_{ji} \cdot (\mathbf{X}_i - \mathbf{X}_j), \begin{cases} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R, \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R, \\ j = 1, 2, \ldots, N, \end{cases} \tag{24}$$

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 places outside or inside of a virtual wire, the magnetic force that exerted on it is computed differently. With this formulation for the 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 is computed based on the inverse relation between distances, the magnitude of the forces will be increased due to decrease in 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. (3). This means that the formulation of the magnetic force for global and local phases should be separated, Eq. (24). A suitable value for R in Eq. (24) can be unity. However, by more investigating 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. (2) Using Eq. (2) 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. (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 paper and the magnetic force is revised in Eq. (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 another CP, regardless to 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:

$$\sum \mathbf{F}_j = \mathbf{F}_{B,j} + \mathbf{F}_{E,j} = q_j \sum_{i,i\neq j} \left(\frac{I_i}{r_{ij}} \cdot \mathrm{pm}_{ji} + \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{r_{ij}^2} \cdot w_2 \right) \cdot p_{ji} \right) \cdot (\mathbf{X}_i - \mathbf{X}_j),$$
$$\begin{cases} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R, \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R, \\ j = 1, 2, \ldots, N, \end{cases} \tag{25}$$

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 a number of other CPs charge. 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 that have 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 the optimization process will be lost. Thus, in the MCSS algorithm, magnetic forces are included to prevent the loss of this kind of information which benefits the algorithm. 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 acted on CPs by the ith CP is 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. (13) 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.

4.2.1 Challenges of MCSS and the solutions

Like other meta-heuristic algorithms, both CSS and MCSS algorithms are sensitive to the number of searching agents (CPs). This sensitivity may affect the algorithm when the number of CPs is decreased. The main reason is that all of the electrical forces are attractive forces, so CPs may be trapped in part of a search space (Local optima). Self-adaption, cooperation, and competition are three steps in the algorithm that automatically help the CPs to escape from local optima, but in some cases, when the number of CPs is relatively small, the aforementioned concern can be increased. In these situations, due to a lack of information gathered in the optimization process, attractive forces can make the CPs to achieve a local optimum. For example, if some of CPs in the first iterations recognize a local optimum, they try to attract other CPs by producing attractive electrical forces. These attractive forces may prevent other CPs from exploring other spaces of search space. The existence of magnetic forces decreases this problem of the algorithm because magnetic forces are both attractive and repelling forces, and they provide a better search ability for the algorithm, but attractive electrical forces still exist, and they may have some impacts on the algorithm when the number of CPs is small. For addressing this problem, two modifications for the algorithm can be considered. The first modification will be in the formulation of forces as

$$\mathbf{F} = p_r \times \mathbf{F}_E + \mathbf{F}_B, \tag{26}$$

where p_r is the probability that an electrical force is a repelling force, and it is defined as

$$p_r = \begin{cases} 1 \Leftrightarrow \ \text{rand} > 0.1 \cdot (1 - \text{iter}/\text{iter}_{\max}), \\ -1 \Leftrightarrow \ \text{else}, \end{cases} \tag{27}$$

where rand is a random number generated based on a uniform distribution, iter is the current number of iterations, and $\text{iter}_{\max}$ is the maximum number of iterations. According to Eq. (27), in the initial iterations of the optimization process, 10 % of attractive electrical forces are randomly changed to repelling forces, and when the number of iterations rises, this amount is linearly decreased to zero. By applying this parameter to electrical forces, the exploration ability of the algorithm in the initial steps of optimization will be increased, and this ability is gradually decreased, and the exploitation ability will be increased. p_r is a control parameter for the algorithm that can help MCSS to perform better with a small number of CPs.

The second modification is about the concept of time in the algorithm. From the optimization point of view, the *iteration* term is used for the time, and thus, it is assumed that the time alters discretely. This means that all alterations are performed when all agents have created their solutions. For example, in MCSS algorithm, after computation of all of forces, all of the CPs move to their new locations, and then CM updating is fulfilled. All these conform to a discrete time concept. In this situation, nothing is changed within an iteration. A more continuous concept of time can be utilized if we have changes within an iteration. According to this concept, time changes continuously, and after creating just one solution, all updating processes such as charge, current, and CM updating can be performed. Using this methodology, the new positions of each CP and what it experienced in its last movement can affect the moving process of the subsequent CPs. This method can save a lot of useful information for optimization processes. Based on explained steps and rules, MCSS algorithm can be summarized as a pseudo-code in Fig. 4.

5 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. 5.2.3, three well-studied engineering design problems are solved by MCSS and the results are compared to those of the CSS.

MCSS Algorithm

```
Randomly initialize positions and velocities of all CPs (from the range of [$X_{i,min}$,$X_{i.max}$]);
Set initial electric currents of the CPs equal to zero;
Generate CM from first population of CPs;
WHILE (i<Maximum number of iterations)
FOR (each CP j)
Calculate Electrical forces exerted on jth CP: Using Eq. (10);
Calculate Magnetic forces exerted on jth CP: Using Eq. (24);
Calculate Net force exerted on jth CP: Using Eq. (26);
Move jth CP to its new position: Using Eq.(11);
IF (each component of the vector of the new CP violates is corresponding boundary)
Use Harmony Search approach for handling the boundaries constraints;
END IF
Calculate new charge, velocity and electric current of jth CP: Using Eqs. (7), (12), (18)
Update Charge Memory (CM);
END FOR
IF (in the new generation of CPs the so far best CPs are lost)
Implement CM to current CPs;
END IF
i=i+1;
END WHILE
Postprocess results and visualization
```

Fig. 4 Pseudo-code of the MCSS

5.1 Mathematical benchmark functions

5.1.1 Comparison between MCSS, CSS and a set of genetic algorithms

In this section, some mathematical benchmarks are chosen from Ref. [16] and optimized using the MCSS algorithm. The description of these mathematical benchmarks is illustrated in Table 1.

5.1.1.1 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 CM size (CMS), the number of CPs, and the maximum number of iterations 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. [14]. 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 2, the results of the MCSS are compared to the results obtained by the CSS from Ref. [14], and GA and some of its variants derived from [16]. For a fair comparison between MCSS and CSS, the random initial solutions of each runs are the same. The numbers in Table 2 indicate the average number of function evaluations from 50 independent runs. The numbers in parentheses 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., $\varepsilon = |f_{\min} - f_{\text{final}}| = 10^{-4}$. The results verify the efficiency of the MCSS algorithm compared to the CSS and other GAs. 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 very few number of CPs and a small value for the 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 comparison is performed to show the difference between the CSS and MCSS algorithm in unsuitable situations, i.e., a 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 100. This means that the computational cost is one-quarter of the previous comparisons. The results of this comparison are presented in Table 3.

Table 1 Description of the mathematical benchmarks

Function name	Interval	Function	Global minimum
Aluffi-Pentiny	$\mathbf{X} \in \left[-10,\ 10\right]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in \left[-100,\ 100\right]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in \left[-50,\ 50\right]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cdot\cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in \left[-10,\ 10\right]^2$	$f(\mathbf{X}) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	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	$\mathbf{X} \in \left[-5,\ 5\right]^2$	$f(\mathbf{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.0316
Cb3	$\mathbf{X} \in \left[-5,\ 5\right]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$\mathbf{X} \in \left[-1,\ 1\right]^n,\ n = 4$	$f(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in \left[-5.12,\ 5.12\right]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$\mathbf{X} \in \left[-1,\ 1\right]^n,\ n = 2, 4, 8$	$f(\mathbf{X}) = -\exp\left(-0.5\cdot\sum_{i=1}^{n} x_i^2\right)$	−1.0
Goldstein and price	$\mathbf{X} \in \left[-2,\ 2\right]^2$	$f(\mathbf{X}) = [1 + (x_1 + x_2 + 1)^2\cdot(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]\times$ $[30 + (2x_1 - 3x_2)^2\cdot(18 - 32x_1 - 12x_1 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	3.0
Griewank	$\mathbf{X} \in \left[-100,\ 100\right]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Hartman 3	$\mathbf{X} \in \left[-30,\ 30\right]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right),\ a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix},\ c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$	−3.862782
		$p = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$	
Hartman 6	$\mathbf{X} \in \left[0,\ 1\right]^6$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2\right),\ a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$	−3.322368
		$c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix},\ p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	
Rastrigin	$\mathbf{X} \in \left[-1,\ 1\right]^2$	$f(\mathbf{X}) = \sum_{i=1}^{2}\left(x_i^2 - \cos(18x_i)\right)$	−2.0
Rosenbrock	$\mathbf{X} \in \left[-30,\ 30\right]^n,\ n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0.0

Table 2 Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS [14]	MCSS
AP	1,360 (0.99)	1,360	1,277	1,253	804	316
Bf1	3,992	3,356	1,640	1,615	1,187	464
Bf2	20,234	3,373	1,676	1,636	742	425
BL	19,596	2,412	2,439	1,436	423	361
Branin	1,442	1,418	1,404	1,257	852	332
Camel	1,358	1,358	1,336	1,300	575	342
Cb3	9,771	2,045	1,163	1,118	436	267
CM	2,105	2,105	1,743	1,539	1,563	421
Dejoung	9,900	3,040	1,462	1,281	630	334
Exp2	938	936	817	807	132	146
Exp4	3,237	3,237	2,054	1,496	867	284
Exp8	3,237	3,237	2,054	1,496	1,426	553
GP	1,478	1,478	1,408	1,325	682	358
Griewank	18,838 (0.91)	3,111 (0.91)	1,764	1,652 (0.99)	1,551	976
Hartman3	1,350	1,350	1,332	1,274	860	322
Hartman6	2,562 (0.54)	2,562 (0.54)	2,530 (0.67)	1,865 (0.68)	1,783	738
Rastrigin	1,533 (0.97)	1,523 (0.97)	1,392	1,381	1,402	1,042
Rosenbrock	9,380	3,739	1,675	1,462	1,452	1,214
Total	112,311 (96.7)	41,640 (96.7)	29,166 (98.16)	25,193 (98.16)	17,367	8,895

The numbers in the Table 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 reason of this problem is discussed in the last part of Sect. 4. Although MCSS is a promising algorithm when the number of CPs is a normal value, a considerable decrease in the number of CPs may have some impacts on the performance of the algorithm. As a solution, R-MCSS which is a modification of MCSS and presented in the last part of Sect. 4 can be used in these cases. The results of R-MCSS are also presented in Table 3. It is shown that R-MCSS has a relatively better performance when the number of CPs is small. The purpose of presenting Table 3 is just to magnify the difference between the CSS and MCSS algorithm and verify the better performance of the MCSS in more challenging situations. For a more detailed presentation, Fig. 5 illustrates the optimization process and convergence.

5.1.1.2 Statistical test Now in the following we want to ensure that the results of MCSS in Table 3 are better than the CSS algorithm. For this purpose, we apply a multi-problem 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 normality tests including Kolmogorov–Smirnov and Shapiro–Wilk test. The p values of the normality tests

Table 3 Numerical comparison between CSS and MCSS algorithms

Function	Global minimum	CSS	MCSS	CSS's error	MCSS's error
AP	−0.352386	−0.198721	−0.34974	0.153665	0.002646
Bf1	0.0	28.809183	1.435E−6	28.80918	1.44E−06
Bf2	0.0	8.938997	5.385E−6	8.938997	5.39E−06
BL	0.0	0.106252	2.542E−8	0.106252	2.54E−08
Branin	0.397887	3.960884	0.401754	3.562997	0.003867
Camel	−1.0316	−0.866765	−1.031598	0.164835	2E−06
Cb3	0.0	0.125161	2.427E−8	0.125161	2.43E−08
CM	−0.4	−0.230142	−0.398751	0.169858	0.001249
Dejoung	0.0	0.166451	4.731E−6	0.166451	4.73E−06
Exp2	−1.0	−0.999366	−0.999984	0.000634	1.6E−05
Exp4	−1.0	−0.990884	−0.999971	0.009116	2.9E−05
Exp8	−1.0	−0.949659	−0.999824	0.050341	0.000176
GP	3.0	15.729613	3.155378	12.72961	0.155378
Griewank	0.0	0.342795	6.815E−8	0.342795	6.82E−08
Hartman3	−3.862782	−3.491627	−3.861415	0.371155	0.001367
Hartman6	−3.322368	−2.054548	−3.320421	1.26782	0.001947
Rastrigin	−2.0	−1.875735	−1.998324	0.124265	0.001676
Rosenbrock	0.0	19.476846	0.074523	19.47685	0.074523

Number of CPs = 10, maximum number of iterations = 100

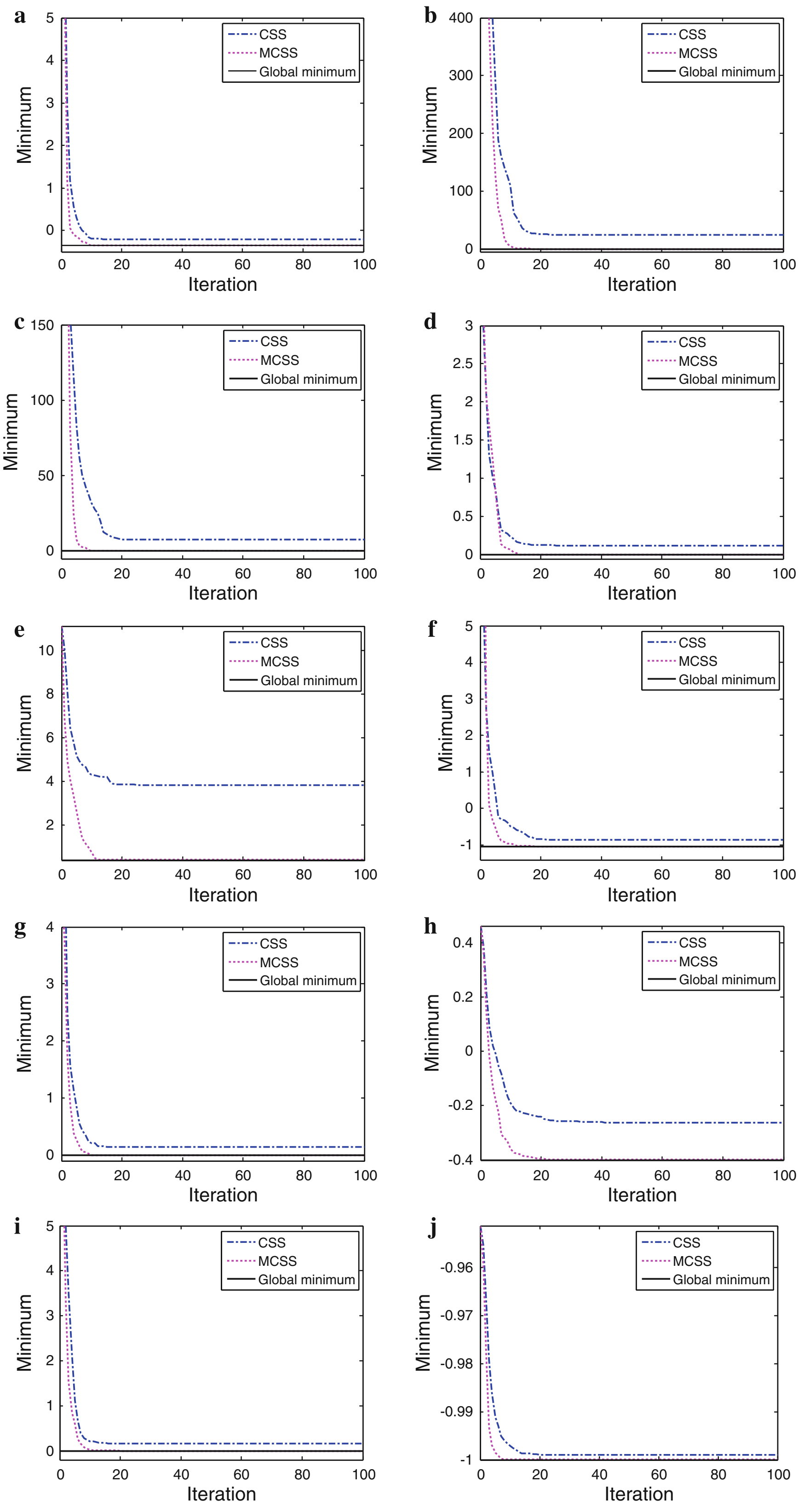

Fig. 5 Comparison of the convergance rate of optimizing mathematical benchmarks; **a** AP, **b** Bf1, **c** Bf2, **d** BL, **e** Branin, **f** Camel, **g** Cb3, **h** CM, **i** Dejoung, **j** Exp2, **k** Exp4, **l** Exp8, **m** Goldstein and Price, **n** Griewank, **o** Hartman3, **p** Hartman6, **q** Rastrigin, **r** Rosenbrock

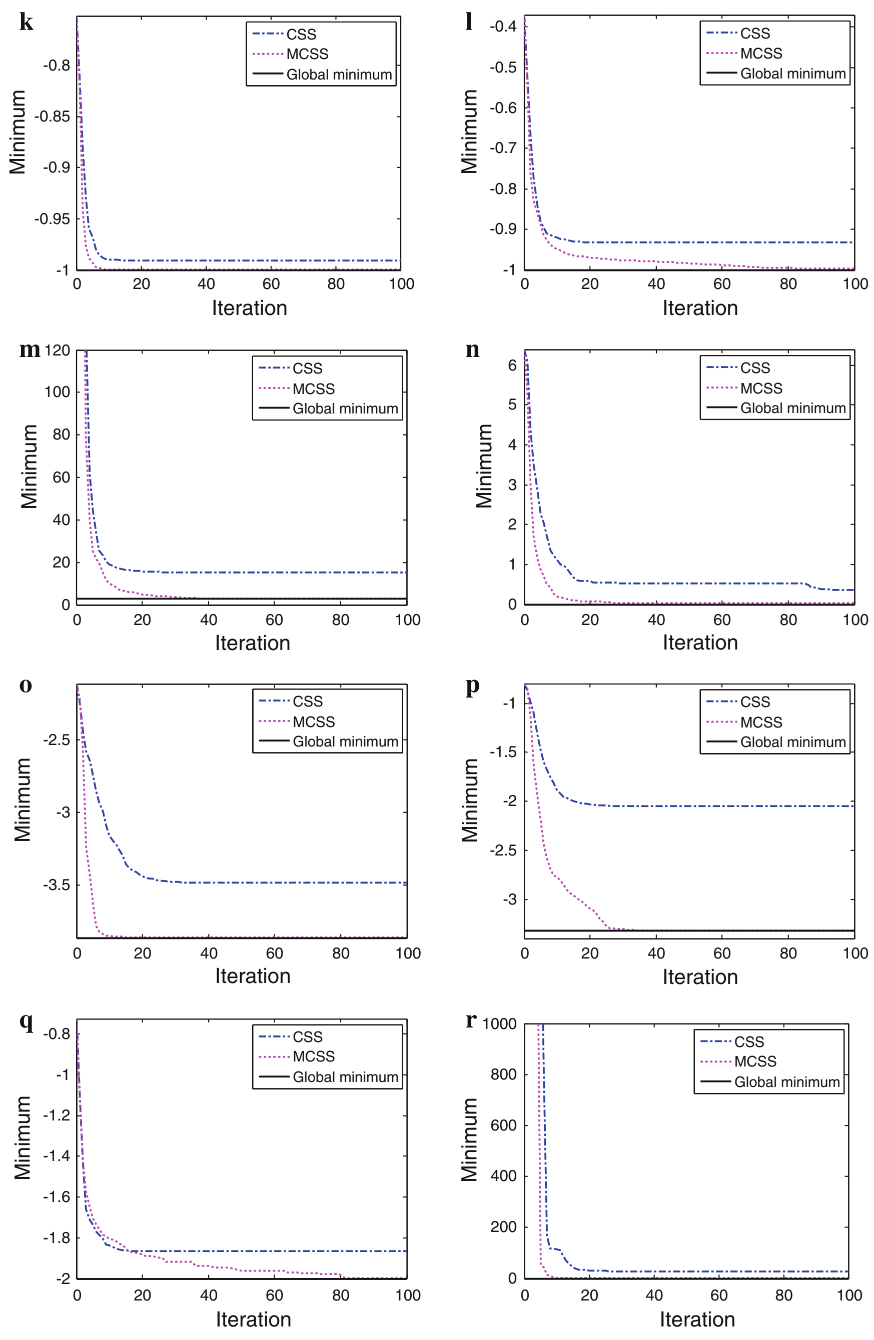

Fig. 5 continued

over the sample results obtained by CSS and MCSS are shown in Table 4. If we consider a significance level $\alpha = 0.05$, all of the p values in Table 4 will be less than 0.05, Thus, the sample results do not follow a normal distribution. The Q–Q plot for sample results is illustrated in Fig. 6, and it can be understood that the normality condition is not satisfied in both CSS and MCSS algorithms. This result was predictable because the sample size, i.e., 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. [17]. 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,

Table 4 Normality tests and their p values over multiple-problem analysis

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

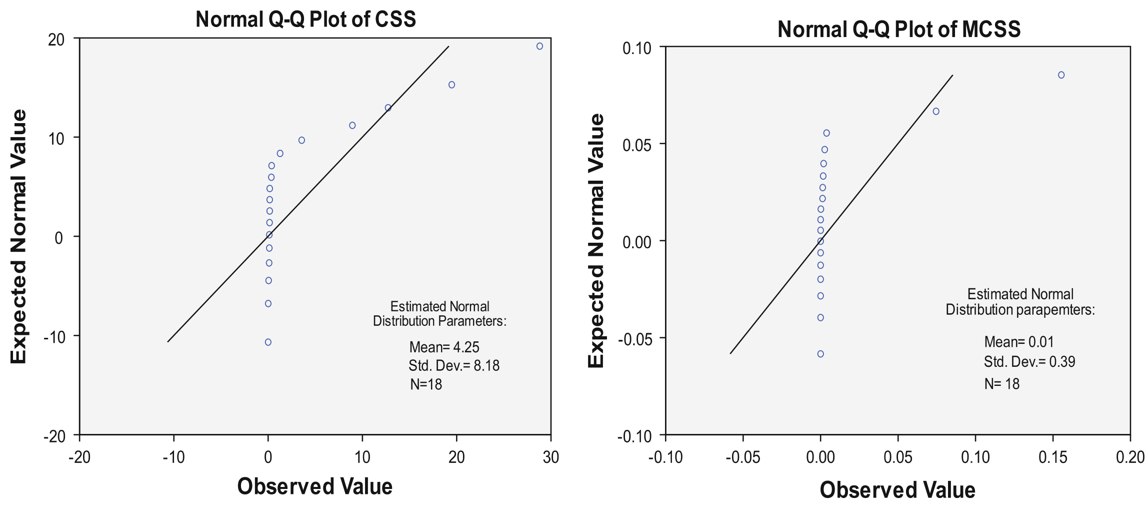

Fig. 6 Normal Q–Q plots of the sample results of the CSS and MCSS algorithms

because of better mean value of the MCSS algorithm results, MCSS outperforms its predecessor, CSS algorithm.

5.1.2 Comparison between MCSS and other state-of-art algorithms

5.1.2.1 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 IEEE Congress on Evolutionary Computation (CEC 2005) are solved by the MCSS algorithm. The detailed description of test functions is presented by Suganthan et al. [18]. The set of these test functions consists of the following functions:

- 5 Unimodals functions (f1–f5)
  - Sphere function displaced.
  - 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 Multimodals functions (f6–f7)
  - 7 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.
  - 2 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. [18]. 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 the maximum number of function evaluations is 10,000.

Table 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	1E−09
$f4$	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	1E−09
$f7$	0.01172	0.1971	0.03705	0.146	0.04519	0.4205	1E−09	0.231	1E−09	0.02	0.08261	1E−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	1E−09
$f10$	4.975	5.643	26.77	12.5	3.622	5.289	0.0796	0.239	40.8	4.969	7.304	1E−09
$f11$	2.334	4.557	9.029	0.847	4.623	3.944	0.934	6.65	3.65	4.891	1.91	0.235
$f12$	406.9	74.3	604.6	31.7	2.4001	442.3	29.3	149	209	4.5E−07	259.5	1.734
$f13$	0.7498	0.7736	1.137	0.977	0.3689	1.841	0.696	0.653	0.494	0.22	0.8379	0.246
$f14$	2.172	2.03	3.706	3.45	2.36	2.63	3.01	2.35	4.01	2.915	3.046	2.045
$f15$	400	269.6	293.8	259	4.854	365	228	510	211	32	253.8	114.278
$f16$	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	87.046
$f18$	420	803.3	901.4	400	760.7	483.2	332	752	497	719.4	439.6	438.156
$f19$	449	762.8	844.5	420	714.3	564.4	326	751	516	704.9	380	243.714
$f20$	446	800	862.9	460	822	651.9	300	813	442	713	440	473.579
$f21$	689.3	721.8	634.9	492	536	484	500	1050	404	464	680.1	436.614
$f22$	758.6	670.9	778.9	718	692.4	770.9	729	659	740	734.9	749.3	622.176
$f23$	638.9	926.7	834.6	572	730.3	640.5	559	1060	791	664.1	575.9	594.217
$f24$	200	224	313.8	200	224	200	200	406	865	200	200	172.921
$f25$	403.6	395.7	257.3	923	365.7	373	374	406	442	375.9	406	344.794
Mean	224.681	2144.92	272.417	189.725	203.541	213.481	152.662	273.153	70635.3	194.261	191.12	145.692

The termination criterion is either reaching the maximum number of function evaluations or achieving an error less than 10^{-8}. Table 5 shows the official results of the participated algorithms obtained from Garcia et al. [19]. The description of each algorithm is given in Ref. [19]. The results of the MCSS algorithm are added to Table 5. The values of Table 5 indicate the average error rate of each algorithm. This value can be considered as a means for measuring the performance of each algorithm.

5.1.2.2 Numerical results and statistical test As the results in Table 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 a statistical test to judge about the performance of MCSS in relation to other algorithms. We want to find out whether the results of MCSS have a significant difference in comparison with the other algorithms. This analysis is a multiple-problem analysis; therefore, a nonparametric test is more suitable in this case. We utilized the Wilcoxon's test. This test performs pairwise comparisons between two algorithms. In this test, MCSS is compared to other remaining algorithms.

Table 6 summarizes the results of applying the Wilcoxon test. Table 6 includes the sum of ranking and p value of each comparison. The method of this test is simply described in Ref. [17]. The significance level α is considered as 0.05. In each comparison where the corresponding p value is less than 0.05, it means that two compared algorithms behave differently, and the one with smaller mean value of error rates has a better performance.

The p value in pairwise comparison is independence 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 statistics 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. [19]). The true statistical significance for combining pairwise comparisons is given by:

$$p = 1 - \prod_{i=1}^{i=k-1} (1 - pH_i), \tag{28}$$

where k is the number of pairwise comparisons considered, and pH_i is the p value of each comparison. For more information, please refer to Ref. [19].

By evaluating the values of Table 6, the p value of all of the comparisons except MCSS versus G-CMA-ES is less than the significance level $\alpha = 0.05$, it cannot be concluded that MCSS is better than all of the algorithms except G-CMA-ES because we have to consider FWER in making a conclusion in multiple pairwise comparisons. The MCSS really outperforms all of the algorithms except G-CMA-ES considering independence pairwise comparisons due to the fact that achieved p values are less than $\alpha = 0.05$. The true p value for multiple pairwise comparisons can be computed using Eq. (28):

$$\begin{aligned} p = 1 - ((1 - 0.001) \cdot (1 - 0.0) \cdot (1 - 0.0) \cdot (1 - 0.006) \cdot (1 - 0.001) \cdot (1 - 0.0) \\ \cdot (1 - 0.0) \cdot (1 - 0.001) \cdot (1 - 0.003) \cdot (1 - 0.001)) = 0.0129. \end{aligned} \tag{29}$$

Based on this algorithm, it can be claimed that the MCSS algorithm has a better performance in relation to all of the algorithms except G-CMA-ES with a p value of 0.01294. As a result, if we consider a significance level $\alpha = 0.0129$, the confidence interval for the mentioned claim will be $100(1 - \alpha) = 98.7\%$.

Table 6 The Wilcoxon test results

MCSS versus	R^+	R^-	p value
BLX-GL50	23	208	0.001
BLX-MA	2	274	0.000
CoEVO	11	220	0.000
DE	37	194	0.006
DMS-L-PSO	17	214	0.001
EDA	0	231	0.000
G-CMA-ES	44	127	0.071
K-PCX	0	253	0.000
L-CMA-ES	13	177	0.001
L-SaDE	34	219	0.003
SPC-PNX	23	208	0.001

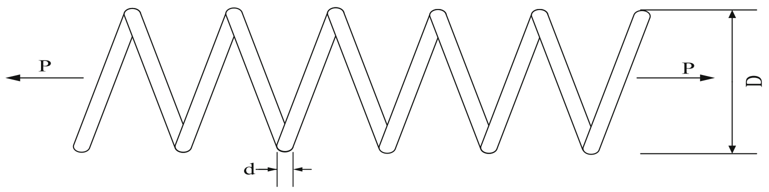

Fig. 7 A tension/compression spring

5.2 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. All of the engineering examples have nonlinear inequality constraints. The general form of these examples is

$$\begin{aligned}&\textbf{Minimize}\text{ objective function } f(\mathbf{X}),\\&\textbf{Subjected to}: \; g_i(\mathbf{X}) \le 0,\\&\mathbf{X}_{\min} \le \mathbf{X} \le .\mathbf{X}_{\max}\end{aligned} \tag{30}$$

For handling constraints, a simple penalty method is utilized to prevent adding the effect of a robust constraint handling method on the performance of the algorithm. Using the penalty method, the problem can be restated as

$$\textbf{Minimize } F(\mathbf{X}) = f(\mathbf{X}) + \sum_{i=1}^{n} (R_i \times \max(g_i(\mathbf{X}), 0)^2) \tag{31}$$

where n is the total number of constraints and R_i is the penalty factor that is set to a large positive number. Boundaries constraints will be handled by using the HS approach described in Sect. 3.

5.2.1 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 [14]. This problem is defined by Belegundu [20] and Arora [21] which is depicted in Fig. 7. The objective of this optimization problem is to minimize the weight of a tension/compression spring. This minimization involves some constraints, 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 of active coils $N(=x_3)$. By considering these decision variables, the cost function can be formulated as:

$$f_{\cos t}(\mathbf{X}) = (x_3 + 2)x_2 x_1^2, \tag{32}$$

$$\begin{aligned}g_1(\mathbf{X}) &= 1 - \frac{x_2^3 x_3}{71,785 \cdot x_1^4} \le 0,\\g_2(\mathbf{X}) &= \frac{4x_2^2 - x_1 x_2}{12,566 \cdot (x_2 x_1^3 - x_1^4)} + \frac{1}{5,108 \cdot x_1^2} - 1 \le 0,\\g_3(\mathbf{X}) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \le 0,\\g_4(\mathbf{X}) &= \frac{x_1 + x_2}{1.5} - 1 \le 0.\end{aligned} \tag{33}$$

The decision variables are limited as:

$$\begin{aligned}0.05 &\le x_1 \le 2,\\0.25 &\le x_2 \le 1.3,\\2 &\le x_3 \le 15.\end{aligned} \tag{34}$$

Table 7 Optimum results for the tension/compression spring design

Methods	Optimal design variables			
	$x_1(d)$	$x_2(D)$	$x_3(N)$	f_{cost}
Belegundu [20]	0.050000	0.315900	14.250000	0.0128334
Arora [21]	0.053396	0.399180	9.1854000	0.0127303
Coello [22]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [23]	0.051989	0.363965	10.890522	0.0126810
He and Wang [24]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [25]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [26]	0.051865	0.361500	11.000000	0.0126432
Kaveh and Talathari (CSS) [14]	0.051744	0.358532	11.165704	0.0126384
Present work	0.051627	0.356290	11.275456	0.0126069

Table 8 Statistical results of different methods for the tension/compression spring

Methods	Best	Mean	Worst	Standard deviation
Belegundu [20]	0.0128334	N/A	N/A	N/A
Arora [21]	0.0127303	N/A	N/A	N/A
Coello [22]	0.0127048	0.012769	0.012822	3.9390e−5
Coello and Montes [23]	0.0126810	0.0127420	0.012973	5.9000e−5
He and Wang [24]	0.0126747	0.012730	0.012924	5.1985e−5
Montes and Coello [25]	0.012698	0.013461	0.16485	9.6600e−4
Kaveh and Talatahari [26]	0.0126432	0.012720	0.012884	3.4888e−5
Kaveh and Talathari (CSS) [14]	0.0126384	0.012852	0.013626	8.3564e−5
Present work	0.0126069	0.012712	0.012982	4.7831e−5

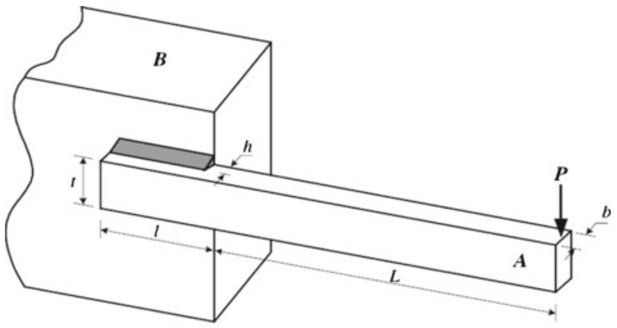

Fig. 8 A welded beam system

This problem has been solved with various methods by different researchers, Belegundu [20], Arora [21], Coello [22], Coello and Montes [23], He and Wang [24], Montes and Coello [25], and Kaveh and Talathari [14,26]. The results of the best solutions found by different methods are presented in Table 7. From Table 7, it can be understood that the best solution found by MCSS is better than other methods. The focus of this comparison is on the difference between CSS and MCSS methods. The results show that MCSS can improve the solution. The statistical simulation results of 30 independent runs for MCSS are illustrated in Table 8 and compared to other methods.

5.2.2 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. 8. The goal of this optimization problem is to minimize the constructing cost of a welded beam that is subjected to different constraints, 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 set-up, welding labor, and the materials costs, the cost function can be expressed as:

$$f_{\cos t}(\mathbf{X}) = 1.1047x_1^2x_2 + 0.04811x_3x_4 \cdot (14.0 + x_2), \tag{35}$$

subjected to the following constraints:

$$\begin{aligned}
&g_1(\mathbf{X}) = \tau(\{x\}) - \tau_{\max} \leq 0,\\
&g_2(\mathbf{X}) = \sigma(\{x\}) - \delta_{\max} \leq 0,\\
&g_3(\mathbf{X}) = x_1 - x_4 \leq 0,\\
&g_4(\mathbf{X}) = 0.10471x_1^2 + 0.04811x_3x_4 \cdot (14.0 + x_2) - 5.0 \leq 0,\\
&g_5(\mathbf{X}) = 0.125 - x_1 \leq 0,\\
&g_6(\mathbf{X}) = \delta(\{x\}) - \delta_{\max} \leq 0,\\
&g_7(\mathbf{X}) = P - P_c(\{x\}) \leq 0
\end{aligned} \tag{36}$$

where

$$\begin{aligned}
\tau(\mathbf{X}) &= \sqrt{(\tau')^2 + 2\tau' \cdot \tau'' \frac{x_2}{2R} + (\tau'')^2},\\
\tau' &= \frac{P}{\sqrt{2}x_1 \cdot x_2}, \tau'' = \frac{MR}{J},\\
M &= P \cdot \left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_2}{2}\right)^2},\\
J &= 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},\\
\sigma(\mathbf{X}) &= \frac{6PL}{x_4 \cdot x_3^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^2x_4},\\
P_c(\mathbf{X}) &= \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
P &= 6{,}000 \text{ lb}, L = 14 \text{ in},\\
E &= 30 \times 10^6 \text{psi}, G = 12 \times 10^6 \text{psi}.
\end{aligned} \tag{37}$$

And variable boundaries are:

$$\begin{aligned}
&0.1 \leq x_1 \leq 2,\\
&0.1 \leq x_2 \leq 10,\\
&0.1 \leq x_3 \leq 10,\\
&0.1 \leq x_3 \leq 2.
\end{aligned} \tag{38}$$

This is a well-studied problem that is solved by different researchers using different approaches. Regsdell and Phillips [27] solved it using mathematical-based methods. Deb [28], Coello [22], and Coello and Montes [23] solved it using GA-based algorithms. Also, He and Wang [24] solved it by CPSO, Montes and Coello [25] by Evolutionary strategies, and Kaveh and Talathari [26] by ACO. This problem is also solved by Kaveh and Talathari [14] utilizing the CSS algorithm. The results of the best solution found by each method are listed in Table 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 presented in Table 10. Similar to the CSS algorithm, MCSS has a small value for the standard deviation.

5.2.3 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. 9. The construction cost consists of the cost of materials, forming, and welding [29]. 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 the cylindrical section of the vessel $L(=x_4)$. T_s

Table 9 Optimum results for the design of a welded beam

Methods	Optimal design variables				
	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	f_{cost}
Regsdell and Phillips [27]					
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 [28]	0.248900	6.173000	8.178900	0.253300	2.433116
Coello [22]	0.248900	3.420500	8.997500	0.210000	1.748309
Coello and Montes [23]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang [24]	0.202369	3.544214	9.048210	0.205723	1.728024
Montes and Coello [25]	0.199742	3.612060	9.037500	0.206082	1.737300
Kaveh and Talathari [26]	0.205700	3.471131	9.036683	0.205731	1.724918
Kaveh and Talathari (CSS) [14]	0.205820	3.468109	9.038024	0.205723	1.724866
Present work	0.205730	3.470489	9.036624	0.205730	1.724855

Table 10 Statistical results of different methods for the design of a welded beam

Methods	Best	Mean	Worst	Standard deviation
Regsdell and Phillips [27]	2.3815	N/A	N/A	N/A
Deb [28]	2.433116	N/A	N/A	N/A
Coello [22]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [23]	1.728226	1.792654	1.993408	0.074713
He and Wang [24]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [25]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [26]	1.724918	1.729752	1.775961	0.009200
Kaveh and Talathari (CSS) [14]	1.724866	1.739654	1.759479	0.008064
Present work	1.724855	1.735374	1.750127	0.007571

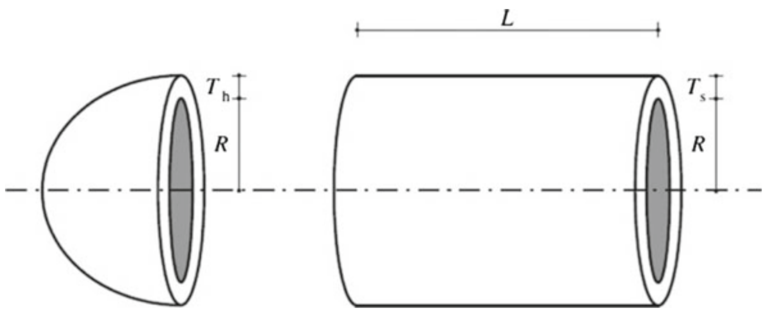

Fig. 9 Schematic of the pressure vessel, and its design variables

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:

$$f_{\cos t}(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2 + 19.84x_1^2x_3. \tag{39}$$

The constraints are as follows:

$$\begin{aligned} g_1(\mathbf{X}) &= -x_1 + 0.0193x_3 \leq 0, \\ g_2(\mathbf{X}) &= -x_2 + 0.00954x_3 \leq 0, \\ g_3(\mathbf{X}) &= -\pi \cdot x_3^2x_4 - \frac{4}{3}\pi \cdot x_3^3 + 1,296,000 \leq 0, \\ g_4(\mathbf{X}) &= x_4 - 240 \leq 0. \end{aligned} \tag{40}$$

Table 11 Optimum results for the design of a pressure vessel

Methods	Optimal design variables				
	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	f_{cost}
Sandgren [29]	1.125000	0.625000	47.700000	117.701000	8,129.1036
Kannan and Kramer [30]	1.125000	0.625000	58.291000	43.690000	7,198.0428
Deb and Gene [31]	0.937500	0.500000	48.329000	112.679000	6,410.3811
Coello [22]	0.812500	0.437500	40.323900	200.000000	6,288.7445
Coello and Montes [23]	0.812500	0.437500	42.097398	176.654050	6,059.9463
He and Wang [24]	0.812500	0.437500	42.091266	176.746500	6,061.0777
Montes and Coello [25]	0.812500	0.437500	42.098087	176.640518	6,059.7456
Kaveh and Talatahari [26]	0.812500	0.437500	42.098353	176.637751	6,059.7258
Kaveh and Talathari (CSS) [14]	0.812500	0.437500	42.103624	176.572656	6,059.0888
Present work	0.812500	0.437500	42.104550	176.560967	6,058.9710

Table 12 Statistical results of different methods for the design of a pressure vessel

Methods	Best	Mean	Worst	Standard deviation
Sandgren [29]	8,129.1036	N/A	N/A	N/A
Kannan and Kramer [30]	7,198.0428	N/A	N/A	N/A
Deb and Gene [31]	6,410.3811	N/A	N/A	N/A
Coello [22]	6,288.7445	6,293.8432	6,308.1497	7.4133
Coello and Montes [23]	6,059.9463	6,177.2533	6,469.3220	130.9297
He and Wang [24]	6,061.0777	6,147.1332	6,363.8041	86.4545
Montes and Coello [25]	6,059.7456	6,850.0049	7,332.8798	426.0000
Kaveh and Talatahari [26]	6,059.7258	6,081.7812	6,150.1289	67.2418
Kaveh and Talathari (CSS) [14]	6,059.0888	6,067.9062	6,085.4765	10.2564
Present work	6,058.9710	6,063.1798	6,074.7391	9.73494

The search space is defined as:

$$
\begin{aligned}
&0 \le x_1 \le 99,\\
&0 \le x_2 \le 99,\\
&10 \le x_3 \le 200,\\
&10 \le x_3 \le 200.
\end{aligned}
\tag{41}
$$

Various types of methods has been used for solving this problem. Some of these approaches are as: a branch and bound method [29], an augmented Lagrangian multiplier approach [30], genetic adaptive search [31], a GA-based algorithm [22], a feasibility-based tournament selection scheme [23], a co-evolutionary particle swarm method [24], an evolution strategy [25], an improved ACO [26], and the CSS algorithm [14]. The results of the best solution found by different methods are presented in Table 11. MCSS algorithm found a better solution compared to other techniques and the standard CSS. In Table 12, the results of statistical simulations are listed.

6 Concluding remarks

A new meta-heuristic algorithm called MCSS is developed to improve the performance of the CSS algorithm. In CSS, search agents that are called CPs attract each other by electrical forces, so the CPs moves in search space based on Newtonian mechanics laws. According to physics, when a charged particle moves, it produces a magnetic field, and this magnetic field can exert a magnetic force on other moving CPs. In MCSS, magnetic forces are considered in addition to the electrical forces to make the algorithm closer to the nature. The superposed force, that is, the combination of both electric and magnetic forces moves the CPs. The magnetic force provides more beneficiary information for the optimization process. It is assumed that each CP moves on a virtual wire. By each movement, it produces magnetic fields all around the search space, so it exerts an attractive or repelling force on other CPs. The concept of the magnetic field is based on the movement of CPs. If a CP experiences an increase in its objective function value, it will absorb other CPs.

According to the formulation of the MCSS, exploitation and exploration abilities of the standard CSS are improved. One of the most significant features of the MCSS algorithm is that it has a good performance even if with very small values for the number of CPs and maximum permitted iterations in comparison with the standard CSS. The difference between CSS and MCSS becomes more obvious when these values are set to small values. This feature can decrease the computational cost of the optimization.

Application of the MCSS on mathematical benchmarks and engineering problems illustrates the efficiency of the new proposed algorithm. Especially, the results of comparison between MCSS and CSS show that MCSS improves the search ability and convergence speed of the CSS algorithm.

Acknowledgments The first author is grateful to the Iran National Science Foundation for the support.

References

1. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
2. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Boston (1989)
3. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan (1995)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B **26**, 29–41 (1996)
5. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
6. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm; harmony search. Simulation **76**, 60–68 (2001)
7. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
8. Erol, O.K., Eksin, I.: New optimization method: Big Bang–Big Crunch. Adv. Eng. Softw. **37**, 106–111 (2006)
9. Kaveh, A., Talatahari, S.: Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput. Struct. **87**, 1129–1140 (2009)
10. Kaveh, A., Laknejadi, K., Alinejad, B.: Performance based multi-objective optimization of large steel structures. Acta. Mech. **223**, 355–369 (2012)
11. Kaveh, A., Talatahari, S.: Charged system search for optimal design of planar frame structures. Appl. Soft Comput. **12**, 382–393 (2012)
12. Kaveh, A., Talatahari, S.: Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput. Struct. **87**, 267–283 (2009)
13. Kaveh, A., Talatahari, S.: Optimization of large-scale truss structures using modified charged system search. Int. J. Optim. Civil Eng. **1**, 15–28 (2011)
14. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mech. **213**, 267–286 (2010)
15. Halliday, D., Resnick, R., Walker, J. : Fundamentals of Physics. Wiley, New York (2008)
16. Tsoulos, I.G.: Modifications of real code genetic algorithm for global optimization. Appl. Math. Comput. **203**, 598–607 (2008)
17. Hines, W., Montgomery, D.: Probability and Statistics in Engineering and Management Science. 3rd edn. Wiley, New York (1990)
18. Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S.: 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 (2005)
19. Garcia, S., Molina, D., Lozano, M., Herrera, F.: 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 (2009)
20. Belegundu, A.D.: A Study of Mathematical Programming Methods for Structural Optimization. Ph.D. Thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa (1982)
21. Arora, J.S.: Introduction to Optimum Design. McGraw-Hill, New York (1989)
22. Coello, C.A.C.: Use of a self-adaptive penalty approach for engineering optimization problems. Comput. Ind. **41**, 113–127 (2000)
23. Coello, C.A.C., Montes, E.M.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv. Eng. Inform. **16**, 193–203 (2002)
24. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. **20**, 89–99 (2007)
25. Montes, E.M., Coello, C.A.C.: An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int. J. Gen. Syst. **37**, 443–473 (2008)
26. Kaveh, A., Talatahari, S.: An improved ant colony optimization for constrained engineering design problems. Eng. Comput. **27**, 155–182 (2010)
27. Ragsdell, K.M., Phillips, D.T.: Optimal design of a class of welded structures using geometric programming. ASME J. Eng. Ind. Ser. B **98**, 1021–1025 (1976)
28. Deb, K.: Optimal design of a welded beam via genetic algorithms. AIAA J. **29**, 2013–2015 (1991)
29. Sandgren, E.: Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME Design Technology Conference, pp. 95–105. Kissimine (1988)
30. Kannan, B.K., Kramer, S.N.: 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 (1994)
31. Deb, K., Gene, A.S.: A robust optimal design technique for mechanical component design. In: Dasgupta, D., Michalewicz, Z. (eds.) Evolutionary Algorithms in Engineering Applications., pp. 497–514. Springer, Berlin (1997)