


## Chaotic Charged System Search Algorithm

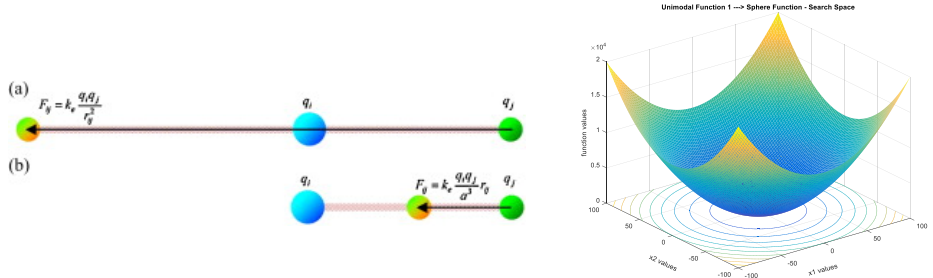
Serdar Özyön<sup>1\*</sup> 

<sup>1</sup> Department of Electric Electronic Engineering, University of Dumlupınar, Kütahya, Türkiye

### HIGHLIGHTS

- Chaotic number generators are more efficient than random number generators.
- Charged system search (CSS) algorithm is one of the effective optimization methods in literature.
- Chaotic charged system search (CCSS) algorithm has been developed with different chaotic mapping methods.

### GRAPHICAL ABSTRACT



### ARTICLE INFO

#### Article History

Received : 03/01/2020  
Revised : 26/02/2020  
Accepted : 26/02/2020  
Available online : 26/02/2020

#### Keywords

Random number generator  
Chaotic number generator  
Test functions  
Charged system search (CSS) algorithm  
Chaotic charged system search (CCSS) algorithm

### ABSTRACT

Optimization algorithms are frequently used in literature in order to be able to obtain a suitable solution, within acceptable times, for the problems which are impossible to solve with numerical methods or which take a long time to solve. Recently, an increasing number of optimizations have been designed and contributed to the literature. Besides, many of the designed algorithms show a great similarity with each other. Therefore, researchers do studies on strong and decisive optimization algorithms with various methods. The aim of these studies is to make the algorithm stronger, faster and more decisive. One of these aforementioned methods is the use of chaotic number generators in random number generations while forming the initial populations of the algorithms with good results, which has been applied to many algorithms. In this study, five different chaotic number generators with different structures have been integrated to charged system search (CSS) algorithm which is one of the strong optimization algorithms and has been applied in the solution of many problems successfully and chaotic charged system search (CCSS) algorithms have been proposed. In order to evaluate the performances of these suggested algorithms and to define which chaotic structure is more compatible with CSS algorithm, five unimodal test functions have been solved and the obtained results have been evaluated. In the suggested algorithms, while forming the particles in the first population, the first particle has been positioned in the search space randomly. As for the other particles, they have been positioned depending on the position of the first particle by using chaotic mapping methods. Namely, the individuals have spread in the search space according to a chaotic order, not randomly. The selected five chaotic methods have been defined as Duffing, Gauss/Mouse, Henon, Icmic ve Ikeda mapping methods. The test functions solved in the study are Schwefel's No: 2.22, Schwefel's No: 1.2, Schwefel's No: 2.21 and Rosenbrock functions. All functions have been solved 30 times each for 30 dimension and the statistical results and the graphics have been given.

### 1. INTRODUCTION

In many studies in literature, solution for the complicated engineering problems, which are difficult to solve with numerical methods or take a long time to solve, is searched by the help of optimization algorithms. This is because the solution of the multi-variable problems with big search spaces takes a long time with the classical methods. For the faster

\*Corresponding Author: serdar.ozyon@dpu.edu.tr

To cite this article: Özyön, S. (2020). Chaotic Charged System Search Algorithm. *Techno-Science*, 3(1), 38-45.

solution of these kinds of problems, many optimization algorithms have been developed recently. In these studies, it has been proposed that with the addition of some features to the algorithm, ignored previously during the design of the optimization algorithms, later by another researcher, the performance of the optimization algorithms can be bettered [1]. One of these betterment methods is the use of chaotic number generators in the optimization algorithms instead of random number generators [2,3].

In literature many optimization algorithms have been developed using different chaotic number generators and have been published in different studies with various names. These are chaotic biogeography-based optimization [4], chaotic fruit fly optimization [5], chaotic bat [6], chaotic krill swarm [7], chaotic harmony search [8], chaotic particle swarm optimization [9], chaotic gravitational search [10], chaotic differential evolution [11], chaotic bat swarm optimization [12] and chaotic artificial bee colony optimization [13] algorithms. In these studies, the performance of the selected optimization algorithms have been tried to be bettered by adding different number of chaotic mapping method with different methods.

In order to do a more effective and decisive search together with chaotic number generators having different structures, the use of charged system search (CSS) algorithm has been selected because of the positive feedbacks obtained from the literature. In the study five different chaotic mapping methods have been used with CSS algorithm. By the use of CSS algorithm together with chaotic mapping methods chaotic charged system search (CCSS) algorithm has been designed. In these developed algorithms, while forming the particles in the first population, the first particle has been positioned in the search space randomly. As for the other particles, they have been positioned depending on the position of the first individual by using five different chaotic mapping methods defined in the subparts. CCSS algorithm has been applied to five different, high dimensional, unimodal test functions successfully and the obtained results have been evaluated.

## 2. CHARGED SYSTEM SEARCH ALGORITHM(CSS)

Charged system search (CSS) algorithm is based on Coulomb and Gauss laws in electromagnetics, and basic act laws from Newton mechanics. This algorithm can be considered as a multi agent approach where each agent is a charged particle. Every charged particle is thought to be a sphere having  $a$  radius and uniform load density and it is expressed with the equation below [14-16].

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

In the equation,  $fitbest$  ve  $fitworst$  show the best and the worst fitness values belonging to all particles,  $fit(i)$   $i^{th}$  the fitness of the charged particle, as for  $N$  it shows the total number of charged particles (CPs). The initial positions of CPs in the search field is defined randomly and while defining equation (2) is used [14-16].

$$x_{i,j}^{(0)} = x_{i,min} + rand_{ij} \cdot (x_{i,max} - x_{i,min}), \quad i = 1, 2, \dots, N \quad (2)$$

In this equation  $x_{i,j}$ , defines the initial value of  $i^{th}$  variable of  $j^{th}$  particle.  $x_{i,min}$  and  $x_{i,max}$  are minimum and maximum values belonging to  $i^{th}$  variable. As for  $rand_{ij}$ , it is a random number generated at  $[0,1]$  interval. The initial velocity of the charged particles are taken as in the following [14-16].

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

Each CP, according to Coulomb law applies force on other CP's. The size of this force, for a CP in the sphere, is proportional with the distance among the CPs. For the CPs outside the sphere it is inversely proportional with the square of the distance among the particles. These forces can appear as pulling or pushing and they are calculated with the  $ar_{ij}$  force parameter defined in equation (4) [14-16].

$$ar_{ij} = \begin{cases} +1 & k_t < rand_{ij} \\ -1 & k_t > rand_{ij} \end{cases} \quad (4)$$

In the equation  $+1$  shows that force appears as pulling and  $-1$  value shows that it appears as pushing.  $k_t$  is the parameter that controls the influence of the appearing type of force. Generally, while the force appearing as pulling, gathers the CPs in a specific region in the search area, the force appearing as pushing tries to distribute the CPs.  $p_{ij}$  which defines the possibility of each CP to move towards other CPs, has been given in equation (5) [14-16].

$$p_{ij} = \begin{cases} 1, & \frac{fit(i) - fitbest}{fit(j) - fit(i)} > rand \vee fit(j) > fit(i) \\ 0, & else \end{cases} \quad (5)$$

The force appearing as a result, is calculated according to equation (6) [14-16].

$$F_j = q_j \sum_{i,i \neq j} \left( \frac{q_i}{a^3} r_{ij} i_1 + \frac{q_i}{r_{ij}^2} i_2 \right) p_{ij} (X_i - X_j) \begin{cases} j = 1, 2, \dots, N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} > a \end{cases} \quad (6)$$

Here,  $F_j$  is the force value that has an influence on  $j^{th}$  CP.  $r_{ij}$  is the distance between two charged particles and it has been defined in equation (7). As for  $a$ , it has been given in equation (8) [14-16].

$$r_{ij} = \frac{\|X_i - X_j\|}{\|(X_i - X_j) / 2 - X_{best}\| + \varepsilon} \tag{7}$$

$$a = 0,10 \times \max(\{x_{i,max} - x_{i,min} \mid i = 1, 2, \dots, n\}) \tag{8}$$

In the equation,  $X_i$  and  $X_j$  are the positions of  $i^{th}$  and  $j^{th}$  CP's respectively.  $X_{best}$  is the position of the aforementioned best CP ve  $\varepsilon$  is a small positive number added to block the uncertainty. The forces and the act laws appearing as a result, define the new positions of CPs. At this stage, under the influence of appearing forces as a result and its previous velocity, each CP moves towards its new position as in the following. The new velocity of CP is calculated according to equation (10) [14-16].

$$X_{j,new} = rand_{j1} \cdot k_a \cdot \frac{F_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot V_{j,old} \cdot \Delta t + X_{j,old} \tag{9}$$

$$V_{j,new} = \frac{X_{j,new} - X_{j,old}}{\Delta t} \tag{10}$$

Here  $k_a$  is the acceleration coefficient,  $k_v$  is the velocity coefficients that controls the influence of the previous velocity.  $rand_{j1}$  and  $rand_{j2}$  are two random numbers at  $[0,1]$  interval scattered to the sequence properly. If any CP moves out of the search area, its new position is corrected using manual correction approach. Also, a memory known as charged memory is utilized in order to save the best results. The flowchart of CSS algorithm, the equations of which were given above, has been given in Figure 1 [15].

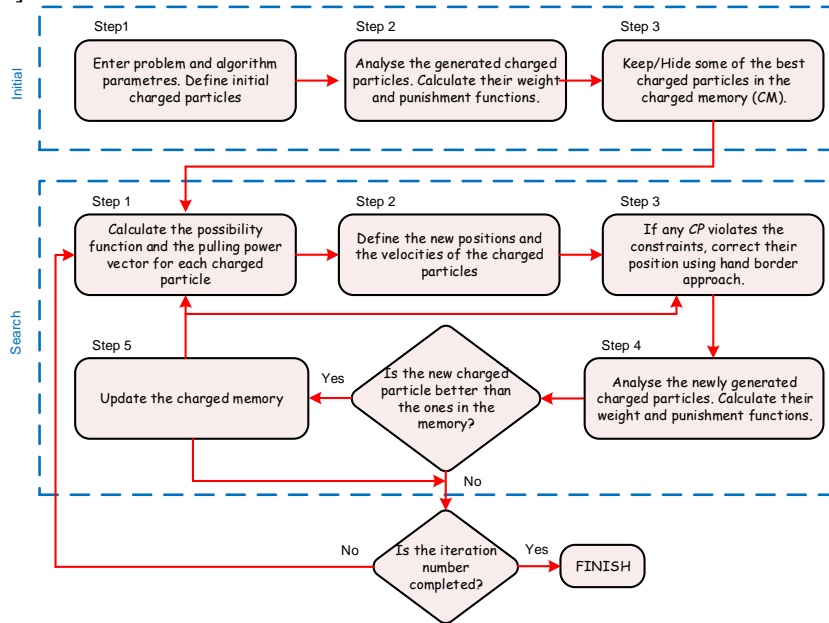


Fig. 1. The flowchart of CSS algorithm

### 3. CHAOTIC MAPPING METHODS AND TEST FUNCTIONS

While the charged particles in the initial population are formed according to equation (2) in classical CSS algorithm, random number generator is used for the  $rand_{ij}$  variable in the equation. Therefore, the first charged particles are positioned in the search space randomly. However, in this study the charged particles in the initial population have been formed using equation (11) together with a chaotic mapping method instead of equation (2). This situation enables the charged particles in the algorithm, to position according to a chaotic order in the search space [15,16].

$$x_{i,j}^{(0)} = x_{i,min} + CM_{ij} \cdot (x_{i,max} - x_{i,min}), \quad i = 1, 2, \dots, N \tag{11}$$

Here CM shows the used chaotic mapping method. In the study, five different chaotic mapping method, which are defined below, have been integrated to CSS algorithm respectively. A new chaotic charged system search (CCSS) algorithm has been formed and named for each chaotic map method. The chaotic mapping methods handled in the study;

#### Duffing Map (CCSS-1):

This mapping method depends on two constants as  $a$  and  $b$ . These constants were generally taken as  $a=2,75$  and  $b=0,2$  in literature. This method is a discrete version of Duffing equation in mathematics [17]. The mathematical presentation

belonging to this mapping method has been given in equation (12).

$$\begin{aligned} X_{k+1} &= Y_k \\ Y_{k+1} &= -bX_k + aY_k - Y_k^3, \quad a = 2.75, \quad b = 0.2 \end{aligned} \tag{12}$$

**Gauss/Mouse Map (CCSS-2)**

One of the well-known mapping methods widely used in the generation of chaotic sequences is Gauss/Mouse Map method. [18]. The mathematical presentation belonging to this mapping method has been given in equation (13).

$$\begin{aligned} X_{k+1} &= \begin{cases} X_k / 0.7 & X_k = 0 \\ 1 / X_k \text{ mod}(1) & X_k \in (0,1) \end{cases} \\ 1 / X_k \text{ mod}(1) &= \frac{1}{X_k} - \left[ \frac{1}{X_k} \right] \end{aligned} \tag{13}$$

**Henon Map (CCSS-3)**

The mathematical presentation belonging to this mapping method has been given in equation (14) [19].

$$\begin{aligned} X_{k+1} &= 1 - a(X_k)^2 + Y_k \\ Y_{k+1} &= bX_k, \quad a = 1.4, \quad b = 0.3 \end{aligned} \tag{14}$$

**ICMIC Map (CCSS-4)**

The mathematical presentation belonging to this mapping method has been given in equation (15) [20]. In this study it has been taken as  $a=2$ .

$$X_{k+1} = \sin\left(\frac{a}{X_k}\right) \text{ for } a > 0, \quad X_k \in [-1,0) \cup (0,1] \tag{15}$$

**Ikeda Map (CCSS-5)**

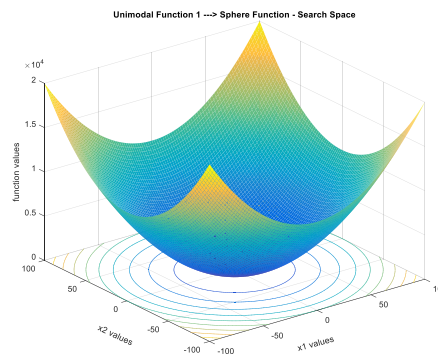
The mathematical presentation belonging to this mapping method has been given in equation (16) [21].

$$\begin{aligned} X_{k+1} &= 1 + u(X_k \cos t_n - Y_k \sin t_n) \\ Y_{k+1} &= u(X_k \sin t_n + Y_k \cos t_n) \quad t_n = 0,4 - \frac{6}{1 + X_k^2 + Y_k^2}, \quad u \geq 0.6 \end{aligned} \tag{16}$$

In order to evaluate their performances, the suggested algorithms have been applied to five test functions which were solved with different algorithms in literature previously. For the two-dimension search belonging to the 5 test functions handled in the study, 3-D drawings of the search spaces and their detailed information have been given as titles. The functions used for the test are high dimensional functions with wide search space. Besides these features of them, they were defined as unimodal functions with only one optimum point in literature. [1].

**Function 1 ( $f_1$ )**

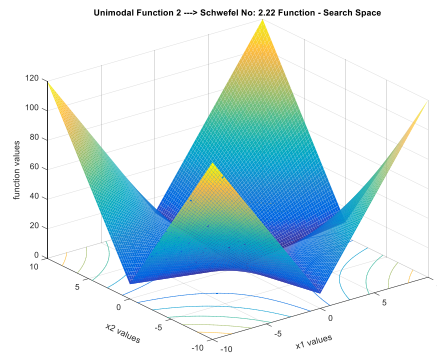
Equation	Function name	Search interval (S)	$f_{min}$
$f_1(x) = \sum_{i=1}^n x_i^2$	Sphere	$[-100,100]^n$	0



**Fig. 2.** 3D search space for  $f_1$

**Function 2 ( $f_2$ )**

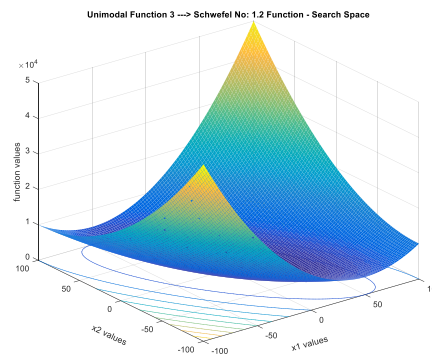
Equation	Function name	Search interval ( $S$ )	$f_{min}$
$f_2(x) = \sum_{i=1}^n  x_i  + \prod_{i=1}^n  x_i $	Schwefel's No: 2.22	$[-10,10]^n$	0



**Fig. 3. 3D search space for  $f_2$**

**Function 3 ( $f_3$ )**

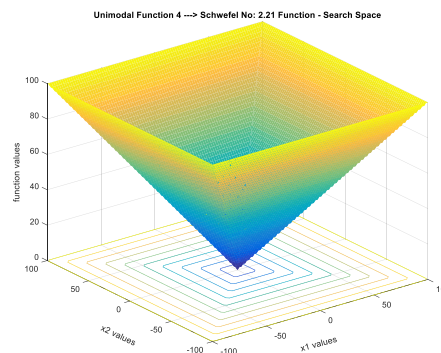
Equation	Function name	Search interval ( $S$ )	$f_{min}$
$f_3(x) = \sum_{i=1}^n \left( \sum_{j=1}^i x_j \right)^2$	Schwefel's No: 1.2	$[-100,100]^n$	0



**Fig. 4. 3D search space for  $f_3$**

**Function 4 ( $f_4$ )**

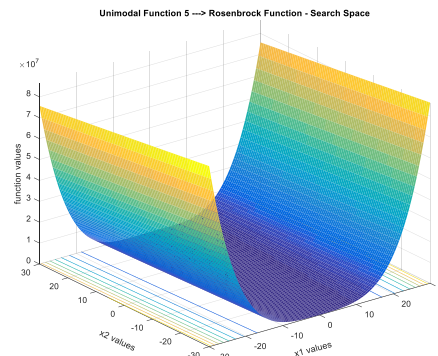
Equation	Function name	Search interval ( $S$ )	$f_{min}$
$f_4(x) = \max\{ x_i , 1 \leq i \leq n\}$	Schwefel's No: 2.21	$[-100,100]^n$	0



**Fig. 5. 3D search space for  $f_4$**

**Function 5 ( $f_5$ )**

Equation	Function name	Search interval ( $S$ )	$f_{min}$
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$	Rosenbrock	$[-30,30]^n$	0



**Fig. 6.** 3D search space for  $f_5$

**4. NUMERICAL RESULTS**

The developed CCSS algorithms have been evaluated in terms of decisiveness, velocity and performance. The program developed for the solution of the test functions has been operated on a workstation with Intel Xeon E5-2637 v4 3.50 GHz processor and ve 128 GB RAM memory. The parametre values belonging to CSS and CCSS algorithms have been given in Table 1, the values obtained from the solution of the test functions 30 times, have been given in Table 2.

**Table 1.** The parametre values belonging to CSS and CCSS algorithms

Iteration number	Particle number	Function call	Size/dimension	$\epsilon$
1000	40	40000	30	1e-6
$r$	Start $k_v$	Finish $k_v$	Start $k_a$	Finish $k_a$
0.01	0.9	0.1	0.9	0.1

**Table 2.** The data obtained for 30-D (30 runs)

		CSS	CCSS-1	CCSS-2	CCSS-3	CCSS-4	CCSS-5
$f_1$	The worst	1.667271e-09	2.725331e-06	9.233870e-08	2.118944e-07	1.041173e-06	2.733135e-10
	Average	5.682847e-11	9.918181e-08	3.098463e-09	7.067764e-09	4.120578e-08	9.361710e-12
	<b>The best</b>	<b>1.768770e-13</b>	<b>1.413612e-13</b>	<b>1.361198e-13</b>	<b>1.259318e-13</b>	<b>1.343800e-13</b>	<b>1.600811e-13</b>
	Std. Dev.	2.991014e-10	4.886578e-07	1.657166e-08	3.803536e-08	1.880472e-07	4.901465e-11
	Time (s)	5.11899	5.76155	5.10742	5.90843	5.2694	5.64954
$f_2$	The worst	1.363912e+01	5.889026e+00	8.075009e+00	4.820353e+00	4.118613e+00	3.670370e+00
	Average	2.094637e+00	1.271554e+00	1.911445e+00	1.057892e+00	8.738752e-01	7.938841e-01
	<b>The best</b>	<b>4.529751e-02</b>	<b>2.135175e-02</b>	<b>5.387716e-04</b>	<b>3.169634e-04</b>	<b>8.452568e-04</b>	<b>3.229249e-03</b>
	Std. Dev.	2.643627e+00	1.541919e+00	2.264557e+00	1.267744e+00	1.099575e+00	9.560239e-01
	Time (s)	5.42154	5.03773	5.17384	5.25671	5.24586	5.26855
$f_3$	The worst	1.280541e+04	2.461637e+04	2.100044e+04	2.037557e+04	2.628303e+04	1.914432e+04
	Average	6.817220e+03	8.877011e+03	9.651896e+03	9.628624e+03	1.014971e+04	9.296130e+03
	<b>The best</b>	<b>2.768918e+03</b>	<b>1.703858e+03</b>	<b>2.532552e+03</b>	<b>2.284302e+03</b>	<b>3.029275e+03</b>	<b>1.998961e+03</b>
	Std. Dev.	2.708370e+03	4.272358e+03	4.417440e+03	4.062541e+03	6.147091e+03	4.446951e+03
	Time (s)	5.12443	5.31113	5.53466	5.20004	5.05361	5.38474
$f_4$	The worst	2.329790e+00	1.343113e+00	6.582713e+00	6.006427e+00	3.113033e+00	4.902553e+00
	Average	6.431875e-01	3.635847e-01	9.587834e-01	1.156143e+00	6.391876e-01	6.654920e-01
	<b>The best</b>	<b>3.339139e-02</b>	<b>2.978997e-02</b>	<b>6.249314e-02</b>	<b>3.161251e-02</b>	<b>3.645939e-02</b>	<b>1.955341e-02</b>
	Std. Dev.	6.763090e-01	3.830884e-01	1.349296e+00	1.480670e+00	6.566506e-01	9.413274e-01
	Time (s)	5.19142	5.33927	5.54385	5.73277	5.36537	5.12443

$f_5$	The worst	3.486424e+04	9.613847e+03	5.798402e+03	8.753550e+03	1.227683e+04	1.017732e+04
	Average	2.436524e+03	4.762503e+03	7.479227e+02	1.108293e+03	1.101072e+03	1.089717e+03
	<b>The best</b>	<b>2.864179e+01</b>	<b>2.761978e+01</b>	<b>2.791248e+01</b>	<b>2.753720e+01</b>	<b>2.781845e+01</b>	<b>2.835373e+01</b>
	Std. Dev.	6.861555e+03	7.154702e+03	1.285696e+03	2.094315e+03	2.337070e+03	1.938622e+03
	Time (s)	5.32622	5.03662	5.03714	5.59948	5.51213	5.69106

When Table 2 is examined, it has been seen that the chaotic structures handled in the study have obtained better solutions for four test functions compared with classical CSS. Only in  $f_3$  test function classical CSS, has outclassed the chaotic structures. When five different chaotic mapping methods developed in the study are evaluated among themselves, it can be said that Henon and Ikeda chaotic mapping methods have more decisive structures compared with the others.

The graphics and the boxplots belonging to the best solutions obtained for the test functions in Table 2 for 30 runs and showing the convergence according to the iteration number have been shown in Figure 7.

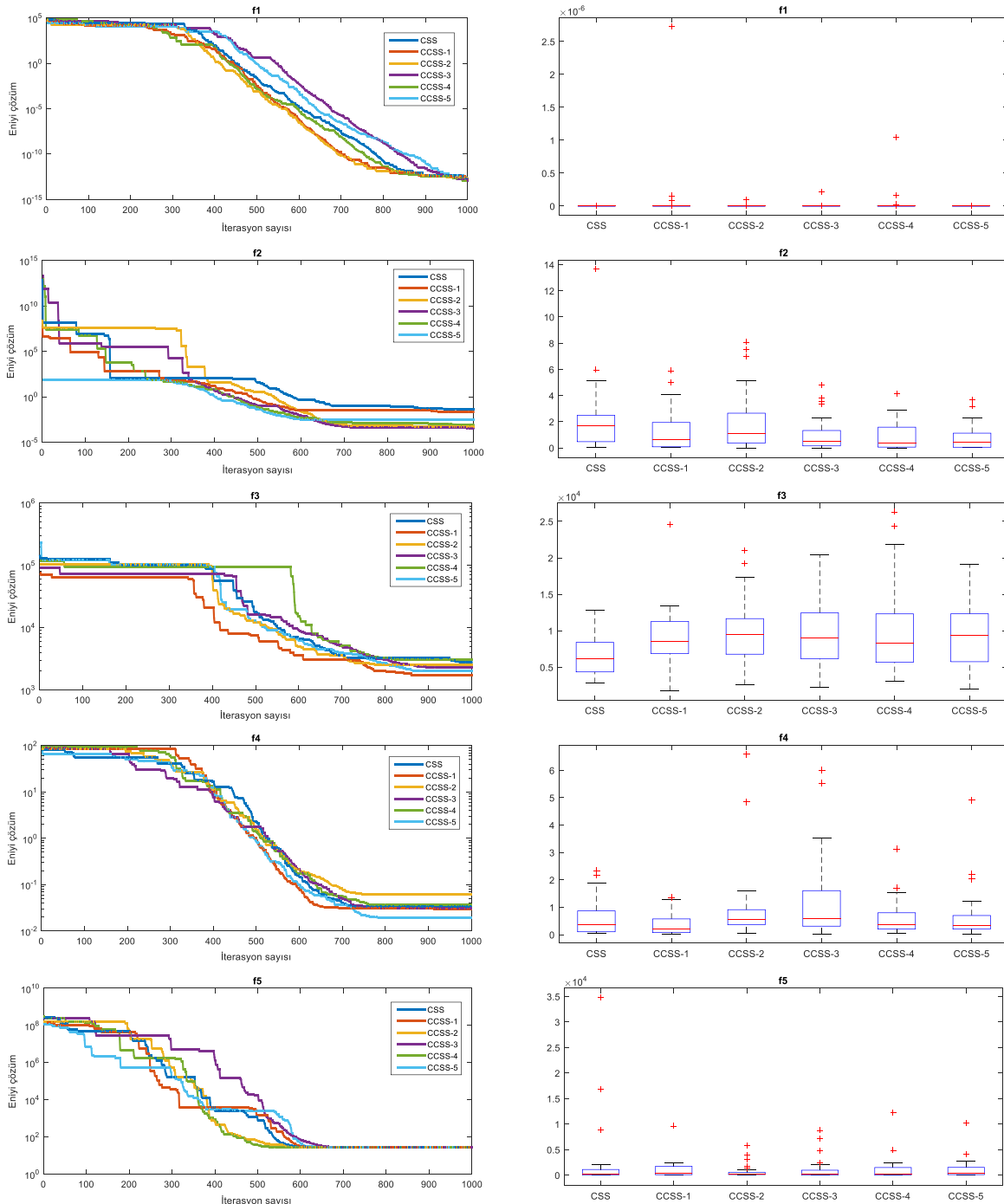


Fig. 7. The convergence curves and the boxplots belonging to  $f_1$ - $f_5$  functions

## 5. CONCLUSIONS

In the study CCSS algorithms have been developed integrating chaotic structures with five different chaotic mapping methods to CSS algorithm. Five test functions have been solved with each approach. As is seen in CCSS algorithms with tables and graphics, in four of the test functions better solutions than CSS have been obtained. The new algorithms, namely CCSS, developed as a result of our studies, have obtained better values in most of the test functions in terms of velocity and decisiveness compared with CSS algorithm, in accordance with the aim of the study. As a result of the comparisons among themselves, it has been reached that CCSS-3 algorithm is more successful in exceeding local minimum points compared with the other algorithms and CCSS-5 algorithm has a more decisive structure. Two different aims come forward in the study; the first is the betterment of the algorithm with the integration of chaotic number generators to the search process done by CSS algorithm, and the other is the defining of the better ones among five different chaotic mapping methods handled in the study. In the following studies, it should be known that optimization algorithms with different structures match with chaotic processes with different structures, and a good evaluation of the handled purpose function, and the selection of the most suitable chaotic mapping method for the function contributes to the optimization process in a positive way.

## ACKNOWLEDGEMENT

This work has been presented and chosen as “selected paper” for publishing in Scientific Journal of Mehmet Akif Ersoy University (Techno-Science) in 2nd International Conference on Technology and Science (Techno-Science 2019) which held on 14-16 November 2019 in Burdur, Turkey.

## REFERENCES

- [1]. Özyön, S, Yaşar, C, Temurtaş, H. Incremental gravitational search algorithm for high-dimensional benchmark functions. *Neural Computing and Applications* 2019, 31(8), 3779-3803.
- [2]. Peitgen, H, Jurgens, H, Saupe, D. *Chaos and Fractals: New frontiers of science*. Berlin, Springer-Verlag, 1992.
- [3]. Durmuş, B, Özyön, S, Temurtaş, H. Gravitational search algorithm with chaotic map (GSA-CM) for solving optimization problems. *International Journal of Research in Engineering and Technology* 2016, 5(2), 204-212.
- [4]. Saremi, S, Mirjalili, S, Lewis, A. Biogeography-based optimisation with chaos. *Neural Computing and Applications* 2014; 25(5), 1077-1097.
- [5]. Mitic, M, Vukovic, N, Petrovic, M, Miljkovic, Z. Chaotic fruit fly optimization algorithm. *Knowledge-Based Systems* 2015, 89, 446-458.
- [6]. Gandomi, AH, Yang, XS. Chaotic bat algorithm. *Journal of Computational Science* 2014, 5(2), 224-232.
- [7]. Wang, GG, Guo, L, Gandomi, AH, Hao, GS, Wang, H. Chaotic krill herd algorithm. *Information Sciences* 2014, 274, 17-34.
- [8]. Alataş, B. Chaotic harmony search algorithms. *Applied Mathematics and Computation* 2016, 216(9), 2687-2699.
- [9]. Alataş, B, Akın, E, Bedri, O. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons and Fractals* 2009, 40(4), 1715-1734.
- [10]. Chaoshun, L, Jianzhong, Z, Jian, X, Han, X. Parameters identification of chaotic system by chaotic gravitational search algorithm. *Chaos, Solitons and Fractals* 2012, 45(4), 539-547.
- [11]. Ozer, AB. CIDE: Chaotically initialized differential evolution. *Expert Systems with Applications* 2010, 37(6), 4632-4641.
- [12]. Jordehi, AR. Chaotic bat swarm optimisation (CBSO). *Applied Soft Computing* 2015, 26, 523-530.
- [13]. Alataş, B. Chaotic bee colony algorithms for global numerical optimization. *Expert Systems with Applications* 2010, 37(8), 5682-5687.
- [14]. Kaveh, A, Talahatari, S. A novel heuristic optimization method: charged system search. *Acta Mechanica* 2010, 213(3-4), 267-289.
- [15]. Özyön, S, Temurtaş, H, Durmuş, B, Kuvat, G. Charged system search algorithm for emission constrained economic power dispatch problem. *Energy* 2012, 46(1), 420-430.
- [16]. Özyön, S, Durmuş, B, Yaşar, C, Temurtaş, H, Kuvat, G. Solution to non-convex economic power dispatch problems with generator constraints by charged system search algorithm. *International Review of Electrical Engineering (IREE)* 2012, 7(5), 5840-5853.
- [17]. Mahdi, A, Jawad, AK, Hreshee, SS. Digital chaotic scrambling of voice based on Duffing Map. *International Journal of Information and Communication Sciences* 2016, 1(2), 16-21.
- [18]. Bucolo, M, Caponetto, R, Fortuna, L, Frasca, M, Rizzo, A. Does chaos work better than noise?. *IEEE Circuits and Systems Magazine* 2002, 2(3), 4-19.
- [19]. Henon, M. A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics* 1976, 50, 69-77.
- [20]. Caponetto, R, Fortuna, L, Fazzino, S, Xibilia, MG. Chaotic sequences to improve the performance of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* 2003, 7(3), 289-304.
- [21]. Aboites, V, Liceaga, D, Kir'yanov, A, Wilson, M. Ikeda Map and phase conjugated ring resonator chaotic dynamics. *Applied Mathematics & Information Sciences* 2016, 10(6), 1-6.

