Acta Mech 213, 267–289 (2010)
DOI 10.1007/s00707-009-0270-4

A. Kaveh · S. Talatahari

A novel heuristic optimization method: charged system search

Received: 28 July 2009 / Revised: 15 November 2009 / Published online: 22 January 2010
© Springer-Verlag 2010

Abstract This paper presents a new optimization algorithm based on some principles from physics and mechanics, which will be called Charged System Search (CSS). We utilize the governing Coulomb law from electrostatics and the Newtonian laws of mechanics. CSS is a multi-agent approach in which each agent is a Charged Particle (CP). CPs can affect each other based on their fitness values and their separation distances. The quantity of the resultant force is determined by using the electrostatics laws and the quality of the movement is determined using Newtonian mechanics laws. CSS can be utilized in all optimization fields; especially it is suitable for non-smooth or non-convex domains. CSS needs neither the gradient information nor the continuity of the search space. The efficiency of the new approach is demonstrated using standard benchmark functions and some well-studied engineering design problems. A comparison of the results with those of other evolutionary algorithms shows that the proposed algorithm outperforms its rivals.

1 Introduction

There are two general methods to optimize a function, namely, mathematical programming and meta-heuristic methods. Various mathematical programming methods such as linear programming, homogenous linear programming, integer programming, dynamic programming, and nonlinear programming have been applied for solving optimization problems. These methods use gradient information to search the solution space near an initial starting point. In general, gradient-based methods converge faster and can obtain solutions with higher accuracy compared to stochastic approaches in fulfilling the local search task. However, for effective implementation of these methods, the variables and cost function of the generators need to be continuous. Furthermore, a good starting point is vital for these methods to be executed successfully. In many optimization problems, prohibited zones, side limits, and non-smooth or non-convex cost functions need to be considered. As a result, these non-convex optimization problems cannot be solved by the traditional mathematical programming methods. Although dynamic programming or mixed integer nonlinear programming and their modifications offer some facility in solving non-convex problems, these methods, in general, require considerable computational effort.

As an alternative to the conventional mathematical approaches, the meta-heuristic optimization techniques have been used to obtain global or near-global optimum solutions. Due to their capability of exploring and

A. Kaveh (✉)
Department of Civil Engineering, 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

S. Talatahari
Department of Civil Engineering, University of Tabriz, Tabriz, Iran
E-mail: siamak.talat@gmail.com

finding promising regions in the search space in an affordable time, these methods are quite suitable for global searches and furthermore alleviate the need for continuous cost functions and variables used for mathematical optimization methods. Though these are approximate methods, i.e., their solution are good, but not necessarily optimal, they do not require the derivatives of the objective function and constraints and employ probabilistic transition rules instead of deterministic ones [1].

Nature has always been a major source of inspiration to engineers and natural philosophers and many meta-heuristic approaches are inspired by solutions that nature herself seems to have chosen for hard problems. The Evolutionary Algorithm (EA) proposed by Fogel et al. [2], De Jong [3] and Koza [4], and the Genetic Algorithm (GA) proposed by Holland [5] and Goldberg [6] are inspired from the biological evolutionary process. Studies on animal behavior led to the method of Tabu Search (TS) presented by Glover [7], Ant Colony Optimization (ACO) proposed by Dorigo et al. [8] and Particle Swarm Optimizer (PSO) formulated by Eberhart and Kennedy [9]. Also, Simulated Annealing proposed by Kirkpatrick et al. [10], the Big Bang–Big Crunch algorithm (BB–BC) proposed by Erol and Eksin [11] and improved by Kaveh and Talatahari [12], and the Gravitational Search Algorithm (GSA) presented by Rashedi et al. [13] are introduced using physical phenomena.

The objective of this paper is to present a new optimization algorithm based on principles from physics and mechanics, which will be called Charged System Search (CSS). We utilize the governing Coulomb law from physics and the governing motion from Newtonian mechanics. The remainder of this paper is organized as follows. Section 2 presents the basic aspects and the characteristics of the CSS. Numerical examples are presented in Sect. 3 to verify the efficiency of the new algorithm, and some concluding remarks are provided in Sect. 4.

2 Charged system search

2.1 Background

2.1.1 Electrical laws

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

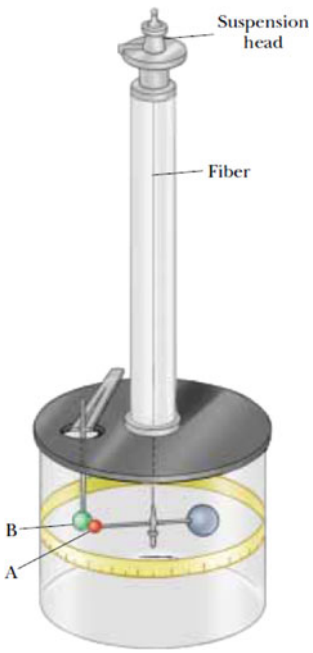

Fig. 1 Coulomb's torsion balance, used to establish the inverse-square law for the electric force between two charges

their separation distance. The electric force between charged spheres A and B in Fig. 1 causes the spheres to either attract or repel each other, and the resulting motion causes the suspended fiber to twist. Since the restoring torque of the twisted fiber is proportional to the angle through which the fiber rotates, a measurement of this angle provides a quantitative measure of the electric force of attraction or repulsion [14]. Coulomb's experiments showed that the electric force between two stationary charged particles

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

From the above observations, Coulomb's law provides the magnitude of the electric force (Coulomb force) between the two point-charges [14] as

$$F_{ij} = k_e \frac{q_i q_j}{r_{ij}^2}, \tag{1}$$

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

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

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

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

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

Note that this result shows that $E_{ij} \to 0$ as $r_{ij} \to 0$. Therefore, the result eliminates the problem that would exist at $r_{ij} = 0$ if E_{ij} is varied as $1/r_{ij}^2$ inside the sphere as it does outside the sphere. That is, if $E_{ij} \propto 1/r_{ij}^2$ the field will be infinite at $r_{ij} = 0$, which is physically impossible. Hence, the electric field inside the sphere varies linearly with r_{ij}. The field outside the sphere is the same as that of a point charge q_i located at $r_{ij} = 0$. Also the magnitudes of the electric fields for a point at inside or outside the sphere coincide when $r_{ij} = a$. A plot of E_{ij} versus r_{ij} is shown in Fig. 2, Ref. [14].

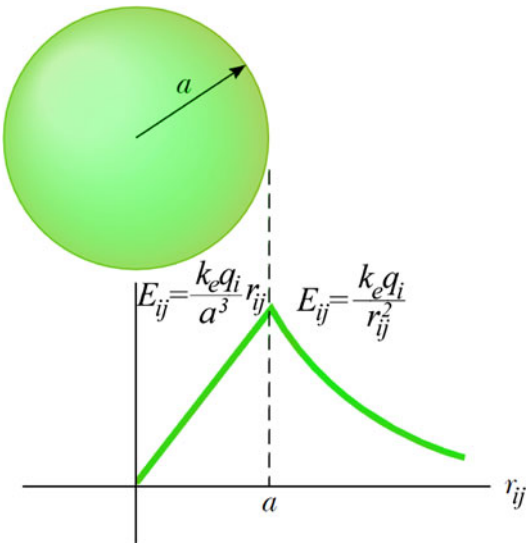

Fig. 2 A plot of E_{ij} versus r_{ij} for a uniformly charged insulating sphere

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

$$E_j = \sum_{i=1,\, i\neq j}^{N} E_{ij}, \tag{4}$$

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

$$E_{ij} = \begin{cases} \frac{k_e q_i}{a^3} r_{ij} & \text{if } r_{ij} < a, \\ \frac{k_e q_i}{r_{ij}^2} & \text{if } r_{ij} \geq a. \end{cases} \tag{5}$$

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

$$\mathbf{F}_{ij} = E_{ij} q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|}. \tag{6}$$

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

$$\mathbf{F}_j = k_e q_j \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \quad \begin{cases} i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a, \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a. \end{cases} \tag{7}$$

2.1.2 Newtonian mechanics laws

Newtonian mechanics or classical mechanics studies the motion of objects. In the study of motion, the moving object is described as a particle regardless of its size. In general, a particle is a point-like mass having infinitesimal size. The motion of a particle is completely known if the particle's position in space is known at all times. The displacement of a particle is defined as its change in position. As it moves from an initial position $\mathbf{r}_{\text{old}}$ to a final position $\mathbf{r}_{\text{new}}$, its displacement is given by

$$\Delta\mathbf{r} = \mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}}. \tag{8}$$

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

$$\mathbf{v} = \frac{\mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}}}{t_{\text{new}} - t_{\text{old}}} = \frac{\mathbf{r}_{\text{new}} - \mathbf{r}_{\text{old}}}{\Delta t}. \tag{9}$$

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

$$\mathbf{a} = \frac{\mathbf{v}_{\text{new}} - \mathbf{v}_{\text{old}}}{\Delta t}. \tag{10}$$

Using Eqs. (8)–(10), the displacement of any object as a function of time is obtained approximately as

$$\mathbf{r}_{\text{new}} = \frac{1}{2}\mathbf{a} \cdot \Delta t^2 + \mathbf{v}_{\text{old}} \cdot \Delta t + \mathbf{r}_{\text{old}}. \tag{11}$$

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

$$\mathbf{F} = m \cdot \mathbf{a}, \tag{12}$$

where m is the mass of the object.

Substituting Eq. (12) into Eq. (11), we have

$$\mathbf{r}_{\text{new}} = \frac{1}{2}\frac{\mathbf{F}}{m} \cdot \Delta t^2 + \mathbf{v}_{\text{old}} \cdot \Delta t + \mathbf{r}_{\text{old}}. \tag{13}$$

2.2 Presentation of charged search system

In this section, a new efficient optimization algorithm is established utilizing the aforementioned physics laws, which is called Charged System Search (CSS). In the CSS, each solution candidate $\mathbf{X}_i$ containing a number of decision variables (i.e. $\mathbf{X}_i = \{x_{i,j}\}$) is considered as a charged particle. The charged particle is affected by the electrical fields of the other agents. The quantity of the resultant force is determined by using the electrostatics laws as discussed in Sect. 2.1.1, and the quality of the movement is determined using the Newtonian mechanics laws. It seems that an agent with good results must exert a stronger force than the bad ones, so the amount of the charge will be defined considering the objective function value, fit(i). In order to introduce CSS, the following rules are developed:

Rule 1 Many of the natural evolution algorithms maintain a population of solutions which are evolved through random alterations and selection [15]–[16]. Similarly, CSS considers a number of Charged Particles (CP). Each CP has a magnitude of charge (q_i) and as a result creates an electrical field around its space. The magnitude of the charge is defined considering the quality of its solution, as follows:

$$q_i = \frac{\text{fit}(i) - \text{fitworst}}{\text{fitbest} - \text{fitworst}}, \quad i = 1, 2, \ldots, N, \tag{14}$$

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

$$r_{ij} = \frac{\|\mathbf{X}_i - \mathbf{X}_j\|}{\|(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best}\| + \varepsilon}, \tag{15}$$

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

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

$$x_{i,j}^{(\text{o})} = x_{i,\min} + \text{rand} \cdot (x_{i,\max} - x_{i,\min}), \quad i = 1, 2, \ldots, n, \tag{16}$$

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

$$v_{i,j}^{(\text{o})} = 0, \quad i = 1, 2, \ldots, n. \tag{17}$$

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

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

$$p_{ij} = \begin{cases} 1 & \text{fit}(j) > \text{fit}(i), \\ 0 & \text{else.} \end{cases} \tag{18}$$

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

$$p_{ij} = \begin{cases} 1 & \frac{\text{fit}(i) - \text{fitbest}}{\text{fit}(j) - \text{fit}(i)} > \text{rand} \vee \text{fit}(j) > \text{fit}(i), \\ 0 & \text{else.} \end{cases} \tag{19}$$

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

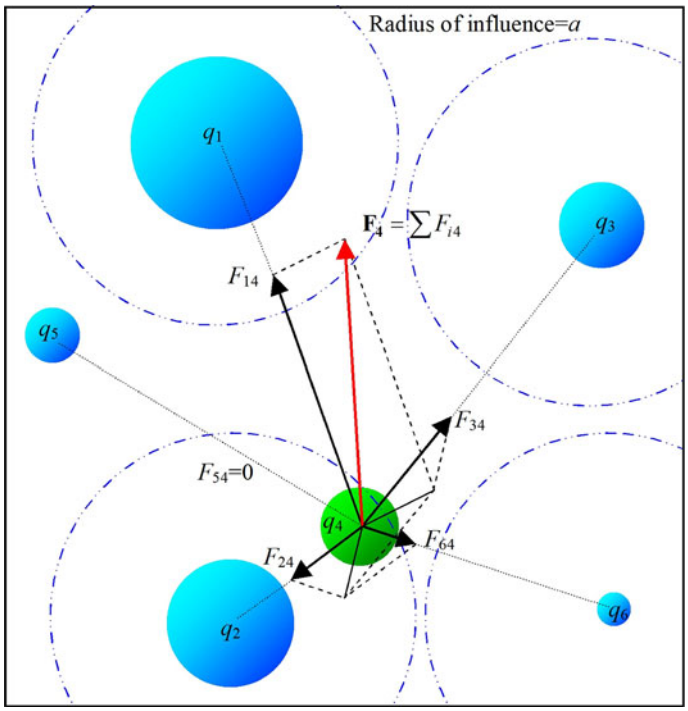

Fig. 3 Determining the resultant electrical force acting on a CP

Rule 4 The value of the resultant electrical force acting on a CP is determined using Eq. (7) as

$$\mathbf{F}_j = q_j \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij}(\mathbf{X}_i - \mathbf{X}_j), \quad \begin{cases} j = 1, 2, \ldots, N, \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a, \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a, \end{cases} \tag{20}$$

where $\mathbf{F}_j$ is the resultant force acting on the jth CP, as illustrated in Fig. 3.

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

$$a = 0.10 \times \max\left(\left\{ x_{i,\max} - x_{i,\min} \right| \quad i = 1, 2, \ldots, n \right\} \right). \tag{21}$$

According to this rule, in the first iteration where the agents are far from each other the magnitude of the resultant force acting on a CP is inversely proportional to the square of the separation between the particles. Thus the exploration power in this condition is high because of performing more searches in the early iterations. It is necessary to increase the exploitation of the algorithm and to decrease the exploration gradually. After a number of searches where CPs are collected in a small space and the separation between the CPs becomes small, say 0.1, then the resultant force becomes proportional to the separation distance of the particles instead of being inversely proportional to the square of the separation distance. According to Fig. 4, if the first equation ($F_{ij} \propto 1/r_{ij}^2$) is used for $r_{ij} = 0.1$, we have $F_{ij} = 100 \times k_e q_i q_j$ that is a large value, compared to a force acting on a CP at $r_{ij} = 2 (F_{ij} = 0.25 \times k_e q_i q_j)$, and this great force causes particles to get farther from each other instead of getting nearer, while the second one ($F_{ij} \propto r_{ij}$) guarantees that a convergence will happen. Therefore, the parameter a separates the global search phase and the local search phase, i.e., when majority of the agents are collected in a space with radius a, the global search is finished and the optimizing process is continued by improving the previous results, and thus the local search starts. Besides, using these principles controls the balance between the exploration and the exploitation.

It should be noted that this rule considers the competition step of the algorithm. Since the resultant force is proportional to the magnitude of the charge, a better fitness (great q_i) can create a stronger attracting force, so the tendency to move toward a good CP becomes more than toward a bad particle.

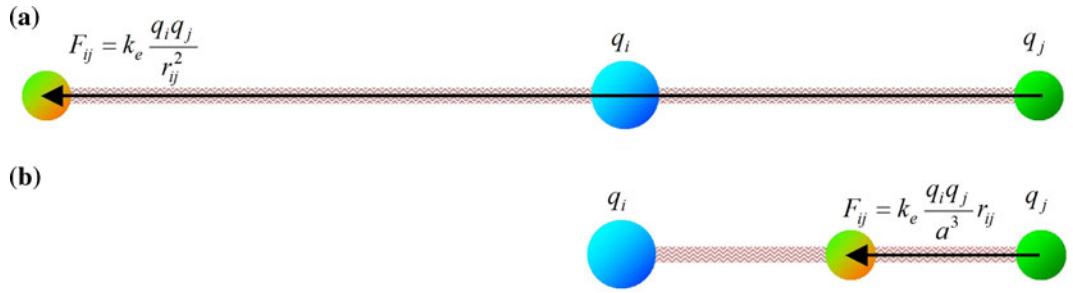

Fig. 4 A comperation between the equation (**a**) $F_{ij} \propto 1/r_{ij}^2$ and (**b**) $F_{ij} \propto r_{ij}$ when $r_{ij} < a$

Rule 5 The new position and velocity of each CP is determined considering Eqs. (9) and (13), as follows:

$$\mathbf{X}_{j,\text{new}} = \text{rand}_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + \text{rand}_{j2} \cdot k_v \cdot \mathbf{V}_{j,\text{old}} \cdot \Delta t + \mathbf{X}_{j,\text{old}}, \tag{22}$$

$$\mathbf{V}_{j,\text{new}} = \frac{\mathbf{X}_{j,\text{new}} - \mathbf{X}_{j,\text{old}}}{\Delta t}, \tag{23}$$

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

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

$$k_v = 0.5(1 - \text{iter}/\text{iter}_{\max}), \quad k_a = 0.5(1 + \text{iter}/\text{iter}_{\max}), \tag{24}$$

where iter is the actual iteration number and $\text{iter}_{\max}$ is the maximum number of iterations. With this equation, k_v decreases linearly to zero while k_a increases to one when the number of iterations rises. In this way, the balance between the exploration and the fast rate of convergence is saved. Considering the values of these parameters, Eqs. (22) and (23) can be rewritten as

$$\begin{aligned}\mathbf{X}_{j,\text{new}} = {} & 0.5\text{rand}_{j1} \cdot (1 + \text{iter}/\text{iter}_{\max}) \cdot \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij} (\mathbf{X}_i - \mathbf{X}_j) \\ & + 0.5\text{rand}_{j2} \cdot (1 + \text{iter}/\text{iter}_{\max}) \cdot \mathbf{V}_{j,\text{old}} + \mathbf{X}_{j,\text{old}}, \end{aligned} \tag{25}$$

$$\mathbf{V}_{j,\text{new}} = \mathbf{X}_{j,\text{new}} - \mathbf{X}_{j,\text{old}}. \tag{26}$$

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

Rule 6 Considering a memory which saves the best CP vectors and their related objective function values can improve the algorithm performance without increasing the computational cost. To fulfill this aim, Charged Memory (CM) is utilized to save a number of the best so far solutions. In this paper, the size of the CM (i.e. *CMS*) is taken as $N/4$. Another benefit of the CM consists of utilizing this memory to guide the current CPs. In other words, the vectors stored in the CM can attract current CPs according to Eq. (20). Instead, it is assumed that the same number of the current worst particles can not attract the others.

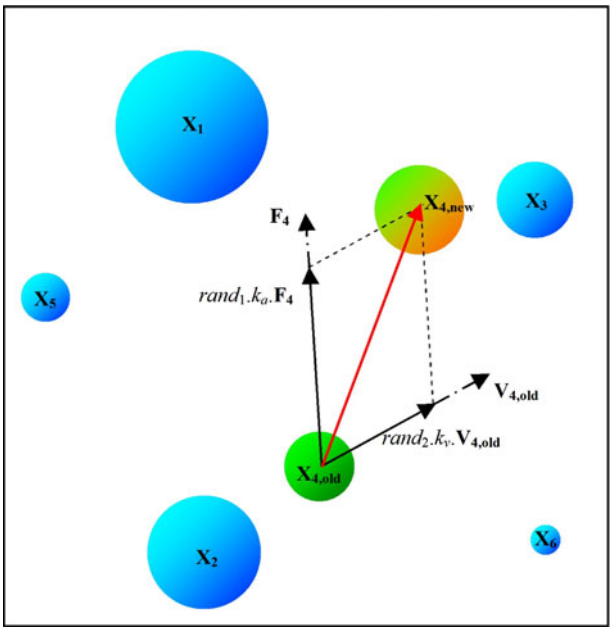

Fig. 5 The movement of a CP to the new position

Rule 7 There are two major problems in relation to many meta-heuristic algorithms; the first problem is the balance between exploration and exploitation in the beginning, during, and at the end of the search, and the second is how to deal with an agent violating the limits of the variables.

The first problem is solved naturally through the application of above-stated rules; however, in order to solve the second problem, one of the simplest approaches is utilizing the nearest limit values for the violated variable. Alternatively, one can force the violating particle to return to its previous position, or one can reduce the maximum value of the velocity to allow fewer particles to violate the variable boundaries. Although these approaches are simple, they are not sufficiently efficient and may lead to reduce the exploration of the search space. This problem has previously been addressed and solved using the harmony search-based handling approach [15,17]. According to this mechanism, any component of the solution vector violating the variable boundaries can be regenerated from the CM as

$$x_{i,j} = \begin{cases} \text{w.p. CMCR} & \Longrightarrow \text{select a new value for a variable from CM,} \\ & \Longrightarrow \text{w.p.}(1-\text{PAR})\text{do nothing,} \\ & \Longrightarrow \text{w.p. PAR choose a neighboring value,} \\ \text{w.p.}(1-\text{CMCR}) & \Longrightarrow \text{select a new value randomly,} \end{cases} \tag{27}$$

where "w.p." is the abbreviation for "with the probability"; $x_{i,j}$ is the ith component of the CP j; The CMCR (the Charged Memory Considering Rate) varying between 0 and 1 sets the rate of choosing a value in the new vector from the historic values stored in the CM, and $(1-\text{CMCR})$ sets the rate of randomly choosing one value from the possible range of values. The pitch adjusting process is performed only after a value is chosen from CM. The value $(1-\text{PAR})$ sets the rate of doing nothing, and PAR sets the rate of choosing a value from neighboring the best CP. For more details, the reader may refer to Refs. [15,17].

Rule 8 The terminating criterion is one of the following:

- Maximum number of iterations: the optimization process is terminated after a fixed number of iterations, for example, 1,000 iterations.
- Number of iterations without improvement: the optimization process is terminated after some fixed number of iterations without any improvement.
- Minimum objective function error: the difference between the values of the best objective function and the global optimum is less than a pre-fixed anticipated threshold.
- Difference between the best and the worst CPs: the optimization process is stopped if the difference between the objective values of the best and the worst CPs becomes less than a specified accuracy.

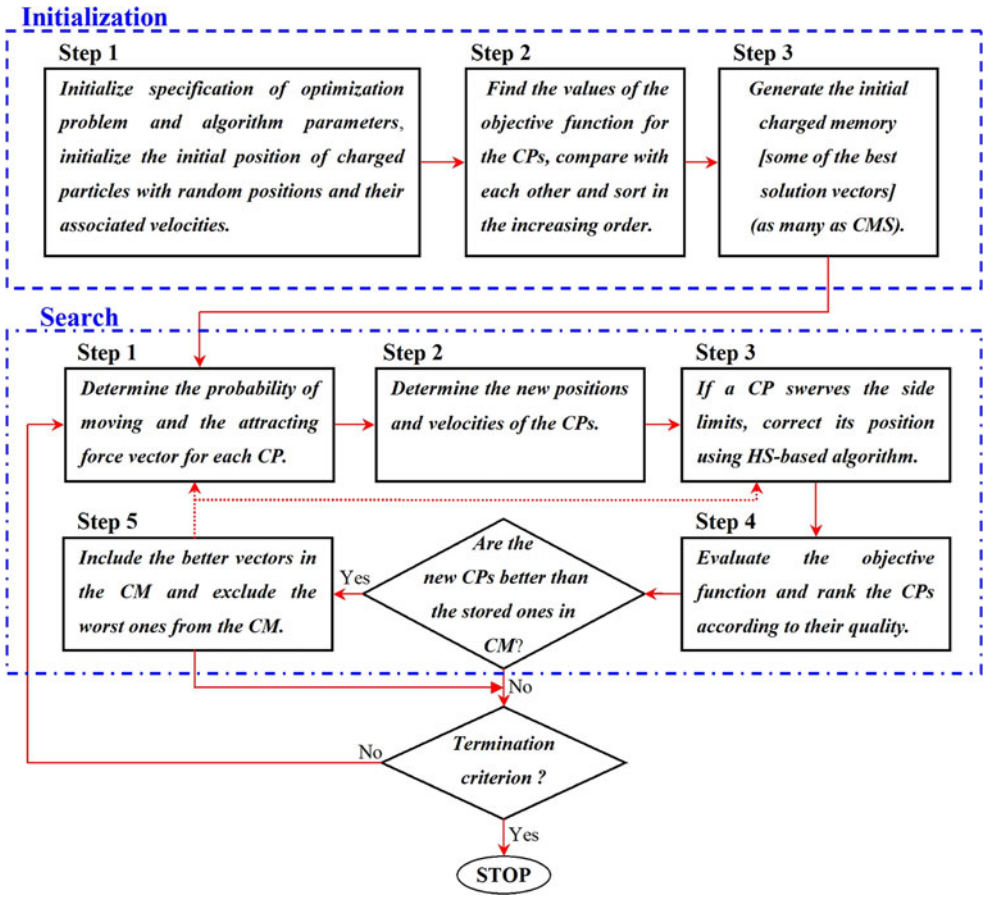

Fig. 6 The flowchart of the CSS

- Maximum distance of CPs: the maximum distance between CPs is less than a pre-fixed value.

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

Level 1: Initialization

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

Level 2: Search

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

Level 3: Terminating criterion controlling

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

The flowchart of the CSS algorithm is illustrated in Fig. 6.

3 Validation of CSS

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

3.1 Description of the examples

In this section a number of benchmark functions chosen from Ref. [18] are optimized using CSS and compared to GA and some of its variations to verify the efficiency of CSS. The description of these test problems is provided in Table 1. When the dimension is selected as 2, a perspective view and the related contour lines for some of these functions are illustrated in Fig. 7.

3.2 Results

Similar to the other meta-heuristics, for the CSS a large value for the number of CPs increases the search strength of the algorithm as well as the computational cost and vice versa a small number causes a quick convergence without performing a complete search. Here, the number of CPs is set to 20 and the maximum number of the permitted iterations is considered as 200. These values seem to be suitable for finding the optimum results. The value of HMCR is set to 0.95 and that of PAR is taken as 0.10 [15]. The results obtained by CSS are listed in Table 2 along with those obtained by GA and some of its variations, which are directly derived from [18]. The numbers denote the average number of function evaluations from 50 independent runs for every objective function described in Sect. 3.1. The numbers in parentheses represent the fraction of successful runs in which the algorithm has located the global minimum with predefined accuracy, which is taken as $\varepsilon = f_{\min} - f_{\text{final}} = 10^{-4}$. The absence of the parentheses denotes that the algorithm has been successful in all independent runs. Although the GEN-S-M-LS finds good results in some cases, it must be noted that GEN-S-M-LS utilizes some auxiliary mechanisms such as an improved stopping rule, a new mutation mechanism, and a repeated application of a local search procedure. To sum up, comparison of the results demonstrates that CSS has a faster convergence than original GA and its variations.

In order to have some general idea about the way the CSS works, Fig. 8 is prepared to show the positions of the current CPs and the stored CPs in the CM for the first example. It can be seen that in the first iterations, the CPs investigate the entire search space to discover a favorite space (global search). When this favorite space containing a global optimum is discovered, the movements of the CPs are limited to this space in order to provide more exploitation (local search).

For many heuristic algorithms it is a common feature that if all the agents get gathered in a small space, i.e., if the agents are trapped in part of the search space, escaping from this may be very difficult. Since prevailing forces for the CSS algorithm are attracting forces, it looks as if the above problem has remained unsolved for this method. However, having a good balance between the exploration and the exploitations, and considering three steps containing self-adaptation, cooperation and competition in the CSS, can solve this problem. As illustrated in Fig. 9 which shows the positions of the CPs for the first example when all the initial agents are located in a small part of the space, CSS can escape from this space and go toward the favorite space.

3.3 Engineering design problems

Three engineering design problems which have been previously solved using a variety of other techniques are considered to show the validity and effectiveness of the proposed algorithm. To handle the constraints, a simple

Table 1 Specifications of the benchmark problems

Function name	Interval	Function	Global minimum
Aluffi-Pentiny	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in [-50, 50]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	0.0
Branin	$0 \le x_2 \le 15 - 5 \le x_1 \le 10$	$f(\mathbf{X}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$	0.397887
Camel	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.0316
Cb3	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^5 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$n = 4, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in [-5.12, 5.12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$n = 2, 4, 8, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	−1
Goldstein and price	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2][30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 - 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2]$	3.0
Griewank	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\cos(\frac{x_i}{\sqrt{i}})$	0.0
Hartman 3	$\mathbf{X} \in [0, 1]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and $p = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$.	−3.862782
Hartman 6	$\mathbf{X} \in [0, 1]^6$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	−3.322368
Rastrigin	$\mathbf{X} \in [-1, 1]^2$	$f(\mathbf{X}) = \sum_{i=1}^{2}\left(x_i^2 - \cos(18x_i)\right)$	−2.0
Rosenbrock	$\mathbf{X} \in [-30, 30]^n,\ n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0.0

penalty approach is employed. In utilizing the penalty functions, if the constraints are between the allowable limits, the penalty will be zero; otherwise, the amount of penalty is equal to the normalized violation [12]. Though there are some better methods which can easily be implemented, however, here utilizing such a handling approach one may think that the improvement of results is due to the use of the good handling approach rather than the application of the CSS. Therefore, here, the penalty approach is utilized. Since the CSS is independent of the type of the penalty functions, one can easily use another approach in his/her application of CSS.

3.3.1 A tension/compression spring design problem

This problem is described by Belegundu [19] and Arora [20]. It consists of minimizing the weight of a tension/compression spring subject to constraints on shear stress, surge frequency, and minimum deflection as shown in Fig. 10.

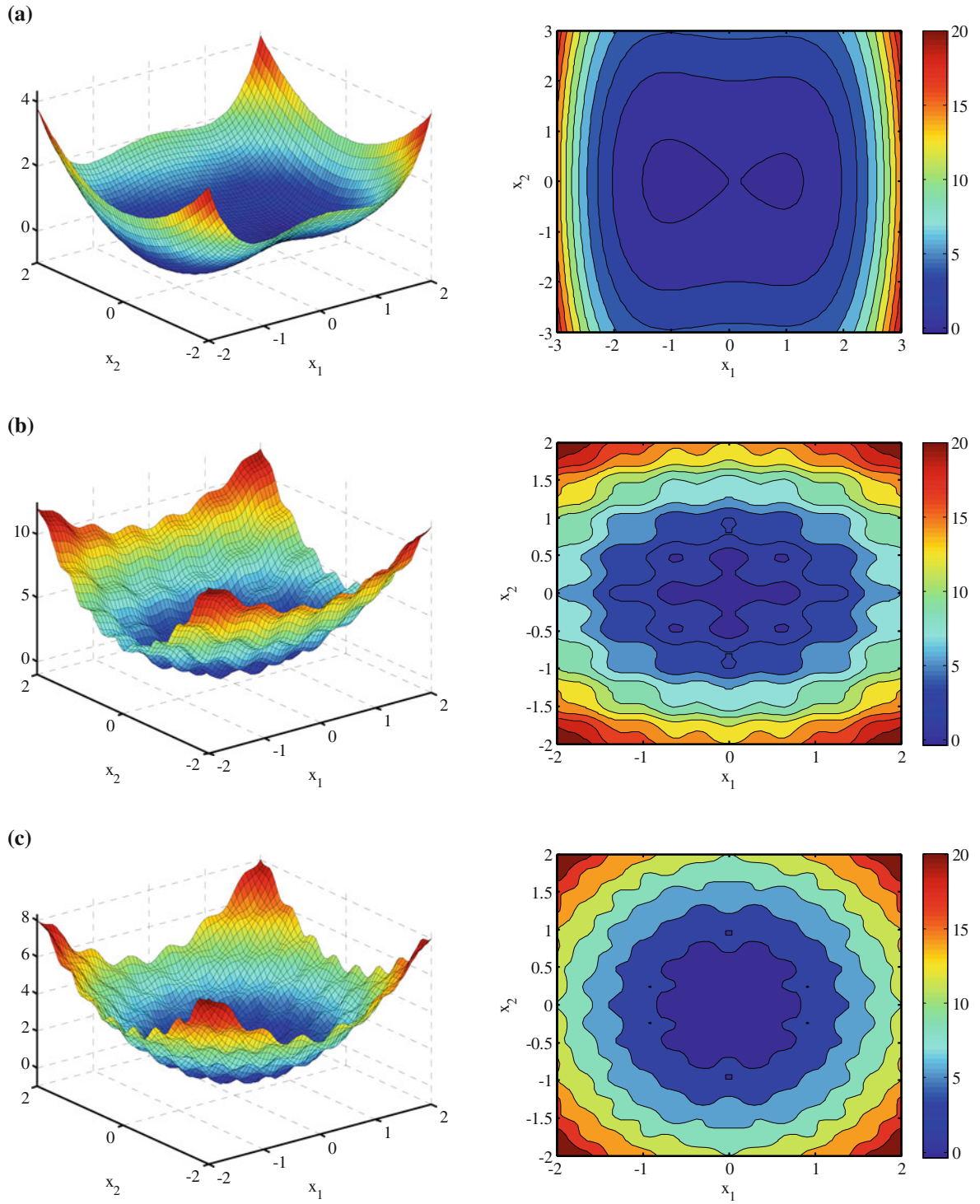

Fig. 7 A perspective view and the related contour lines for some of function when $n = 2$. **a** Aluffi-Pentiny, **b** Bohachevsky 1, **c** Bohachevsky 2, **d** Becker and Lago, **e** Branin, **f** Camel, **g** Cb3, **h** Cosine mixture, **i** Exponential, **j** Griewank, **k** Rastrigin, **l** Rosenbrock

The design variables are the mean coil diameter $D(= x_1)$; the wire diameter $d(= x_2)$, and the number of active coils $N(= x_3)$. The problem can be stated with the cost function

$$f_{\text{cost}}(\mathbf{X}) = (x_3 + 2)x_2x_1^2, \tag{28}$$

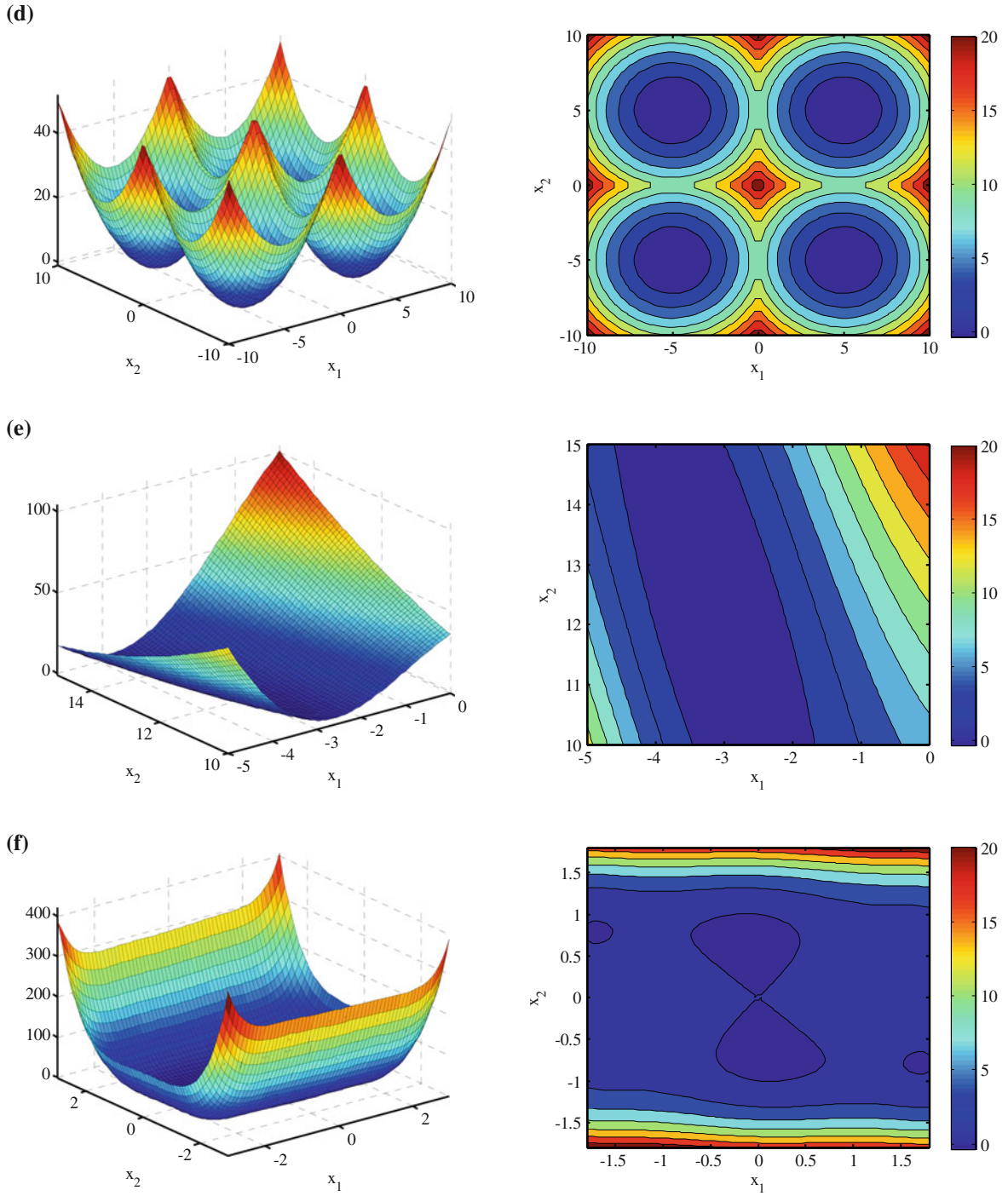

Fig. 7 continued

to be minimized and constraints

$$g_1(\mathbf{X}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0,$$

$$g_2(\mathbf{X}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0,$$

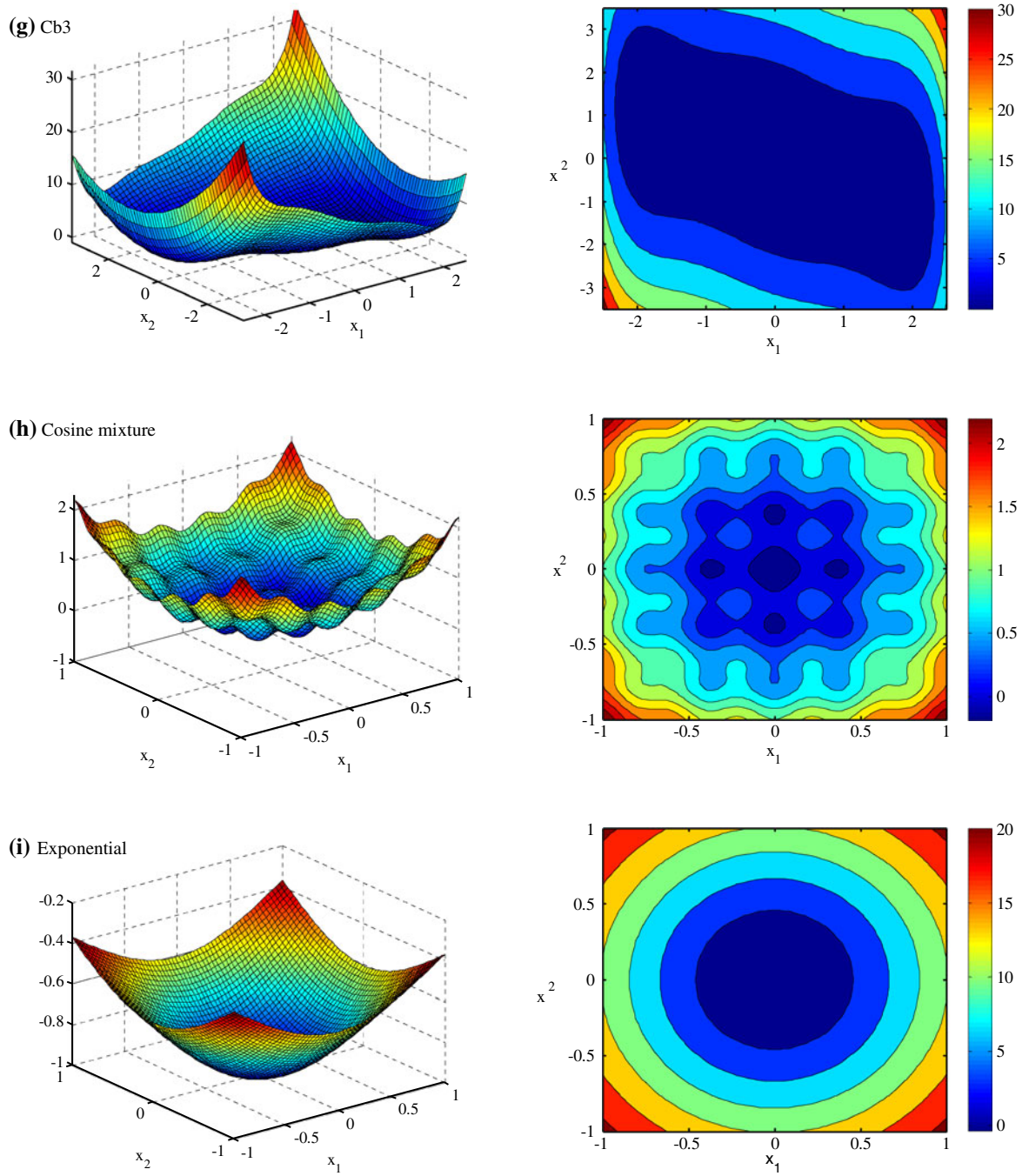

Fig. 7 continued

$$g_3(\mathbf{X}) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0,$$

$$g_4(\mathbf{X}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0. \tag{29}$$

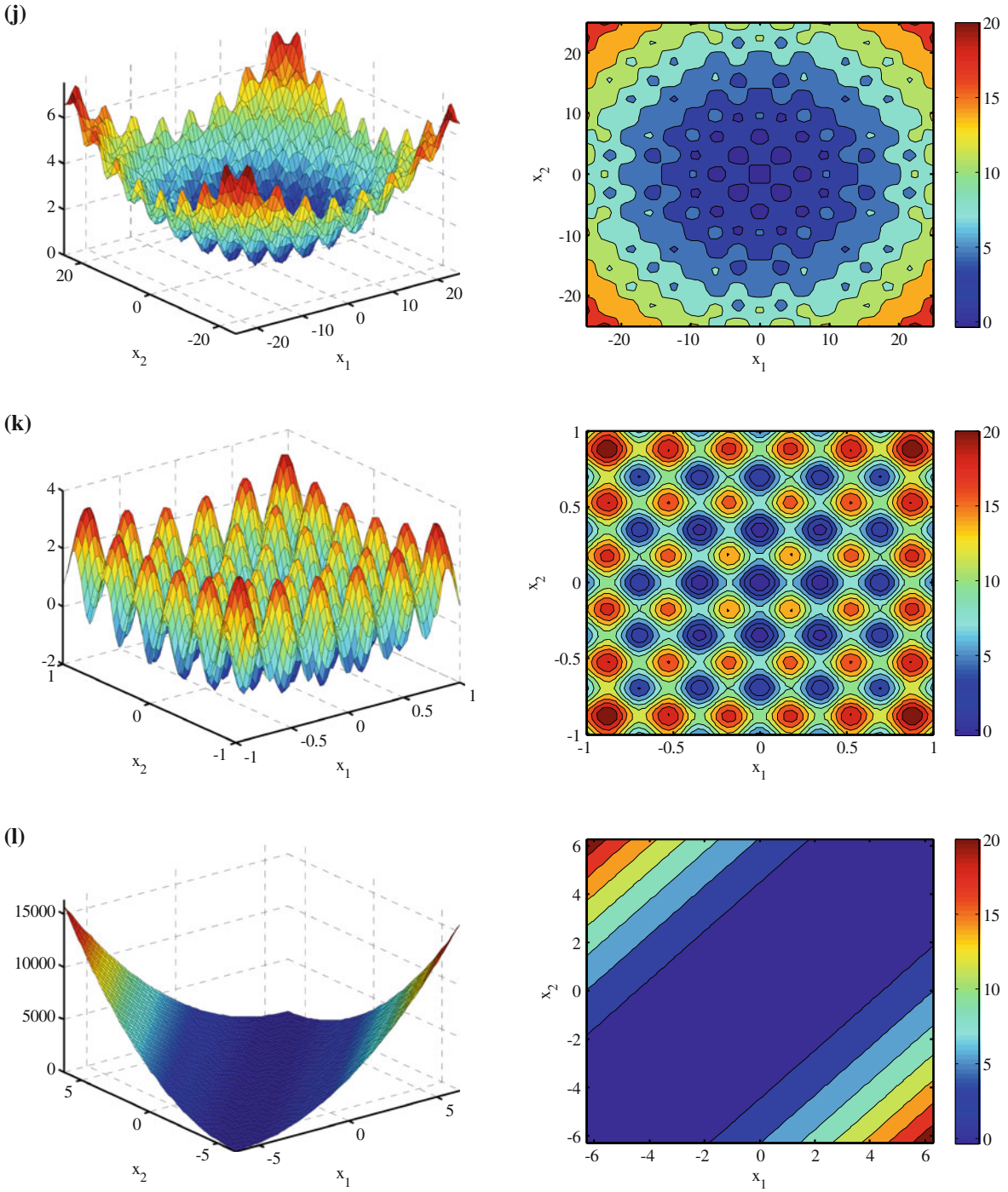

Fig. 7 continued

The design space is bounded by

$$\begin{aligned} 0.05 &\le x_1 \le 2, \\ 0.25 &\le x_2 \le 1.3, \\ 2 &\le x_3 \le 15. \end{aligned} \tag{30}$$

Table 2 Performance comparison for the benchmark problems

Function	GEN	GEN–S	GEN–S–M	GEN–S–M–LS	*CSS*
AP	1,360 (0.99)	1,360	1,277	1,253	804
Bf1	3,992	3,356	1,640	1,615	1,187
Bf2	20,234	3,373	1,676	1,636	742
BL	19,596	2,412	2,439	1,436	423
Branin	1,442	1,418	1,404	1,257	852
Camel	1,358	1,358	1,336	1,300	575
Cb3	9,771	2,045	1,163	1,118	436
CM	2,105	2,105	1,743	1,539	1,563
Dejoung	9,900	3,040	1,462	1,281	630
Exp2	938	936	817	807	132
Exp4	3,237	3,237	2,054	1,496	867
Exp8	3,237	3,237	2,054	1,496	1,426
Goldstein and Price	1,478	1,478	1,408	1,325	682
Griewank	18,838 (0.91)	3,111 (0.91)	1,764	1,652 (0.99)	1,551
Hartman3	1,350	1,350	1,332	1,274	860
Hartman6	2,562 (0.54)	2,562 (0.54)	2,530 (0.67)	1,865 (0.68)	1,783
Rastrigin	1,533 (0.97)	1,523 (0.97)	1,392	1,381	1,402
Rosenbrock	9,380	3,739	1,675	1,462	1,452
Total	112,311 (96.72)	41,640 (96.77)	29,166 (98.16)	25,193 (98.16)	17,367

This problem has been solved by Belegundu [19] using eight different mathematical optimization techniques (only the best results are shown). Arora [20] has also solved this problem using a numerical optimization technique called constraint correction at the constant cost. Coello [21] as well as Coello and Montes [22] solved this problem using a GA-based method. Additionally, He and Wang [23] utilized a co-evolutionary particle swarm optimization (CPSO). Recently, Montes and Coello [24] and the authors [1] used evolution strategies and an improved ant colony optimization to solve this problem, respectively. Table 3 presents the best solution of this problem obtained using the CSS algorithm and compares the CSS results with solutions reported by other researchers. From Table 3, it can be seen that the best feasible solution obtained by CSS is better than those previously reported. Table 4 shows the statistical simulation results.

3.3.2 A welded beam design problem

The welded beam structure, shown in Fig. 11, is a practical design problem that has been often used as a benchmark problem for testing different optimization methods [1], [21–26]. The aim is to find the minimum fabricating cost of the welded beam subjected to constraints on shear stress (τ), bending stress (σ), buckling load (P_c), end deflection (δ), and side constraint. There are four design variables, namely $h(= x_1)$, $l(= x_2)$, $t(= x_3)$, and $b(= x_4)$.

The mathematical formulation of the cost function $f_{\text{cost}}(\mathbf{X})$, which is the total fabricating cost, mainly comprises of the set-up, welding labor, and material costs, as

$$f_{\text{cost}}(\mathbf{X}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2). \tag{31}$$

Constraints are

$$\begin{aligned}
&g_1(\mathbf{X}) = \tau(\{x\}) - \tau_{\max} \le 0,\\
&g_2(\mathbf{X}) = \sigma(\{x\}) - \sigma_{\max} \le 0,\\
&g_3(\mathbf{X}) = x_1 - x_4 \le 0,\\
&g_4(\mathbf{X}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,\\
&g_5(\mathbf{X}) = 0.125 - x_1 \le 0,\\
&g_6(\mathbf{X}) = \delta(\{x\}) - \delta_{\max} \le 0,\\
&g_7(\mathbf{X}) = P - P_c(\{x\}) \le 0,
\end{aligned} \tag{32}$$

where

$$\tau(\mathbf{X}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$

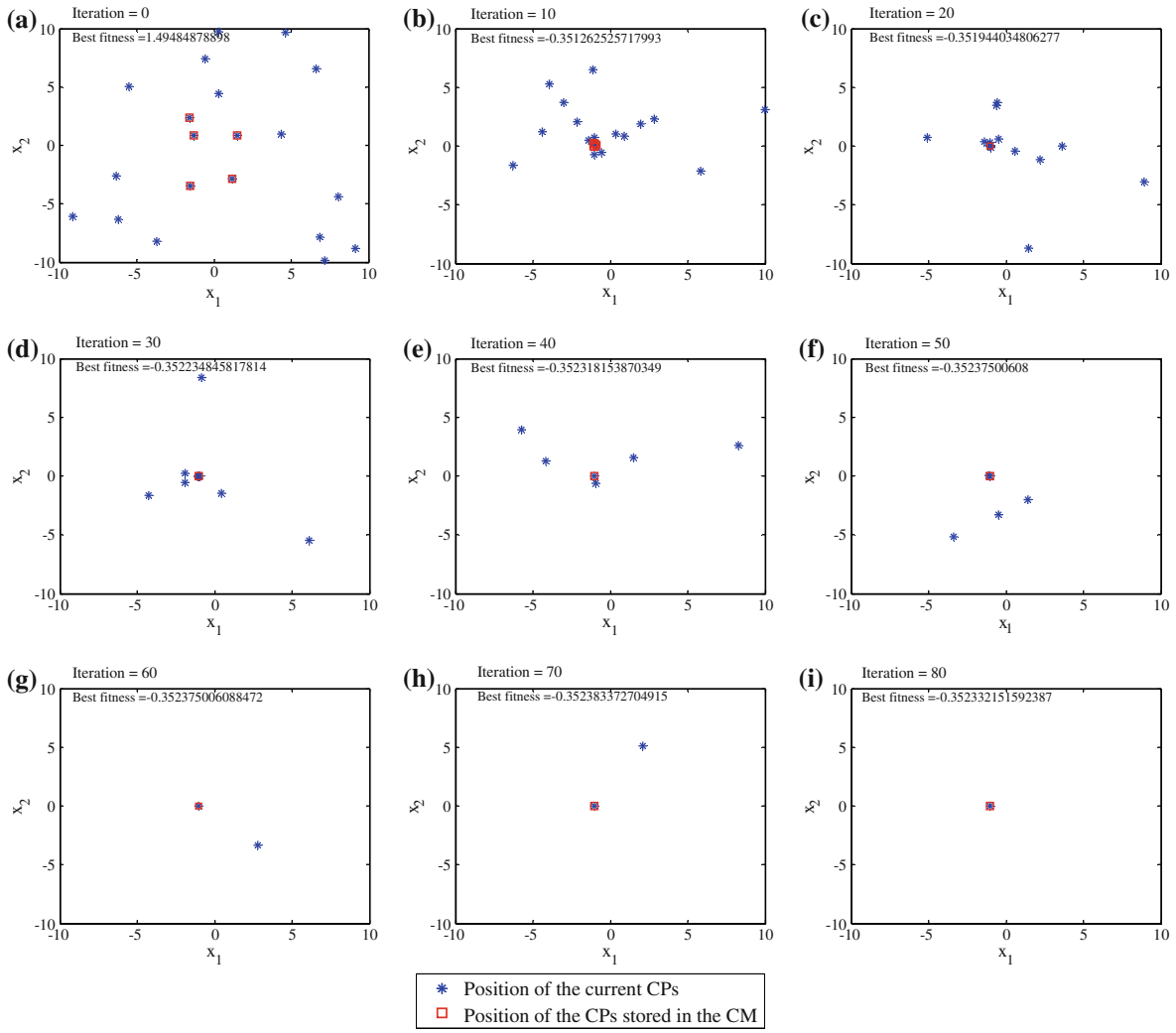

Fig. 8 The positions of the current CPs and the stored CPs in the CM for the first example

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J},$$

$$M = P(L + \frac{x_2}{2}), R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(\mathbf{X}) = \frac{6PL}{x_4x_3^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^3x_4},$$

$$P_c(\mathbf{X}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6,000\,\text{lb}, \quad L = 14\,\text{in},$$

$$E = 30 \times 10^6\,\text{psi}, \quad G = 12 \times 10^6\,\text{psi}. \tag{33}$$

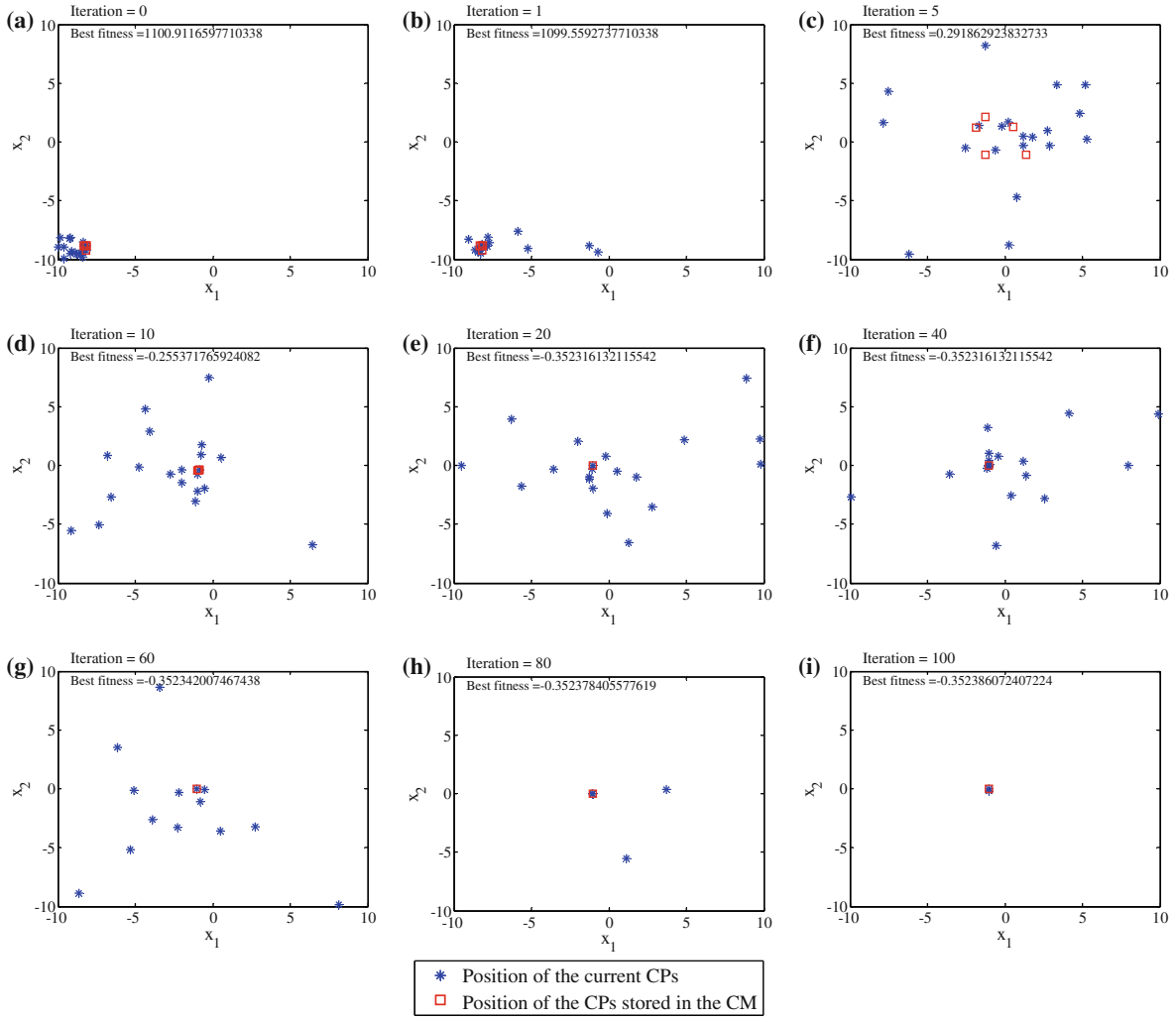

Fig. 9 The positions of the CPs for the first example when the all initial agents are introduced in a small part of the space

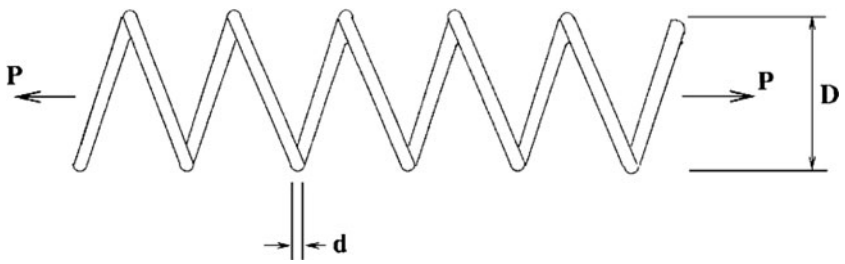

Fig. 10 A tension/compression spring

Variable bounds are

$$\begin{aligned} 0.1 \leq x_1 \leq 2, \\ 0.1 \leq x_2 \leq 10, \\ 0.1 \leq x_3 \leq 10, \\ 0.1 \leq x_4 \leq 2. \end{aligned} \tag{34}$$

Deb [26], Coello [21] and Coello and Montes [22] solved this problem using GA-based methods. Radgsdell and Phillips [25] compared optimal results of different optimization methods that were mainly

Table 3 Optimum results for the tension/compression spring design

Methods	Optimal design variables			
	$x_1(d)$	$x_2(D)$	$x_3(N)$	f_{cost}
Belegundu [19]	0.050000	0.315900	14.250000	0.0128334
Arora [20]	0.053396	0.399180	9.185400	0.0127303
Coello [21]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [22]	0.051989	0.363965	10.890522	0.0126810
He and Wang [23]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [24]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [1]	0.051865	0.361500	11.000000	0.0126432
Present work	0.051744	0.358532	11.165704	0.0126384

Table 4 Statistical results of different methods for the tension/compression spring

Methods	Best	Mean	Worst	Standard deviation
Belegundu [19]	0.0128334	N/A	N/A	N/A
Arora [20]	0.0127303	N/A	N/A	N/A
Coello [21]	0.0127048	0.012769	0.012822	3.9390e-5
Coello and Montes [22]	0.0126810	0.0127420	0.012973	5.9000e-5
He and Wang [23]	0.0126747	0.012730	0.012924	5.1985e-5
Montes and Coello [24]	0.012698	0.013461	0.16485	9.6600e-4
Kaveh and Talatahari [1]	0.0126432	0.012720	0.012884	3.4888e-5
Present work	0.0126384	0.012852	0.013626	8.3564e-5

N/A Not available

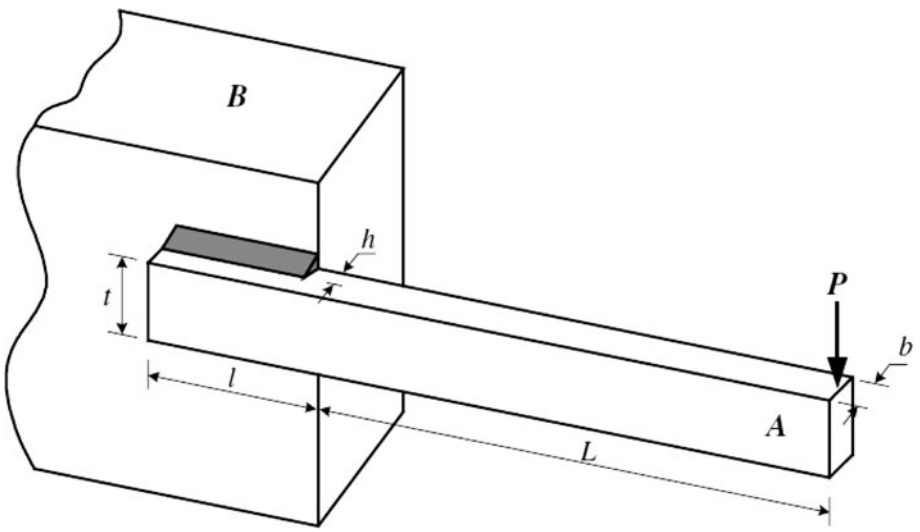

Fig. 11 A welded beam system

based on mathematical optimization algorithms. These methods are APPROX (Griffith and Stewart's successive linear approximation), DAVID (Davidon–Fletcher–Powell with a penalty function), SIMPLEX (Simplex method with a penalty function), and RANDOM (Richardson's random method) algorithms. Also, He and Wang [23] using CPSO, Montes and Coello [24] using evolution strategies and the authors [1] using the ACO solved this problem. The comparison of results are shown in Table 5. The statistical simulation results are summarized in Table 6. From Table 6, it can be seen that the standard deviation of the results by CSS in 30 independent runs is very small. In addition, it can be seen from Table 6 that the worst solution found by CSS is better than the best solution found by Ragsdell and Phillips [25] and the best solution found by Deb [26]. Also, the standard deviation of the results by CSS in 30 independent runs is very small.

Table 5 Optimum results for the design of welded beam

	Optimal design variables				
Methods	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	f_{cost}
Regsdell and Phillips [25]					
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 [26]	0.248900	6.173000	8.178900	0.253300	2.433116
Coello [21]	0.208800	3.420500	8.997500	0.210000	1.748309
Coello and Montes [22]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang [23]	0.202369	3.544214	9.048210	0.205723	1.728024
Montes and Coello [24]	0.199742	3.612060	9.037500	0.206082	1.737300
Kaveh and Talatahari [1]	0.205700	3.471131	9.036683	0.205731	1.724918
Present work	0.205820	3.468109	9.038024	0.205723	1.724866

Table 6 Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Regsdell and Phillips [25]	2.3815	N/A	N/A	N/A
Deb [26]	2.433116	N/A	N/A	N/A
Coello [21]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [22]	1.728226	1.792654	1.993408	0.074713
He and Wang [23]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [24]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [1]	1.724918	1.729752	1.775961	0.009200
Present work	1.724866	1.739654	1.759479	0.008064

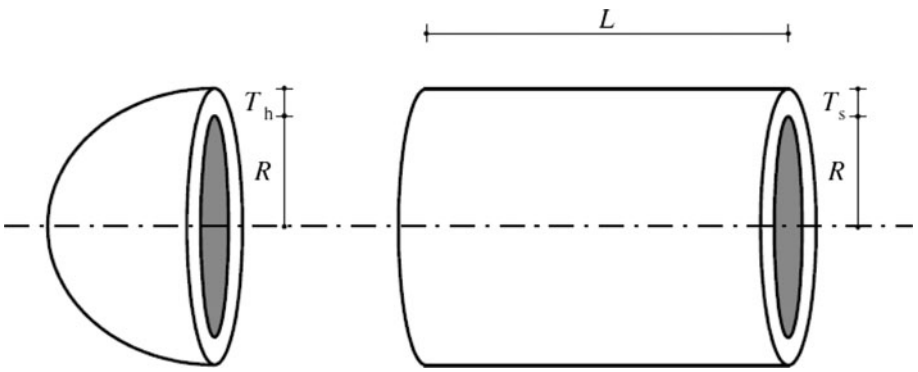

Fig. 12 Schematic of a pressure vessel

3.3.3 A pressure vessel design problem

A cylindrical vessel is clapped at both ends by hemispherical heads as shown in Fig. 12. The objective is to minimize the total cost, including the cost of material, forming and welding [27]:

$$f_{\text{cost}}(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{35}$$

where x_1 is the thickness of the shell (T_s), x_2 is the thickness of the head (T_h), x_3 is the inner radius (R), and x_4 is the length of cylindrical section of the vessel (L), not including the head. T_s and T_h are integer multiples of 0.0625 inch, the available thickness of the rolled steel plates, R and L are continuous.

Table 7 Optimum results for the pressure vessel

	Optimal design variables				
Methods	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	f_{cost}
Sandgren [27]	1.125000	0.625000	47.700000	117.701000	8,129.1036
Kannan and Kramer [28]	1.125000	0.625000	58.291000	43.690000	7,198.0428
Deb and Gene [29]	0.937500	0.500000	48.329000	112.679000	6,410.3811
Coello [21]	0.812500	0.437500	40.323900	200.000000	6,288.7445
Coello and Montes [22]	0.812500	0.437500	42.097398	176.654050	6,059.9463
He and Wang [23]	0.812500	0.437500	42.091266	176.746500	6,061.0777
Montes and Coello [24]	0.812500	0.437500	42.098087	176.640518	6,059.7456
Kaveh and Talatahari [1]	0.812500	0.437500	42.098353	176.637751	6,059.7258
Present work	0.812500	0.437500	42.103624	176.572656	6,059.0888

Table 8 Statistical results of different methods for the pressure vessel

Methods	Best	Mean	Worst	Standard deviation
Sandgren [27]	8,129.1036	N/A	N/A	N/A
Kannan and Kramer [28]	7,198.0428	N/A	N/A	N/A
Deb and Gene [29]	6,410.3811	N/A	N/A	N/A
Coello [21]	6,288.7445	6,293.8432	6,308.1497	7.4133
Coello and Montes [22]	6,059.9463	6,177.2533	6,469.3220	130.9297
He and Wang [23]	6,061.0777	6,147.1332	6,363.8041	86.4545
Montes and Coello [24]	6,059.7456	6,850.0049	7,332.8798	426.0000
Kaveh and Talatahari [1]	6,059.7258	6,081.7812	6,150.1289	67.2418
Present work	6,059.0888	6,067.9062	6,085.4765	10.2564

The constraints can be stated as

$$\begin{aligned} g_1(\mathbf{X}) &= -x_1 + 0.0193x_3 \le 0, \\ g_2(\mathbf{X}) &= -x_2 + 0.00954x_3 \le 0, \\ g_3(\mathbf{X}) &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0, \\ g_4(\mathbf{X}) &= x_4 - 240 \le 0. \end{aligned} \tag{36}$$

The design space is given as

$$\begin{aligned} 0 \le x_1 \le 99, \\ 0 \le x_2 \le 99, \\ 10 \le x_3 \le 200, \\ 10 \le x_4 \le 200. \end{aligned} \tag{37}$$

The approaches applied to this problem include a branch and bound technique [27], an augmented Lagrangian multiplier approach [28], genetic adaptive search [29], a GA-based co-evolution model [21], a feasibility-based tournament selection scheme [22], a co-evolutionary particle swarm optimization [23], an evolution strategy [24], and an improved ant colony optimization [1]. The best solutions obtained by the above-mentioned approaches are listed in Table 7, and their statistical simulation results are shown in Table 8. From Table 7, it can be seen that the best solution found by CSS is better than the best solutions found by other techniques. From Table 8, it can also be seen that the average searching quality of CSS is better than those of other methods.

4 Discussion and conclusion

Many meta-heuristic approaches are inspired by natural phenomena. This paper develops a new nature based algorithm. The method is called Charged System Search, and it is inspired by the Coulomb law known from

electrostatics and the laws of motion from Newtonian mechanics. CSS contains a number of agents which are called charged particles. Each CP is considered as a charged sphere of radius a, which has a uniform volume charge density and can impose an electric force on other CPs according to Coulomb's law. This force is attractive and its magnitude for the CP located inside the sphere is proportional to the separation distance between the CPs and for the CP located outside the sphere is inversely proportional to the square of the separation distance between the charged particles. The superposed forces and the laws for the motion determine the new location of the CPs. In this stage, each CP moves in the direction of the resultant forces and its previous velocity. From optimization point of view, this process provides a good balancing between the exploration and the exploitation paradigms of the algorithm which can considerably improve the efficiency of the algorithm.

The three essential concepts, self-adaptation step, cooperation step, and competition step, are considered in this algorithm. Moving towards good CPs provides the self-adaptation step. Cooperating CPs to determine the resultant force acting on each CP supplies the cooperation step and having larger force for a good CP, comparing a bad one, and saving good CPs in the Charged Memory (CM) provide the competition step.

Application of the CSS method to many benchmark and engineering functions shows that it outperforms evolutionary algorithms, and comparison of the results demonstrates the efficiency of the present algorithm.

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

References

1. Kaveh, A., Talatahari, S.: An improved ant colony optimization for constrained engineering design problems. Engineering Computations 27(1) (2010, in press)
2. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through Simulated Evolution. Wiley, Chichester (1966)
3. De Jong, K.: Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, University of Michigan, Ann Arbor, MI (1975)
4. Koza, J.R.: Genetic programming: a paradigm for genetically breeding populations of computer programs to solve problems. Report No. STAN-CS-90-1314, Stanford University, Stanford, CA (1990)
5. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
6. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Boston (1989)
7. Glover, F.: Heuristic for integer programming using surrogate constraints. Decis. Sci. **8**(1), 156–166 (1977)
8. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B **26**(1), 29–41 (1996)
9. 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)
10. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
11. Erol, O.K., Eksin, I.: New optimization method: Big Bang–Big Crunch. Adv. Eng. Softw. **37**, 106–111 (2006)
12. Kaveh, A., Talatahari, S.: Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput. Struct. **87**(17–18), 1129–1140 (2009)
13. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
14. Halliday, D., Resnick, R., Walker, J.: Fundamentals of Physics, 8th edn. Wiley, New York (2008)
15. Kaveh, A., Talatahari, S.: Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput. Struct. **87**(5–6), 267–283 (2009)
16. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Eng. **191**(11–12), 1245–1287 (2002)
17. Kaveh, A., Talatahari, S.: A particle swarm ant colony optimization for truss structures with discrete variables. J. Constr. Steel Res. **65**(8–9), 1558–1568 (2009)
18. Tsoulos, I.G.: Modifications of real code genetic algorithm for global optimization. Appl. Math. Comput. **203**, 598–607 (2008)
19. 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, USA (1982)
20. Arora, J.S.: Introduction to Optimum Design. McGraw-Hill, New York (1989)
21. Coello, C.A.C.: Use of a self-adaptive penalty approach for engineering optimization problems. Comput. Ind. **41**, 113–127 (2000)
22. 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)
23. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. **20**, 89–99 (2007)
24. 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**(4), 443–473 (2008)
25. Ragsdell, K.M., Phillips, D.T.: Optimal design of a class of welded structures using geometric programming. ASME J. Eng. Ind. Ser. B **98**(3), 1021–1025 (1976)

26. Deb, K.: Optimal design of a welded beam via genetic algorithms. Am. Inst. Aeronaut. Astronaut. J. **29**(11), 2013–2015 (1991)
27. Sandgren, E.: Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, Kissimine, FL, pp. 95–105 (1988)
28. 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)
29. 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. Springer, Berlin, pp. 497–514 (1997)