INTERNATIONAL JOURNAL OF OPTIMIZATION IN CIVIL ENGINEERING
Int. J.Optim. Civil Eng. 2011; 2:305-325

AN EFFICIENT CHARGED SYSTEM SEARCH USING CHAOS FOR GLOBAL OPTIMIZATION PROBLEMS

S. Talatahari[1], A. Kaveh[2,*,†] , R. Sheikholeslami[3]
[1]*Marand Faculty of Engineering, University of Tabriz, Tabriz, Iran*
[2]*Centre of Excellence for Fundamental Studies in Structural Engineering, Iran University of Science and Technology, Narmak, Tehran-16, Iran*
[3]*Department of Civil Engineering, University of Tabriz, Tabriz, Iran*

ABSTRACT

The Charged System Search (CSS) is combined to chaos to solve mathematical global optimization problems. The CSS is a recently developed meta-heuristic optimization technique inspired by the governing laws of physics and mechanics. The present study introduces chaos into the CSS in order to increase its global search mobility for a better global optimization. Nine chaos-based CSS (CCSS) methods are developed, and then for each variant, the performance of ten different chaotic maps is investigated to identify the most powerful variant. A comparison of these variants and the standard CSS demonstrates the superiority and suitability of the selected variants for the benchmark mathematical optimization problems.

Received: March 2011, Accepted: July 2011

KEY WORDS: Charged system search; chaos; optimization; chaos-based charged system search algorithm

1. INTRODUCTION

The Charged System Search (CSS) is one of the most recent meta-heuristic optimization techniques inspired by the governing laws of electrostatics in physics and the governing laws of motion from the Newtonian mechanics [1]. This algorithm is growing and its application is extending to various optimization problems such as discrete optimum design

[*]Corresponding author: A. Kaveh, School of Civil Engineering, Iran University of Science and Technology, Narmak, Tehran-16, Iran
[†]E-mail address: alikaveh@iust.ac.ir

of truss structures [2], design of skeletal structures [3], grillage system design [4], optimization of geodesic domes [5] and configuration optimization [6] etc. The CSS utilizes a number of solution candidates which are called charged particles (CPs). Each CP is treated as a charged sphere and it can exert electrical forces on the other agents (CPs) according to the Coulomb and Gauss laws of electrostatics. The resultant force acts on each CP creating an acceleration according to the Newton's second law. Finally, utilizing the Newtonian mechanics, the position of each CP is determined at any time based on its previous position, velocity and acceleration in the search space [1]. The comparison of the results of the CSS with those of the other heuristics shows a better performance of the CSS and demonstrates its efficiency in finding the optimum solutions [4].

On the other hand, chaos is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and includes infinite unstable periodic motions in nonlinear systems. Although it appears to be stochastic, it occurs in a deterministic nonlinear system under deterministic conditions [7].

Recently, the idea of using chaotic systems instead of random processes has been noticed in several fields. One of these fields is optimization theory. In random-based optimization algorithms, the role of randomness can be played by a chaotic dynamics. Experimental studies show the benefits of using chaotic signals instead of random signals, however, this is not mathematically proved yet. For example in evolutionary algorithms, chaotic sequences increase the value of some measured algorithm-performance indexes with respect to random sequences [8]. Chaotic sequences have been proven to be easy and fast to generate and store, and there is no need for storing long sequences. Merely a few functions (chaotic maps) and few parameters (initial conditions) are needed even for very long sequences. In addition, an enormous number of different sequences can be generated simply by changing its initial condition. Moreover these sequences are deterministic and reproducible [9].

This paper presents Chaotic Charged System Search (CCSS) methods for finding global optimization problems. The CCSS algorithms utilizing different chaotic systems substitute random numbers for different parameters of the CSS. Thus different methods that use chaotic maps as efficient alternatives to pseudorandom sequences have been proposed. In order to evaluate these algorithms, some mathematical benchmark examples are studied. The results reveal the improvement of the new algorithm due to the application of the deterministic chaotic signals in place of the random sequences.

The remaining of this paper is organized as follows. Review of the CSS is briefly presented in Section 2. The chaotic maps utilized for generating the chaotic sequences in the CSS steps of the present experiments are listed in Section 3. In Section 4 different chaotic-based methods are proposed which are called Chaotic Charged System Search (CCSS) algorithms. Initialization and parametric studies are presented in Section 5 and in Section 6, the suggested methods are evaluated through benchmark problems, and the results are compared to designate the most efficient approach in Section 7. Finally, the conclusion is drawn in Section 8 based on the reported comparison analyses.

2. CHARGED SYSTEM SEARCH ALGORITHM

The Charged System Search (CSS) algorithm is based on the Coulomb and Gauss laws from electrical physics and the governing laws of motion from the Newtonian mechanics. This algorithm can be considered as a multi-agent approach, where each agent is a Charged Particle (CP). Each CP is considered as a charged sphere with radius a, having a uniform volume charge density and is equal to

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

where *fitbest* and *fitworst* are the best and the worst fitness of all the particles; *fit*(*i*) represents the fitness of the agent *i*, and *N* is the total number of CPs. The initial positions of CPs are determined randomly in the search space using

$$x_{i,j}^{(o)} = x_{i,\min} + rand_{ij} \cdot (x_{i,\max} - x_{i,\min}), \quad i = 1,2,...,N \tag{2}$$

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

$$v_{i,j}^{(o)} = 0, \quad i = 1,2,...,N \tag{3}$$

CPs can impose electric forces on the others, and its magnitude for the CP located inside the sphere is proportional to the separation distance between the CPs, and for a CP located outside the sphere is inversely proportional to the square of the separation distance between the particles. The kind of the forces can be attractive or repelling determined by using a force parameter ar_{ij} defined as:

$$ar_{ij} = \begin{cases} +1 & k_t < \mathrm{r}and_{ij} \\ -1 & k_t > \mathrm{r}and_{ij} \end{cases} \tag{4}$$

where ar_{ij} determines the type of the force, in which +1 represents the attractive force and −1 denotes the repelling force, and k_t is a parameter to control the effect of the kind of the force. In general the attractive force collects the agents in a part of search space and the repelling force strives to disperse the agents. The resultant force is redefined as

$$\mathbf{F}_j = \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) ar_{ij} p_{ij} (\mathbf{X}_i - \mathbf{X}_j) \quad \begin{cases} j = 1,2,...,N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{5}$$

where $\mathbf{F}_j$ is the resultant force acting on the jth CP; r_{ij} is the separation distance between two charged particles defined as

$$r_{ij} = \frac{\| \mathbf{X}_i - \mathbf{X}_j \|}{\| (\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best} \| + \varepsilon} \tag{6}$$

Here $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and jth CPs, respectively; $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number to avoid singularity. **The** p_{ij} determines the probability of moving each CP toward the others as

$$p_{ij} = \begin{cases} 1 & \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand \vee fit(i) > fit(j) \\ 0 & \text{else} \end{cases} \tag{7}$$

The resultant forces and the laws of the motion determine the new location of the CPs. At this stage, each CP moves towards its new position under the action of the resultant forces and its previous velocity as

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{8}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{9}$$

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 (0,1). If each CP moves out of the search space, its position is corrected using the harmony search-based handling approach [1]. In addition, to save the best results, a memory, known as the Charged Memory, is utilized.

3. CHAOTIC MAPS

For simulating complex phenomena, sampling, numerical analysis, decision making and in particular in heuristic optimization, random sequences are needed with a long period and reasonable uniformity [10,11]. Chaos is a deterministic, random-like process found in nonlinear, dynamical system, which is non-period, non-converging and bounded [12]. The nature of chaos looks to be random and unpredictable, possessing an element of regularity. Mathematically, chaos is randomness of a simple deterministic dynamical system, and chaotic system may be considered as the sources of randomness [10, 11].

A chaotic map is a discrete-time dynamical system running in chaotic state as

$$cm_{k+1} = f(cm_k) \tag{10}$$

The following chaotic sequence

$$\{cm_k : k = 1,2,...\} \tag{11}$$

can be used as a spread-spectrum sequence for random number sequence. Chaotic sequences have been proven to be easy and fast to generate and store, and therefore there is no need for storing long sequences [9]. One needs merely a few functions (chaotic maps) and few parameters (initial conditions) for very long sequences. Also an enormous number of different sequences can be generated simply by changing its initial condition. In addition, these sequences are deterministic and reproducible. The choice of chaotic sequences can by justified theoretically through their unpredictability, corresponding to their spread-spectrum characteristic and ergodic properties.

When a random parameter is needed in the CSS algorithm, it can be generated by iterating one step of the chosen chaotic map (*cm*). The selected chaotic maps for the experiments are listed in the following subsections.

3.1. Logistic map

This map, whose equation appears in nonlinear dynamics of biological population, highlights the chaotic behavior [13]

$$cm_{k+1} = a \cdot cm_k(1 - cm_k) \tag{12}$$

In this equation, x_k is the kth chaotic number, with k denoting the iteration number. Obviously, $cm_k \in (0,1)$ under the conditions that the initial $cm_o \in (0,1)$. In the experiments $a = 4$ is used.

3.2. Tent map

Tent map [14] resembles the logistic map. It generates chaotic sequences in (0,1) assuming the following form

$$cm_{k+1} = \begin{cases} cm_k / 0.7 & cm_k < 0.7 \\ 10/3 cm_k(1 - cm_k) & otherwise \end{cases} \tag{13}$$

3.3. Sinusoidal iterator

This iterator [13] is represented by

$$cm_{k+1} = a \cdot cm_k^2 \sin(\pi \cdot cm_k) \tag{14}$$

For $a = 2.3$ and $cm_o = 0.7$ it has the following simplified form

$$cm_{k+1} = \sin(\pi \cdot cm_k) \tag{15}$$

It generates chaotic sequence in (0, 1).

3.4. Gauss map

The Gauss map is utilized for testing purpose in the literature [14] and is represented by

$$cm_{k+1} = \begin{cases} 0 & cm_k = 0 \\ 1/cm_k \bmod(1) & otherwise \end{cases}$$

$$1/cm_k \bmod(1) = \frac{1}{cm_k} - \left[\frac{1}{cm_k}\right] \tag{16}$$

Here, [*cm*] denotes the largest integer less than *cm* and acts as a shift on the continued fraction representation of numbers. This map also generates chaotic sequences in (0,1).

3.5. Circle map

The Circle map [15] is represented by

$$cm_{k+1} = cm_k + b - (a/2\pi)\sin(2\pi \cdot cm_k) \bmod(1) \tag{17}$$

With $a = 0.5$ and $b = 0.2$, it generates a chaotic sequence in (0, 1).

3.6. Sinus map

Sinus map is defined as

$$cm_{k+1} = 2.3(cm_k)^{2\sin(\pi \cdot cm_k)} \tag{18}$$

3.7. Henon map

This map is a nonlinear 2-dimensional map most frequently employed for testing purposes, and it is represented by

$$cm_{k+1} = 1 - a \cdot cm_k^2 + b \cdot cm_{k-1} \tag{19}$$

The suggested parameter values are $a = 1.4$ and $b = 0.3$.

3.8. Ikeda map

An Ikeda map is a discrete-time dynamical system defined by [16]

$$\begin{aligned} x_{n+1} &= 1 + 0.7\left(x_n \cos(\theta_n) - y_n \sin(\theta_n)\right), \\ y_{n+1} &= 0.7\left(x_n \sin(\theta_n) + y_n \cos(\theta_n)\right), \\ \theta_n &= 0.4 - \frac{6}{1 + x_n^2 + y_n^2} \end{aligned} \tag{20}$$

3.9. Liebovtech map

Another example of chaotic maps is Liebovitch map proposed by Liebovitch and Toth [17]. This map consists of three piecewise linear segments on non-overlapping subintervals on the interval (0, 1). This map is defined by the following equations

$$x_{k+1} = \begin{cases} \alpha_1 x_k & 0 < x_k \le d_1, \\ \dfrac{d_2 - x_k}{d_2 - d_1} & d_1 < x_k \le d_2, \\ 1 - \alpha_2 (1 - x_k) & d_2 < x_k \le 1 \end{cases} \tag{21}$$

where d_1, $d_2 \in (0,1)$ with $d_1 < d_2$ and

$$\alpha_1 = \frac{d_2}{d_1}(1 - (d_2 - d_1)),$$
$$\alpha_2 = \frac{1}{d_2 - 1}((d_2 - 1) - d_1(d_2 - d_1)). \tag{22}$$

3.10. Zaslavskii map

One of the interesting dynamic systems evidencing chaotic behavior is the Zaslavskii map [18], The corresponding equation is given by:

$$\begin{aligned} y(k+1) &= [y(k) + \nu + az(k+1)] (\text{mod } 1) \\ z(k+1) &= \cos(2\pi y(k)) + e^{-r} z(k) \end{aligned} \tag{23}$$

where mod is the modulus after division and $\nu = 400$, $r = 3$, $a = 12.6695$. In this case, the values of $z(t) \in [-1.0512, 1.0512]$.

4. CHAOTIC CHARGED SYSTEM SEARCH ALGORITHM

In the standard CSS since it is not possible to change the parameters during subsequent iterations, random initialization of the CSS and the adjusted limit parameters may affect the performance of the algorithm and reduce its convergence speed. Though, the standard CSS uses fixed predefined values for k_t, k_a and k_v, however when these are multiplied to random numbers, the resultant values will have randomized nature. These values are the key factors to control the balance of the exploration and exploitation of the algorithm. However their changes are limited to effects of the related random numbers during the subsequent iterations. Therefore, determining suitable values for the constant parameters (k_t, k_a and k_v) becomes important while it is time consuming. In addition, for this purpose there is no deterministic approach. Due to the importance of these parameters on the performance of the algorithm in

one hand, and having no definite and reliable approach to determine these parameters on the other hand; their coefficients may be selected chaotically by using chaotic maps.

In this paper, sequences generated from chaotic systems substitute the random parameters utilized in the CSS algorithm, where it is necessary to make a random-based choice. In this way, it is intended to improve the global convergence and to prevent being trapped in a local solution. The use of chaotic sequences in the CSS algorithm can be helpful to escape more easily from local minima because of the ergodic property of chaotic variables that plays a central role in ensuring that the local agent information diffuses eventually over the entire network of agents.

The new chaotic CSS algorithms, denoted by CCSS, may simply be classified and described as follows:

4.1. CCSS-1

The initial positions of CPs are determined chaotically in the search space by iterating the selected chaotic map (*cm*) as shown in Figure 1.

```
j=0
while (j <= Number of CPs)
 Randomly initialize the first chaotic variable
 i=0
   while (i < Number of variables)
        Generate chaotic variable cm_{i,j} according to the selected
map:
        x_{i,j}=x_{i,min}+cm_{i,j} x (x_{i,max}-x_{i,min})
        i=i+1
   end while
 j=j+1
end while
```

Figure 1. Pseudo-code of CCSS-1

4.2. CCSS-2

In this algorithm the kind of the forces (attracting or repelling) is determined by using chaotic ar_{ij} defined as

$$ar_{ij} = \begin{cases} +1 & k_t < cm_{ij} \\ -1 & k_t > cm_{ij} \end{cases} \tag{24}$$

where cm_{ij} is a chaotic variable according to the selected map.

4.3. CCSS-3

In this algorithm the probability of moving each CP toward the others is determined as

$$p_{ij} = \begin{cases} 1 & \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > cm_{ij} \vee fit(i) > fit(j) \\ 0 & \text{else} \end{cases} \tag{25}$$

4.4. CCSS-4

The CCSS-2 and CCSS-3 are combined, i.e. the kind of the forces is determined by using Eq. (24), and p_{ij} value is modified by the selected chaotic map.

4.5. CCSS-5

The coefficient of the force term in Eq. (8) is modified by the selected chaotic maps and position update equation is modified by

$$\mathbf{X}_{j,new} = cm_{j1} \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{26}$$

where cm_{j1} is a chaotic variable based on the selected map.

4.6. CCSS-6

The coefficient of the velocity term in Eq. (8) is modified by the selected chaotic maps and the position update equation is modified as

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + cm_{j2} \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{27}$$

where cm_{j2} is a chaotic variable based on the selected map.

4.7. CCSS-7

The coefficients of the force and velocity terms in Eq. (8) are modified by the selected chaotic maps and position update equation is modified as:

$$\mathbf{X}_{j,new} = cm_{j1} \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + cm_{j2} \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{28}$$

4.8. CCSS-8

In this algorithm, CCSS-4 and CCSS-7 are combined.

4.9. CCSS-9

CCSS-1 and CCSS-8 are combined, that is the initial CPs locations are generated by the selected chaotic maps and ar_{ij}, p_{ij}, $rand_{j1}$ and $rand_{j2}$ values are modified by the selected chaotic maps when needed.

5. INITIALIZATION AND PARAMETRIC STUDIES

We used 50 different runs for each setting with completely different initial conditions. The final results are found to be almost independent of the initial guess. In fact, we have used

statistical measures such as mean objective values and their standard deviations to measure the performance of the algorithm, rather than relying simply on a few runs. This approach is reflected in the tables provided. The number of variables is set to 10 for the examples. The simulations for map limits of k_a as (0,0.5) results in a better performance while for the other generated maps, no changes are observed.

For most cases in our implementation, we have carried out some extensive sensitivity studies of the parameters such as the population size. From these simulations, we observed that the population size N = 20~30 is sufficient for most of the problems; here, N is set to 30 for the examples. With a fixed number of CPs (N) at each run, the benchmark mathematical functions are optimized within 500 iterations. This means the number of function evaluations is set to15,000.

6. TESTING WITH BENCHMARK PROBLEMS

In order to compare these variants of the new method, some well-known benchmark mathematical examples are considered from literature. The explanation of the examples is presented in Sections 6.1 and the criterion of success is explained in Section 6.3. The performance of the CCSS algorithms to optimize these functions is investigated in the next section.

6.1. Description of the examples

From the standard set of benchmark problems available in the literature, four well-known functions, one of which is uni-modal (containing only one optimum) and three of them are multi-modal (containing many local optima, but only one global optimum), are considered to test the efficacy of the proposed methods. 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 these functions are illustrated in Figure 2.

Table 1. Specifications of the benchmark problems

Function name (Property)	Interval	Function	Global minimum
Griewank (multi-modal)	$\mathbf{X} \in [-100,100]^n$	$f(X) = 1 + \frac{1}{200}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Rosenbrock (uni-modal)	$\mathbf{X} \in [-30, 30]^n$	$f(X) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0.0
Rastring (multi-modal)	$\mathbf{X} \in [-10, 10]^n$	$f(X) = \sum_{i=1}^{10} \left(x_i^2 - 10.\cos(2\pi x_i) + 10\right)$	0.0
Ackley (multi-modal)	$\mathbf{X} \in [-32, 32]^n$	$f(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\sqrt{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}\right) + 20 + e$	0.0

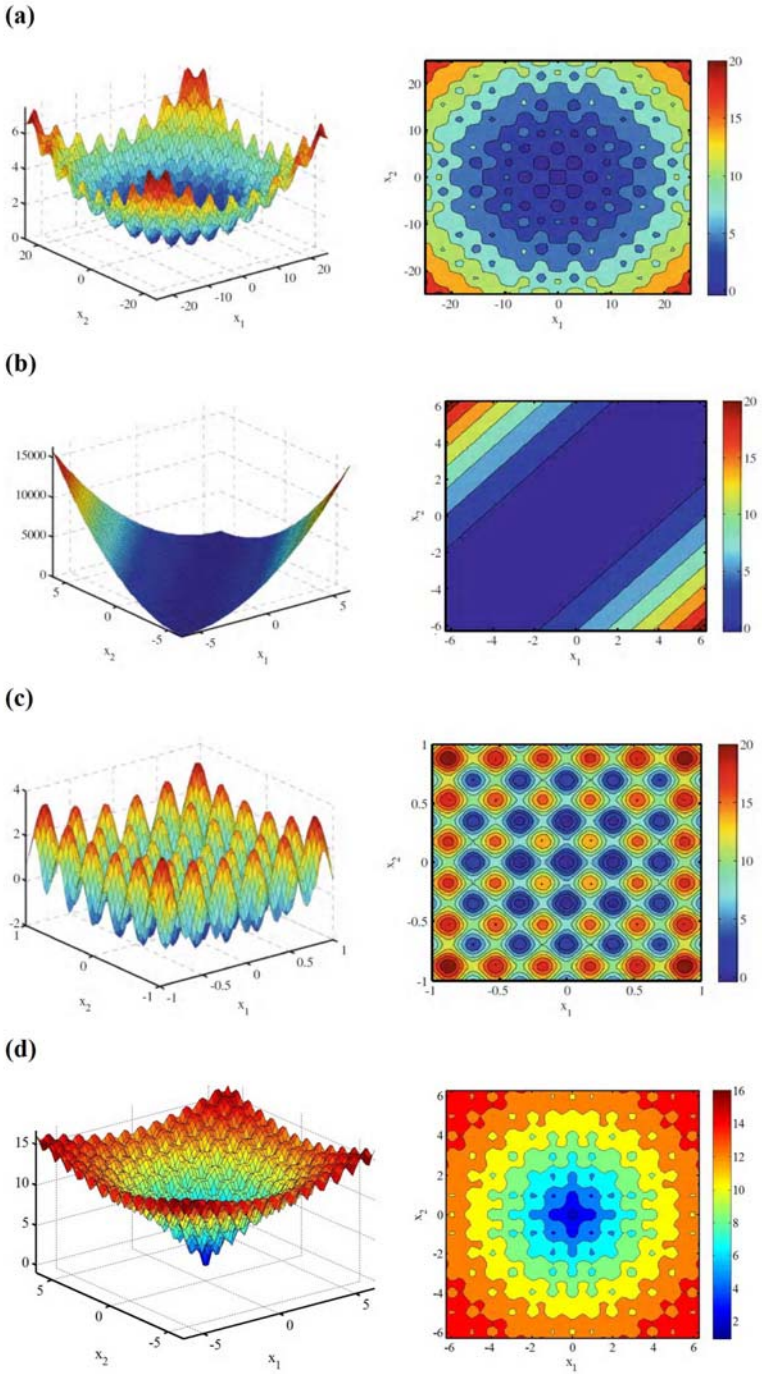

Figure 2. A perspective view and the related contour lines for some of function when n = 2., (a) Griewank (b) Rosenbrock, (c) Rastring , (d) Ackley.

6.2. Criterion of Success for the examples

There are many criteria in the literature for evaluating the performance of the algorithms. Here, the success rate is defined as

$$S_r = 100 \times \frac{N_{successful}}{N_{all}} \tag{29}$$

where N_{all} is the number of all trials, and $N_{successful}$ is the number of trials which finds the successful solution. Here, we identify a run as a successful run when it is very near to the global optimum for the examples. It should be noted that this distance changes for different search spaces. The criterion for a successful run can be defined as

$$\left\| fit^{gb} - fit^{*} \right\| \le (UB - LB) \times Q \tag{30}$$

where fit^{gb} is the global best obtained result by the proposed algorithms; UB and LB are the upper and lower bounds, respectively, and Q is an accepted tolerance.

7. EXPERIMENTAL RESULTS

All the four benchmark mathematical problems are solved by simulating the different variants of CCSS and the standard CSS methods. Two criteria are applied to terminate the algorithms: reaching a maximum number of iterations (a constant number) or a minimum error.

7.1. Results for Griewangk function

The five statistical analyses of the fitness values obtained through 50 simulation runs for Griewangk function using different chaotic maps are performed for each the CCSS algorithm.

All of the algorithms considered in this paper are simulated 50 times and the results are recorded. All swarms are initialized in the regions which do not include the global optimum, for a fair evaluation. From the recorded results statistical analyses are carried out and for the two first algorithms, these are presented in Tables 2 and 3. For each method, the Best (Min), Average (Mean), Worst (Max), Median, Standard Deviation (SD) are calculated from the simulated runs and then these are compared to determine the approximate rank of the related methods. For this propose, the rank of each algorithm is evaluated when for example comparing Best values and then the approximate rank can easily be evaluated. The similar comparative work is performed for the other seven methods; however in order to save some space only the two best maps for the other algorithms as well as two first ones are collected in Table 4.

Many statistical measures justify the superiority of the proposed methods in compression to the standard CSS. The low standard value of the proposed methods ensures the degree of consistency in producing the global optimal value. The results of Table 4 show the superior of the CCSS-7 and CCSS-6 methods to the other CCSS approaches. The CCSS-7 can find the best minimum value and the CCSS-6 is capable of reaches the best average and standard deviation.

Table 2. Statistical results for Griewank's function for the CCSS-1

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.	Approximated Rank
Logistic CCSS-1	1.50E-05 (7)*	2.56E-3 (2)	1.17E-3 (3)	1.01E-2 (2)	3.059E-3 (2)	2
Tent CCSS-1	8.01E-07 (2)	**2.33E-3 (1)**	1.26E-3 (4)	**8.10E-3 (1)**	**2.415E-3 (1)**	**1**
Sinusoidal CCSS-1	2.50E-05 (8)	1.99E-2 (9)	**9.26E-4 (1)**	4.17E-1 (9)	7.826E-2 (9)	9
Gauss CCSS-1	1.43E-06 (5)	3.06E-3 (3)	1.73E-3 (7)	1.10E-2 (3)	3.336E-3 (3)	3
Circle CCSS-1	3.56E-05 (9)	4.11E-3 (4)	1.32E-3 (5)	4.02E-2 (4)	8.219E-3 (4)	5
Sinus CCSS-1	1.12E-06 (4)	1.53E-2 (7)	3.75E-3 (9)	2.44E-1 (8)	4.496E-2 (7)	8
Henon CCSS-1	5.82E-05 (10)	8.76E-2 (10)	7.14E-3 (10)	4.58E-1 (10)	1.307E-1 (10)	10
Ikeda CCSS-1	8.11E-07 (3)	6.31E-3 (5)	1.10E-3 (2)	1.46E-1 (6)	2.637E-2 (6)	4
Liebovtech CCSS-1	**9.14E-08 (1)**	1.64E-2 (8)	2.43E-3 (8)	2.42E-1 (7)	4.582E-2 (8)	7
Zaslavski CCSS-1	3.34E-06 (6)	7.16E-3 (6)	1.68E-3 (6)	9.12E-2 (5)	1.755E-2 (5)	6

* The numbers in ( ) identify the rank of the utilized map compared to the remaining maps.

Table 3. Statistical results for Griewank's function for the CCSS-2

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.	Approximated Rank
Logistic CCSS-2	1.08E-05 (5)	3.95E-3 (3)	1.48E-3 (8)	2.04E-2 (2)	5.33E-3 (3)	3
Tent CCSS-2	**1.52E-09 (1)**	3.69E-3 (2)	9.11E-4 (4)	3.26E-2 (3)	6.34E-3 (4)	2
Sinusoidal CCSS-2	3.19E-4 (10)	1.21E-2 (9)	4.55E-3 (10)	1.31E-1 (6)	2.95E-2 (8)	6
Gauss CCSS-2	5.09E-0 (4)	9.70E-3 (6)	3.47E-4 (2)	2.41E-1 (10)	4.38E-3 (2)	4
Circle CCSS-2	2.40E-05 (7)	8.29E-3 (5)	7.45E-4 (3)	1.51E-1 (8)	2.97E-2 (9)	7
Sinus CCSS-2	4.89E-05 (8)	9.80E-3 (7)	3.92E-3 (9)	5.58E-2 (5)	1.30E-2 (6)	9
Henon CCSS-2	6.30E-05 (9)	5.55E-3 (4)	1.32E-3 (7)	4.52E-2 (4)	9.72E-3 (5)	5
Ikeda CCSS-2	4.63E-06 (3)	**1.49E-3 (1)**	**3.21E-4 (1)**	**1.32E-2 (1)**	**3.08E-3 (1)**	**1**
Liebovtech CCSS-2	1.53E-5 (6)	1.31E-2 (10)	1.31E-3 (6)	1.34E-1 (7)	2.93E-2 (7)	10
Zaslavski CCSS-2	4.16E-09 (2)	1.13E-2 (8)	1.22E-3 (5)	1.88E-1 (9)	3.48E-2 (10)	8

The success rates of the CCSS methods using different chaotic maps for Griewank function are presented in Table 5. The success rate for the standard CSS is equal to 56%, and according to Table 5, it is clear that many CCSS algorithms have improved the performance of the standard CSS. The CCSS-6 with an average value of 75.4% for the success rate is the best method. The CCSS-8, CCSS-9 and CCSS-7 are categorized as the next better approaches. In addition, good results for success rates are obtained when the CCSS-7 and CCSS-8 have been used with the sinus map.

Table 4. Statistical results of the best maps for Griewank's function for the CCSS algorithms

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.
Standard CSS	8.86E-06	2.88E-3	1.15E-3	1.55E-2	4.21E-2
CCSS-1					
Tent map	8.01E-7	2.33E-3	1.26E-3	8.10E-3	2.415E-3
Logistic map	1.50E-5	2.56E-3	1.17E-3	1.01E-2	3.059E-3
CCSS-2					
Ikeda map	4.63E-6	1.49E-3	3.21E-4	1.32E-2	3.08E-3
Tent map	1.52E-9	3.69E-3	9.11E-4	3.26E-2	6.34E-3
CCSS-3					
Logistic map	2.86e-7	1.38E-3	6.55E-4	**6.32E-3**	1.69E-3
Sinusoidal map	2.95E-6	1.46E-3	4.03E-4	1.61E-2	3.01E-3
CCSS-4					
Circle map	7.64E-7	2.32E-3	9.25E-4	1.41E-2	3.65E-3
Zaslavski map	8.62E-6	2.45E-3	1.17E-3	9.29E-3	2.88E-3
CCSS-5					
Sinus map	7.22E-9	6.85E-3	1.07E-3	1.48E-1	2.66E-2
Liebovtech map	1.28E-9	1.03E-2	2.28E-3	8.88E-1	9.44E-2
CCSS-6					
Liebovtech map	1.98E-7	1.24E-2	6.70E-4	2.49E-1	4.60E-2
Zaslavski map	2.77E-6	**1.06E-3**	4.81E-4	7.45E-3	**1.55E-3**
CCSS-7					
Sinusoidal map	7.03E-5	3.61E-3	2.01E-3	2.69E-2	5.17E-3
Sinus map	**1.00E-9**	1.11E-2	1.09E-4	1.96E-1	4.32E-2
CCSS-8					
Sinus map	2.01E-07	5.75E-3	2.91E-05	7.60E-2	1.73E-2
Sinusoidal map	7.47E-6	1.92E-2	5.67E-4	2.7272E-2	6.42E-2
CCSS-9					
Sinus map	6.22E-08	2.93E-3	**1.78E-05**	4.64E-2	1.02E-2
Tent map	8.12E-05	03.82E-2	5.27E-4	8.93E-2	9.09E-2

Table 5. Success rate of the standard and chotic-based CSS for Griewank's function. (Q =1E-5)

Chaotic Map	CCSS								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Logistic map	10	56	74	58	74	76	60	66	60
Tent map	66	60	58	50	74	90	68	70	84
Sinusoidal map	64	42	80	64	50	84	66	70	64
Gauss map	70	84	68	54	56	78	64	70	84
Circle map	50	88	58	64	50	84	66	74	70
Sinus map	60	34	54	74	94	68	98	98	86
Henon map	40	60	48	48	10	40	50	46	42
Ikeda map	24	88	78	68	64	78	66	90	86
Liebovtech map	70	58	68	78	66	68	94	94	80
Zaslavski map	48	58	58	70	66	88	68	74	70
Aveage rate	50	62.4	64.4	62.8	60.4	75.4	70	75.2	72.6
Best rate	70	88	80	78	94	90	98	98	86

7.2. Results for Rosenbrock function

Table 6 represents the results obtained from different CCSS methods for the Rosenbrock function. For this example, the CCSS-9, CCSS-6 and CCSS-7 methods have shown better performance than other methods and especially the standard CSS. The overall success rates of the CCSS variants as well as the standard CSS for Rosenbrock function are shown in Table 7, when Q is set to 1E-3. The success rate for the standard CSS is obtained to be 40%. The CCSS-8, CCSS-9 and CCSS-6 have better performance in average. The best success rate belongs to the CCSS-8 when the sinus map is utilized, improving the performance of the standard CSS twice comparing their success rates.

Table 6. Statistical results of the best maps for Rosenbrock's function for the CCSS algorithms

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.
Standard CSS	1.36E-3	5.84E-2	8.21E-2	1.15E-1	3.83E-1
CCSS-1					
Tent map	5.66E-06	7.60E-2	7.68E-2	1.21E-1	3.66E-1
Circle map	4.20E-04	6.33E-2	6.88E-2	4.82E-2	2.04E-1
CCSS-2					
Sinusoidal map	1.56E-06	9.80E-2	8.40E-2	1.11E0	1.96E-1
Gauss map	3.00E-04	4.09E-2	2.20E-2	9.82E-2	4.04E-2
CCSS-3					
Sinus map	9.09E-04	3.94E-2	1.34E-2	8.93E-2	4.13E-2
Circle map	1.20E-06	5.23E-2	5.99E-2	1.00E-1	4.06E-2
CCSS-4					
Circle map	9.14E-04	5.00E-2	4.01E-2	9.21E-2	3.74E-2
Tent map	3.33E-04	4.55E-2	3.69E-2	1.03E-1	3.75E-2
CCSS-5					
Liebovtech map	7.36E-07	2.80E-2	9.85E-2	9.63E-2	6.57E-2
Zaslavski map	4.94E-06	1.03E-1	3.00E-1	1.16E-1	2.25E-1
CCSS-6					
Sinusoidal map	**1.24E-08**	6.57E-2	8.87E-2	2.44E-1	5.57E-2
Sinus map	1.45E-07	1.07E-1	9.56E-2	2.81E-1	6.98E-2
CCSS-7					
Sinus map	7.11E-05	3.24E-2	1.71E-2	2.11E-1	4.66E-1
Liebovtech map	2.18E-05	7.99E-2	**1.03E-2**	3.11E-1	9.33E-1
CCSS-8					
Sinus map	3.60E-06	5.11E-2	1.61E-2	2.65E-1	7.50E-2
Sinusoidal map	2.28E-06	9.03E-2	1.71E-2	3.82E-1	5.28E-2
CCSS-9					
Sinus map	1.19E-6	1.96E-2	8.30E-2	**2.42E-2**	**1.19E-2**
Gauss map	9.57E-6	**1.30E-2**	1.14E-2	5.24E-2	9.57E-2

7.3. Results for Rastrigin function

Overall success rates of the CCSS variants are computed and summarized in Table 8. The

success rate for the standard CSS is only 46%. Similar to the previous example, The CCSS-8, CCSS-9 and CCSS-6 have the best performances in average.The results obtained for this example is presented in Table 9. The CCSS methods have shown better performance than the standard CSS method. The CCSS-8, CCSS-7 and CCSS-9 methods have shown better performance when the statistical analyses are compared. CCSS-8 using the sinus map seems be the best CCSS among others.

Table 7. Success rate of the standard and chotic-based CSS for Rosenbrock's function (Q =1E-3)

Chaotic Map	**CCSS**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Logistic map	28	50	50	36	44	30	32	42	44
Tent map	60	56	42	62	60	60	60	62	60
Sinusoidal map	44	50	54	46	50	72	54	72	54
Gauss map	50	60	54	46	50	54	48	58	66
Circle map	54	52	42	54	50	60	42	46	44
Sinus map	40	50	66	46	50	60	50	78	76
Henon map	28	28	24	24	22	30	34	30	32
Ikeda map	42	50	42	44	60	58	50	54	56
Liebovtech map	56	46	54	52	68	72	60	72	64
Zaslavski map	50	50	54	50	52	56	60	52	60
Aveage rate	45.2	49.2	48.2	46	50.6	55.2	49	56.6	55.6
Best rate	60	60	66	62	68	72	60	78	76

Table 8. Success rate of the standard and chotic-based CSS for Rastring's function (Q =1E-4)

Chaotic Map	**CCSS**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Logistic map	50	62	62	56	62	60	60	60	48
Tent map	**78**	56	68	68	56	**74**	50	88	86
Sinusoidal map	48	56	66	52	44	60	42	50	40
Gauss map	52	68	62	60	**72**	64	60	66	80
Circle map	62	50	42	44	68	42	62	62	60
Sinus map	28	44	66	50	64	62	56	**90**	70
Henon map	22	66	50	60	28	42	34	38	40
Ikeda map	60	**86**	**76**	**72**	60	60	60	64	64
Liebovtech map	50	54	72	60	66	68	68	62	64
Zaslavski map	56	60	44	60	60	68	**78**	68	72
Aveage rate	50.6	60.2	60.8	58.2	58	60	57	64.8	62.4
Best rate	78	86	76	72	72	74	78	90	86

Table 9. Statistical results of the best maps for Rastring's function for the CCSS algorithms

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.
Standard CSS	4.03E-3	5.34E-3	3.21E-3	1.58E-1	5.37E-3
CCSS-1					
Tent map	1.01E-07	1.98E-3	1.04E-4	1.03E-1	3.53E-3
Logistic map	4.17E-5	4.07E-3	1.48E-3	1.49E-1	4.92E-3
CCSS-2					
Circle map	2.22E-08	7.27E-3	7.89E-3	3.94E-1	9.73E-3
Ikeda map	4.71E-08	2.92E-3	1.33E-4	2.59E-1	5.97E-3
CCSS-3					
Sinusoidal map	2.43E-10	3.12E-3	3.46E-4	1.04E-1	4.24E-3
Liebovtech map	4.05E-09	4.04E-3	**6.52E-5**	3.98E-1	7.96E-3
CCSS-4					
Circle map	2.86E-08	6.89E-3	8.32E-3	3.98E-1	8.38E-3
Tent map	1.15E-07	4.39E-3	3.35E-4	2.11E-1	5.93E-3
CCSS-5					
Sinus map	2.90E-11	2.77E-3	1.52E-4	9.95E-2	3.63E-3
Circle map	1.53E-08	3.80E-3	1.38E-4	2.37E-1	5.92E-3
CCSS-6					
Liebovtech map	7.58E-5	1.75E-3	8.91E-4	6.62E-2	2.12E-3
Tent map	3.53E-08	6.42E-3	5.05E-3	4.00E-1	8.31E-3
CCSS-7					
Sinus map	5.59E-09	**8.43E-4**	2.54E-4	4.70E-2	1.25E-3
Tent map	7.13E-07	3.56E-3	2.39E-4	3.98E-2	7.87E-3
CCSS-8					
Sinus map	8.15E-5	1.35E-3	1.23E-3	3.44E-2	**1.23E-3**
Liebovtech map	**2.41E-11**	4.85E-3	5.89E-4	4.2E-1	8.8E-3
CCSS-9					
Sinus map	2.02E-6	3.74E-3	3.47E-3	**1.34E-2**	2.00E-3
Gauss map	4.05E-6	9.74E-3	9.47E-3	2.98E-2	408E-3

7.4. Results for Ackley function

Tables 10 and 11 collect the statistical results and the success rates of the Ackley function using the CCSS methods. The success rate for the standard CSS is 70%. The CCSS-8 using the sinus map can find the global optimum in all 50 different runs. In addition, this method

is the best method even when the statistical results are considered. The CCSS-3 utilizing the Gauss map is the second best method. This algorithm can also reach to an accept solution in all 50 runs.

Table 10. Statistical results of the best maps for Ackley's function for the CCSS algorithms

Chaotic Map	Best	Mean	Median	Worst	Std. Dev.
Standard CSS	8.10E-5	3.01E-3	4.90E-4	4.91E-2	1.09E-2
CCSS-1					
Sinus map	9.01E-5	8.79E-3	2.07E-3	9.46E-2	3.69E-3
Liebovtech map	6.06E-05	6.33E-3	2.90E-3	8.29E-2	7.68E-3
CCSS-2					
Tent map	3.50E-6	2.74E-4	5.87E-5	2.30E-3	6.32E-4
Gauss map	1.66E-6	7.63E-4	3.83E-5	1.07E-2	2.39E-3
CCSS-3					
Gauss map	5.02E-7	5.50E-5	3.07E-5	2.80E-4	7.17E-05
Zaslavski map	2.09E-6	1.45E-4	4.99E-5	1.05E-3	2.93E-4
CCSS-4					
Tent map	2.97E-6	8.48E-5	3.85E-5	4.84E-4	1.24E-4
Ikeda map	**1.64E-7**	5.12E-4	2.57E-5	4.86E-3	1.30E-3
CCSS-5					
Sinus map	1.19E-6	1.35E-4	1.15E-4	7.44E-2	1.81E-3
Gauss map	1.05E-5	1.45E-4	1.33E-4	3.52E-3	9.99E-4
CCSS-6					
Circle map	1.51E-6	5.86E-4	1.29E-4	5.495E-3	1.24E-3
Sinusoidal map	2.27E-7	1.15E-3	3.10E-5	1.76E-2	3.98E-2
CCSS-7					
Sinus map	2.68E-5	3.82E-4	1.78E-4	1.56E-2	4.19E-3
Zaslavski map	6.61E-6	5.09E-4	9.88E-5	2.34E-2	6.94E-3
CCSS-8					
Sinus map	1.46E-06	**3.08E-05**	**1.77E-05**	**1.02E-5**	**2.93E-05**
Liebovtech map	5.33E-07	1.02E-4	5.72E-05	9.35E-4	2.04E-4
CCSS-9					
Sinus map	8.00E-6	6.41E-4	3.11E-4	8.01E-3	8.00E-3
Tent map	5.63E-5	1.75E-4	1.59E-4	8.11E-3	5.63E-3

Table 11. Success rate of the standard and chotic-based CSS for Ackley's function (Q =1E-5)

Chaotic Map	CCSS								
	1	2	3	4	5	6	7	8	9
Logistic map	40	50	66	56	64	80	92	86	84
Tent map	90	90	96	100	86	76	80	96	96
Sinusoidal map	55	76	66	86	78	100	90	90	80
Gauss map	60	80	100	86	86	98	86	100	96
Circle map	60	86	76	90	80	100	94	60	80
Sinus map	70	60	86	86	100	80	100	100	100
Henon map	76	60	46	60	56	60	60	60	64
Ikeda map	50	90	96	96	90	82	90	100	96
Liebovtech map	50	65	90	84	96	84	86	60	74
Zaslavski map	60	85	86	82	78	70	90	86	80
Aveage rate	61.1	74.2	80.8	82.6	81.4	83	86.8	83.8	85
Best rate	90	90	100	100	100	100	94	100	100

8. DISCUSSION AND CONCLUSION

As the first report of hybridizing the charged system search and chaos, different methods are developed to solve numerical global optimization problems. The proposed approaches utilize different chaotic maps to adapt the parameters of the CSS algorithm. Nine new chaotic CSS algorithms are proposed and ten different chaotic maps are analyzed for the benchmark functions. The simulation resulting from 50 runs are presented in this paper for each method and each example demonstrate that some tested CCSS approaches are efficient methods to explore the search space and discover the global solution. From the statistical investigation, it is shown that the CCSS-8, utilizing the chaotic maps, instead of a_{ij}, p_{ij}, k_v, k_a, have better performance than other approaches. The second best methods can be listed as CCSS-6 and CCSS-9. In the CCSS-6, chaotic maps are only utilized as k_v and this shows the importance of this parameter in the CSS-based methods.

In order to simplify determining the most adaptive maps, Table 12 is presented containing two best maps for each described algorithm. The reported maps are obtained considering both statistical results and the success rates reported for the numerical examples. Then to sum up, for each algorithm three maps as the best ones are chosen. The results show that Tent, Sinus, Liebovtech, Gauss, Ikeda, Sinusoidal, Zaslavski, Circle maps can be categorized as better maps for the CCSS methods, respectively. This is not a general result for the CCSS methods and based on the parameter utilizing the chaotic map, this order can be changed. As a result, due to the superiorities of the present methods, more elaborated experiments may be performed to design/discover the best map which can be considered as the future work.

Table 12. Most success CCSS methods and their related best maps

	CCSS								
Function	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Griewank									
	Tent	Ikeda	Sinusoidal	Zaslavski	Sinus	Zaslavski	Sinus	Sinus	Sinus
	Gauss	Tent	Logistic	Circle	Liebovtech	Liebovtech	Liebovtech	Liebovtech	Tent
Rosenbrock									
	Tent	Gauss	Sinus	Tent	Liebovtech	Liebovtech	Liebovtech	Sinus	Sinus
	Circle	Tent	Sinusoidal	Circle	Tent	Sinusoidal	Sinusoidal	Sinusoidal	Gauss
Rastring									
	Tent	Ikeda	Liebovtech	Tent	Circle	Tent	Liebovtech	Sinus	Tent
	Circle	Gauss	Ikeda	Ikeda	Gauss	Liebovtech	Zaslavski	Tent	Sinus
Ackley									
	Tent	Tent	Gauss	Tent	Sinus	Sinusoidal	Zaslavski	Sinus	Sinus
	Sinus	Gauss	Ikeda	Ikeda	Liebovtech	Circle	Sinus	Liebovtech	Tent
Best maps									
1	Tent	Tent	Sinusoidal	Tent	Liebovtech	Liebovtech	Liebovtech	Sinus	Sinus
2	Circle	Gauss	Ikeda	Ikeda	Sinus	Sinusoidal	Zaslavski	Liebovtech	Tent
3	Sinus	Ikeda	Sinus/Gauss	Circle	Tent	Zaslavski	Sinus	Tent	Gauss

REFERENCES

1. Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search, *Acta Mech* 2010; **213**(3-4): 267-89.
2. Kaveh A, Talatahari S. A charged system search with a fly to boundary method for discrete optimum design of truss structures, *Asian J Civil Eng* 2010; **11**(3): 277-93.
3. Kaveh A, Talatahari S. Optimal design of skeletal structures via the charged system search algorithm, *Struct Multidiscip Optim* 2010; **41**(6): 893-911.
4. Kaveh A, Talatahari S. Charged system search for optimum grillage systems design using the LRFD-AISC code, *J Construct Steel Res* 2010; **66**(6):767-71.
5. Kaveh A, Talatahari S. Geometry and topology optimization of geodesic domes using charged system search. *Struct Multidiscip Optim* 2011; **43**(2): 215-29.
6. Kaveh A, Talatahari S. An enhanced charged system search for configuration optimization using the concept of fields of forces, *Struct Multidiscip Optim* 2011; **43**(3): 339-51.
7. Alatas B, Akin E, Bedri O. 2009. Chaos embedded particle swarm optimization algorithms, *Chaos Soliton Fract* 2009; **40**: 1715-34.
8. Tavazoei M, Haeri M. 2007. Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms, *J Comput Appl Math* 2007;**187**: 1076-85.
9. Heidari-Bateni G, McGillem CD. A chaotic direct-sequence spread spectrum communication system, *IEEE T Commun* 1994; **42**(2-4): 1524-7.
10. Schuster GG. *Deterministic Chaos: An Introduction* (2nd revised ed.), Federal Republic of Germany: Physick-Verlag, GmnH, Weinheim, 1998.

11. Coelho LS, Mariani VC. Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, *Expert Syst Appl* 2008; **34**: 1905–13.
12. Alatas B. Chaotic bee colony algorithms for global numerical optimization, *Expert Syst Appl* 2010; **37**: 5682–7
13. May RM. Simple mathematical models with very complicated dynamics, *Nature* 1976; **261**: 459.
14. Peitgen H, Jurgens H, Saupe D. *Chaos and Fractals*, Springer-Verlag, Berlin, Germany, 1992.
15. Zheng WM. Kneading plane of the circle map, *Chaos Soliton Fract* 1994; **4**:1221.
16. Dressler U, Farmer JD. Generalized Lyapunov exponents corresponding to higher derivatives, Physica D, 1992; **59**: 365-77.
17. Erramilli A, Singh RP, Pruthi P. Modeling Packet Traffic with Chaotic Maps, Royal Institute of Technology, Stockholm-Kista, Sweden,1994.
18. Zaslavskii GM. The simplest case of a strange attractor, *Physics Letters A*, 1978; **69**(3): 145–7.