GRAVITATIONAL SEARCH ALGORITHM WITH CHAOTIC MAP (GSA-CM) FOR SOLVING OPTIMIZATION PROBLEMS

Burhanettin DURMUŞ[1], Serdar ÖZYÖN[2,*], Hasan TEMURTAŞ[3]

[1]*Dumlupınar University, Department of Electrical and Electronics Engineering, 43100, Kütahya, TURKEY*
[2,*]*Dumlupınar University, Department of Electrical and Electronics Engineering, 43100, Kütahya, TURKEY*
[3]*Dumlupınar University, Department of Computer Engineering, 43100, Kütahya, TURKEY*
**Corresponding author. Tel: +90 (274) 265 2031 / 4264; Fax: +90 (274) 265 2066*
E-mail address: ***serdar.ozyon@dpu.edu.tr*** *(Serdar Özyön)*

Abstract

Gravitational Search Algorithm (GSA) is a newly heuristic algorithm inspired by nature which utilizes Newtonian gravity law and mass interactions. It has captured much attention since it has provided higher performance in solving various optimization problems. This study hybridizes the GSA and chaotic equations. Ten chaotic-based GSA (GSA-CM) methods, which define the random selections by different chaotic maps, have been developed. The proposed methods have been applied to the minimization of benchmark problems and the results have been compared. The obtained numeric results show that most of the proposed algorithms have increased the performance of GSA and have developed its quality of solution.

Keywords: *Computational Intelligence, Evolutionary Computation, Heuristic Algorithms, Chaotic Maps, Optimization Methods.*

----------***----------

1. INTRODUCTION

In physics, massive objects tend to accelerate towards each other. In Newton's law of gravitation, each object attracts one another with a particular force, which is "gravitational force". GSA is a heuristic algorithm which utilizes Newtonian gravity law and mass interactions [1]. It has captured much attention. And it has been admitted this algorithm provides higher performance in solving various problems [1, 2]. In GSA, a number of agents referred to as masses are defined in order to find the optimal solution by simulations of Newtonian laws. The direction of the agent is calculated according to the total force gained by all other agents. The force is proportionate with fitness value. The meaning of a heavier mass is a more efficient agent, namely better agents possess higher attractions and walk more slowly.

As with other heuristic algorithms also in GSA, in different phases of evolution random alterations and selections are needed [3-9]. Generally, this randomness is provided with random equations with a long period.

In these days, the notion of employing chaotic systems as a substitution of random processes has been gained acception in optimization studies. The chaotic dynamics can play the role of the randomness in optimization algorithms. The benefits of employing chaotic signals as substitution of random signals are shown in empirical studies [10-12]. Thus, the employment of defining chaotic signals as substitution of random sequences may be a potential way to improve the performance of GSA.

This paper proposes Gravitational Search Algorithm with Chaotic Map (GSA-CM) methods to solve the global optimization problems. The GSA-CM algorithms utilizing different chaotic systems replace random numbers for GSA's different parameters. Thus, different methods employing chaotic maps as efficient substitutions for pseudorandom sequences have been suggested. Some benchmark problems have been studied in order to assess these algorithms. The results display the development of the new algorithm owing to the employment of the deterministic chaotic signals instead of random sequences.

2. GRAVITATIONAL SEARCH ALGORITHM

GSA has been inspired by the Newtonian laws is one of the newest heuristic algorithms [1]. According to the gravity law, each particle draws the other particles with a force. Similarly, in GSA, agents are a group of masses that interact with each other due to the laws of gravity and motion. The performance of agents is measured by their masses. This structure causes global motion of all of the agents towards the heavier agents.

Considering the N agents (masses) in a system, position of i_{th} agent is give as follows:

$$X_i = \left(x_i \ldots\ldots x_i^d \ldots\ldots x_i^n\right) \text{ for } i = 1, 2, \ldots\ldots, N. \quad (1)$$

where x_i^d is position of the i_{th} agent with dimension d and n which is the dimension of the search space.

In respect of GSA, the force from the j_{th} agent proceeding action the i_{th} agent is described as in the following:

$$F_{ij}^{d}(t) = G(t)\frac{M_i(t)\times M_j(t)}{\left\|x_i(t), x_j(t)\right\| + \varepsilon}\left(x_j^d(t) - x_i^d(t)\right) \quad (2)$$

Where $M_i(t)$ is mass of i_{th} agent, $M_i(t)$ is mass of j_{th} agent, $x_i^d(t)$ is position of i_{th} agent in the d dimension, $x_j^d(t)$ is position of j_{th} agent in the d dimension, $G(t)$ is gravitational constant at time t, ε is a small constant, and $\left\|x_i(t), x_j(t)\right\|$ is Euclidean distance between two agents i_{th} and j_{th}. In order to give stochastic characteristic to GSA, it supposes that the force that acts on i_{th} agent in the d dimension be a randomly weighted sum of d_{th} components of the forces exerted from other agents [1]:

$$F_i^d(t) = \sum_{j\in kbest, j\neq i}^{N} rand_j F_{ij}^d(t) \quad (3)$$

In GSA, the masses are calculated by fitness function. The better fitness value of agent has heavier mass. The masses are updated by the following equations:

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (4)$$

$$i = 1,2,.....N \ (population\ number)$$

$$m_i(t) = \frac{f_i(t) - worst(t)}{best(t) - worst(t)} \quad (5)$$

where $f_i(t)$ is fitness value of i_{th} agent at time t, best(t) is fitness value of the best agent at time t, worst(t) is fitness value of the worst agent at time t, $m_i(t)$ and $m_j(t)$ inertia masses of agents i_{th} and j_{th}, respectively. Furthermore, acceleration of each agent is computed using the law of motion. The acceleration of the i_{th} agent at time t and in direction d_{th} is given as in the following:

$$a_i^d(t) = \frac{F_i^d(t)}{M_i(t)} \quad (6)$$

Taken together, next velocity of an agent is computed as a fraction of its recent velocity added to its acceleration (7). After that, the next position of the agent can be computed employing (8):

$$V_i^d(t+1) = rand_i V_i^d(t) + a_i^d(t) \quad (7)$$

$$x_i^d(t+1) = x_i^d(t) + V_i^d(t+1) \quad (8)$$

The gravitational constant, $G(t)$ given in (9) is important paradigm for performance of GSA. It is initialized at the beginning, after that it will be decreased with time. $G(t)$ is defined as follows:

$$G(t) = G_0 e^{-\beta\left(\frac{t}{T}\right)} \quad (9)$$

where G_0 is initial value of $G(t)$, T is total iteration number, t is current iteration number and β is a constant.

3. GRAVITATIONAL SEARCH ALGORITHM WITH CHAOTIC MAP

Recently, the constant increase in interest to use the chaotic equations instead of random process in different fields encourages the chaos studies also in optimization algorithms [10, 13, 14]. The random based optimization algorithms need, by their nature, random sequences with a long period in most of the phases of evolution [15]. On the other hand, chaotic sequences provide easiness simplicity and rapidity. Therefore, randomness role can be provided with (obtained by) chaos sequences. Moreover, compared with the traditional algorithms, the employment of the chaotic sequences in stochastic algorithms can be beneficial to escape more easily from local minima. The advantages of the employment of chaotic signals as substitution of random signals are shown in empirical studies [16-19]. Hence, in this study, the evolution of chaotic based GSA, which defines the random selections by chaotic maps to develop the performance of GSA, has been focused on.

In standard GSA, in the assignment of initial positions of agents, in the phase of weighted sum of the forces and velocity update random are needed. In this study, the proposed GSA-CM algorithms utilize chaotic systems-generated sequences as substitution of random numbers when random selection is required. The chaotic maps used in numeric calculations are given below:

3.1 Logistic Map

The logistic map takes places of chaotic behaviors of biological population [20]. It is represented by:

$$X_{k+1} = aX_k(1 - X_k) \quad (10)$$

where X_k is the k_{th} chaotic number, k is iteration number. In the experiments a=4 is used.

3.2 Tent Map

This map seems to logistic map [21]. It is defined as follows:

$$X_{k+1} = \begin{cases} X_k / 0.7 & X_k < 0.7 \\ 10/3 X_k (1 - X_k) & otherwise \end{cases} \quad (11)$$

3.3 Gauss Map

The Gauss map [21] is represented by:

$$X_{k+1} = \begin{cases} X_k / 0.7 & X_k = 0 \\ 1/X_k \bmod(1) & X_k \in (0,1) \end{cases} \quad (12)$$

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

It is used for testing purpose in the literature.

3.4 Circle Map

The circle map [22] is defined as follows:

$$X_{k+1} = X_k + b - (a/2\pi)\sin(2\pi X_k) \quad (13)$$

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

3.5 Liebovtech Map

Liebovtech map is formed of three linear segments [23]. It is defined by the following equations:

$$X_{k+1} = \begin{cases} a_1 X_k & 0 < X_k \le d_1, \\ \frac{d_2 - X_k}{d_2 - d_1} & d_1 < X_k \le d_2, \\ 1 - a_2 (1 - X_k) & d_2 < X_k \le 1 \end{cases} \quad (14)$$

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

$$a_1 = \frac{d_2}{d_1}(1 - (d_2 - d_1)),\ a_2 = \frac{1}{d_2 - 1}((d_2 - 1) - d_1(d_2 - d_1))$$

3.6 Zaslavskii Map

The Zaslavskii map has spread-spectrum characteristic and large Lyapunov exponent [24]. It is defined by:

$$X_{k+1} = (X_k + v + aY_{k+1}) \bmod(1)$$
$$Y_{k+1} = \cos(2\pi X_k) + e^{-r} Y_k \quad (15)$$

3.7 Arnold's Cat Map

The Arnold Cat Map (ACM) was discovered by Vladimir Arnold, and it utilizes the image of a cat while working on it [25]. Mathematically the ACM is represented as the following [26]:

$$\begin{bmatrix} X_{k+1} \\ Y_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} X_k \\ Y_k \end{bmatrix} (\text{mod}1) \quad (16)$$

where $X_k, Y_k \in [0,1)$.

3.8 Henon Map

This map is a nonlinear 2-dimensioanan map employed for testing purpose [27]. It is defined as follows:

$$X_{k+1} = 1 - a(X_k)^2 + Y_k$$
$$Y_{k+1} = bX_k \quad (17)$$

The suggested parameter values are a=1.4 and b=0.3 [27].

3.9 Kaplan-Yorke Map

This map was used by Kaplan and Yorke introducing the idea of "Lyapunov dimension" [28]. It is conjectured that for an attractor of a typical dynamical system in a Euclidean space. The Kaplan-Yorke Map is defined as

$$X_{k+1} = aX_k \bmod(1)$$
$$Y_{k+1} = bY_k + \cos(4\pi X_k) \quad (18)$$

This is bounded for $0 \le a \le 2$ and $0 \le b \le 1$ [29].

3.10 Lozi Map

This map is a simplification of the Henon map [27]. It is given in the following equations:

$$X_{k+1} = 1 - a|X_k| + Y_k$$
$$Y_{k+1} = bX_k \quad (19)$$

The parameters used in this study are a=1.7 and b=0.5 as suggested in [13].

In the standard GSA, positions of agents (masses) are determined with random values. The derived GSA-CM algorithms make this definition in a chaotic way. The initial positions of the agents are formed by using the chaotic maps defined above.

As it has been shown in (3), in standard GSA, in the calculation of the weighted sum of the forces, which acts on each agent, random selection is needed. The derived GSA-CM algorithms define this random selection by chaotic maps. If (3) is modified again, it is defined as in the following:

$$F_i^d(t) = \sum_{j \in kbest, j \ne i}^{N} X_j F_{ij}^d(t) \quad (20)$$

where X_j is a chaotic variable based on selected map.

Lastly, in the velocity update of the agents given in (7), random generation is defined by using the chosen chaotic maps. If (7) is modified for the calculation of the next velocity vector of an agent, it is defined as in the following:

$$V_i^d(t+1) = X_i V_i^d(t) + a_i^d(t) \quad (21)$$

The new position of the agent is defined by adding the obtained velocity to previous position value.

4. TABLES, FIGURES, EQUATIONS

This section provides an illustrative example to verify the efficiency of the proposed GSA-CM algorithm for solving optimization problems. A number of benchmark functions chosen from Back and Schwefel [30], Törn and Zilinskas [31], Leung and Wang [32], Yao et al. [33] are optimized using GSA-CM and compared with classical GSA in order to verify the efficiency of GSA-CM. These functions are widely used for measuring the performance of the optimization methods [1, 5, 7, 9]. The description of the test functions is provided in Table 1.

For the GSA-CM using different chaotic maps, the numeric results belonging to *F1* and *F2* have been shown in Tables 2 and 3 and the success rates have been shown in Tables 4 and 5.

Table -1: Benchmark Functions

Test Functions	Function Name	S	Global Optimum
$F1 = \sum_{i=1}^{n} (x_i)^2$	Sphere	$[-100,100]^n$	0.0
$F2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	Schwefel's problem 1.2	$[-100,100]^n$	0.0
$F3 = \sum_{i=1}^{n} \left[x_i^2 - 10Cos(2\pi x_i) + 10 \right]$	Rastrigin	$[-5.12,5.12]^n$	0.0
$F4 = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos 2\pi x_i \right) + 20 + e$	Ackley	$[-32,32]^n$	0.0
$F5 = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $*[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Branin 2	$[-2,2]^n$	3.0
$F6 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-Hump Camel-Back	$[-5,5]^n$	-1.031628

Table -2: Numerical Results of the Standard GSA and GSA-CM's for **F1**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	1.08504e-008	1.55138e-008	2.32555e-008	4.76666e-009
Logistic	1.27586e-009	1.3817e-008	4.05096e-008	1.56888e-008
Tent	5.88335e-009	2.79714e-008	5.36184e-008	2.11517e-008
Gauss	5.4202e-010	9.15599e-009	2.39934e-008	9.65906e-009
Circle	1.08009e-009	**5.99466e-009**	1.44881e-008	**5.85688e-009**
Liebovtech	1.88308e-009	4.93247e-008	9.8853e-008	3.60687e-008
Zaslavskii	1.41438e-008	2.65609e-008	4.6166e-008	1.22976e-008
Arnold's cat	6.00897e-009	1.47541e-008	2.87475e-008	9.49585e-009
Henon	1.57695e-009	2.80361e-008	8.39646e-008	3.29036e-008
Kaplan-Yorke	**6.58545e-011**	2.9106e-008	9.83325e-008	4.01756e-008
Lozi	1.80762e-008	3.31738e-008	5.89339e-008	1.77055e-008

Table -3: Numerical Results of the Standard GSA and GSA-CM's for **F2**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	7.96574e-009	1.85204e-008	5.28681e-008	1.93108e-008
Logistic	2.92869e-009	2.87001e-008	6.56822e-008	2.41473e-008
Tent	2.79917e-009	8.66532e-009	2.28325e-008	8.15879e-009
Gauss	4.73325e-009	1.75748e-008	3.76001e-008	1.35134e-008
Circle	4.03783e-009	2.60412e-008	5.95176e-008	2.37868e-008
Liebovtech	1.58165e-009	**6.4012e-009**	1.52882e-008	5.83012e-009
Zaslavskii	1.09711e-008	1.98896e-008	3.82406e-008	1.09273e-008
Arnold's cat	4.60037e-009	1.32515e-008	3.73009e-008	1.36296e-008
Henon	7.96127e-009	1.58814e-008	2.89502e-008	8.49083e-009
Kaplan-Yorke	**3.84667e-010**	9.53969e-009	1.45353e-008	**5.76835e-009**
Lozi	1.41173e-008	4.41964e-008	1.09321e-007	4.11041e-008

In this table, global optimum is the optimum value of the function, and *S* is the search space. *F1* and *F2* functions are unimodal, *F3* and *F4* functions are multimodal having several local minima. Multimodal functions comprise the most difficult class of problems because the number of local minima increases exponentially with the problem dimension [31]. F5 and F6 functions are low-dimensional functions having only a few local minima.

Comparisons are performed over 30 runs in simulation studies, and the best solutions of the 30 independent runs are summarized in Tables 2-13. For each method, the Best, Mean and Standard Deviation (Std. Dev.) are calculated from the simulated runs. Another criterion defining the performance of the algorithms is the success rate. Here, the success rate (*SR*) can be defined as in the following:

$$SR = 100 \times \frac{N_{successfull}}{N_{all}}|_{Q_{level}} \quad (22)$$

where $N_{successful}$ is the number of trials, which found the successful solution in the allowable maximum iteration. N_{all} is the number of all trials. Q_{level} is an acceptance tolerance.

In all cases, the population size N, is set to 30. The number of variables is set to two. For GSA and GSA-CM, system parameters G_0 and β are set to 100 and 10, respectively. The number of iterations is considered to be 500.

According to the numeric results, it is seen that both GSA and GSA-CM using different chaotic maps have given similar results for unimodal functions. On the other hand, when the algorithms are evaluated according to success rate criterion, some proposed algorithms are seen to have been more successful. It can be said that the GSA-CM using Kaplan-Yorke map has given the best value in terms of success rate.

Table -4: Success Rates of the Standard GSA and GSA-CM's for **F1**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	81	15
Logistic	75	15
Tent	78	16
Gauss	91	24
Circle	90	25
Liebovtech	82	17
Zaslavskii	81	19
Arnold's cat	78	18
Henon	66	17
Kaplan-Yorke	**91**	**29**
Lozi	49	13

Table -5: Success Rates of the Standard GSA and GSA-CM's for **F2**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	63	12
Logistic	63	8
Tent	77	21
Gauss	80	27
Circle	71	15
Liebovtech	82	21
Zaslavskii	69	15
Arnold's cat	79	23
Henon	67	17
Kaplan-Yorke	**84**	**28**
Lozi	41	5

Table -6: Numerical Results of the Standard GSA and GSA-CM's for **F3**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	0.000953528	0.796295	0.995271	0.444609
Logistic	0.000148137	0.796246	1.9906	0.83261
Tent	5.78428e-005	0.199203	0.995007	0.444868
Gauss	0.000121711	**0.000544**	0.00136897	**0.000578**
Circle	5.07353e-005	0.398111	0.995078	0.544933
Liebovtech	0.000164111	0.39834	0.995473	0.545002
Zaslavskii	2.17192e-005	0.199191	0.994985	0.444862
Arnold's cat	8.71806e-006	0.199324	0.995359	0.444997
Henon	0.000219196	0.597304	0.995468	0.544842
Kaplan-Yorke	**7.82784e-007**	0.596978	1.98992	0.889921
Lozi	0.000222194	0.399055	0.996895	0.544972

Table -7: Numerical Results of the Standard GSA and GSA-CM's for **F4**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	0.000182623	0.000389226	0.000567831	0.000184589
Logistic	0.000220184	0.000375717	0.000521622	0.000118569
Tent	0.000211031	0.000364735	0.000522111	0.000141234
Gauss	0.00013141	0.000314793	0.00057734	0.000166245
Circle	0.000158603	0.000348678	0.000472937	0.000145746
Liebovtech	0.000146454	0.000311924	0.000710498	0.000248748
Zaslavskii	0.000141938	0.000327622	0.000737807	0.000251274
Arnold's cat	0.000177128	0.000356762	0.00049968	0.000147754
Henon	0.000159436	0.000460282	0.00109478	0.000384062
Kaplan-Yorke	**0.000126635**	**0.000281737**	0.000488754	**0.000118427**
Lozi	0.000152235	0.00054905	0.00094647	0.000356016

Table -8: Success Rates of the Standard GSA and GSA-CM's for **F3**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	9	1
Logistic	20	3
Tent	57	12
Gauss	67	6
Circle	47	11
Liebovtech	47	3
Zaslavskii	56	**13**
Arnold's cat	51	7
Henon	23	3
Kaplan-Yorke	**68**	9
Lozi	17	2

Table -9: Success Rates of the Standard GSA and GSA-CM's for **F4**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	100	16
Logistic	100	21
Tent	100	21
Gauss	100	26
Circle	100	17
Liebovtech	100	23
Zaslavskii	100	22
Arnold's cat	100	22
Henon	100	16
Kaplan-Yorke	100	**29**
Lozi	100	21

Table -10: Numerical Results of the Standard GSA and GSA-CM's for **F5**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	3.0	3.00001	3.00158	3.82537e-006
Logistic	3.0	3.00001	3.00121	9.49787e-006
Tent	3.0	3.00001	3.0029	4.6935e-006
Gauss	3.0	3.00001	3.00117	2.83164e-006
Circle	3.0	3.00001	3.00158	1.39615e-005
Liebovtech	3.0	3.00001	3.00145	9.47017e-006
Zaslavskii	3.0	3.00001	3.00244	3.3804e-006
Arnold's cat	3.0	3.00001	3.00122	2.69802e-006
Henon	3.0	3.00001	3.00192	6.67969e-006
Kaplan-Yorke	3.0	3.00001	3.00147	**2.3994e-006**
Lozi	3.0	3.00002	3.01787	2.17198e-005

Table -11: Numerical Results of the Standard GSA and GSA-CM's for **F6**

Algorithms	Best	Mean	Worst	Std. Dev.
Standard GSA	-1.03163	-1.03163	-1.03157	2.81315e-007
Logistic	-1.03163	-1.03163	-1.0316	2.68685e-007
Tent	-1.03163	-1.03163	-1.03159	1.76981e-007
Gauss	-1.03163	-1.03163	-1.03161	9.67557e-008
Circle	-1.03163	-1.03163	-1.0316	1.44294e-007
Liebovtech	-1.03163	-1.03163	-1.03161	9.84037e-008
Zaslavskii	-1.03163	-1.03163	-1.0316	9.68491e-008
Arnold's cat	-1.03163	-1.03163	-1.03161	1.55772e-007
Henon	-1.03163	-1.03163	-1.03157	2.18973e-007
Kaplan-Yorke	-1.03163	-1.03163	-1.03162	**8.50546e-008**
Lozi	-1.03163	-1.03163	-1.03136	1.48907e-006

Tables 10-13 show the numeric results obtained for *F5* and *F6* functions. All algorithms have succeeded to reach the global optimum.

Table -12: Success Rates of the Standard GSA and GSA-CM's for **F5**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	36	7
Logistic	41	5
Tent	57	5
Gauss	58	8
Circle	42	5
Liebovtech	47	5
Zaslavskii	49	4
Arnold's cat	51	7
Henon	37	4
Kaplan-Yorke	**66**	**10**
Lozi	27	5

Table -13: Success Rates of the Standard GSA and GSA-CM's for **F6**

Algorithms	$Q_{level} = 1e-6$	$Q_{level} = 1e-7$
Standard GSA	100	94
Logistic	100	93
Tent	100	94
Gauss	100	97
Circle	100	93
Liebovtech	100	95
Zaslavskii	100	96
Arnold's cat	100	94
Henon	100	86
Kaplan-Yorke	100	**99**
Lozi	97	56

The numeric results for the multimodal functions are given in Tables 6-9. Since these functions have many local minima, the solution of the problem becomes difficult. According to the obtained numeric results, the proposed

algorithms can be said to have been successful and to have developed the performance of the standard GSA.

According to Table 12, the success rate of the standard GSA for *F5* function is 36% for $Q_{level} = 1e-4$. The GSA-CM using Kaplan-Yorke map heads with the success rate of 66%. Similar results have been obtained also for *F6* function, GSA-CM algorithms using Kaplan-Yorke and Gauss map have been seen to have the highest success rate.

5. CONCLUSIONS

This study generates GSA-CM methods, which use different chaotic maps, in order to develop the performance of the standard GSA. Ten different chaotic maps by hybridizing with the GSA have been used for solving benchmark problems. The proposed algorithms have been compared both in terms of numeric results and success rates. The obtained results show that when random generation is done with chaotic maps in the algorithms where random selections are used, it has increased the performance of the algorithm, and the results show that chaotic maps like Kaplan-Yorke, Gauss and Zaslavskii can be best categorized as GSA-CM methods.

REFERENCES

[1] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, "GSA: a gravitational search algorithm," Information Sciences, vol. 179, no. 13, pp. 2232–2248, 2009.

[2] C. Li, J. Zhou, "Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm," Energy Conversion and Management, vol. 52, no. 1, pp. 374-381, 2011.

[3] J. Kennedy, R. C. Eberhart, "Particle swarm optimization," in Proc. IEEE 1995 Neural Networks Conference, Perth, WA, Australia, 1995, pp. 1942-1948.

[4] M. Dorigo, V. Maniezzo, A. Colorni, "The ant system: optimization by a colony of cooperating agents", IEEE Transaction on Systems, Man, and Cybernetics-Part B, vol. 26, pp. 1–13, 1996.

[5] J. T. Tsai, J. H. Chou, T. K. Liu, "Optimal design of digital IIR filters by using hybrid taguchi genetic algorithm," IEEE Transactions on Industrial Electronics, vol. 53, no. 3, pp. 867-879, 2006.

[6] W. D. Chang, "Nonlinear system identification and control using a real coded genetic algorithm," Applied Mathematical Modeling, vol. 31, pp. 541–550, 2007.

[7] L. Zhao, F. Qian, Y. Yang, Y. Zeng, H. Su, "Automatically extracting T-S fuzzy models using cooperative random learning particle swarm optimization," Applied Soft Computing, vol. 10, no. 3, pp. 938-944, 2010.

[8] M. Mahdavi, M. Fesanghary, E. Damangir, "An improved harmony search algorithm for solving optimization problems," Applied Mathematics and Computation, vol. 188, pp. 1567–1579, 2007.

[9] B. Akay, D. Karaboğa, "A modified artificial bee colony algorithm for real-parameter optimization," Information Sciences, vol. 192, pp. 120-142, 2012.

[10] L. S. Coelho, V. C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect," IEEE Transactions on Power Systems, vol. 21, no. 2, pp. 989-996, 2006.

[11] B Alataş, "Chaotic harmony search algorithms," Applied Mathematics and Computation, vol. 216, pp. 2687-2699, 2010.

[12] S. Talatahari, A. Kaveh, R. Sheikholeslami, "An Efficient charged system search using chaos for global optimization problems," International Journal of Optimization in Civil Engineering, vol. 2, pp. 305-325, 2011.

[13] R. Caponetto, L. Fortuna, S. Fazzino, M. G. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms," IEEE Transactions on Evolutionary Computation, vol. 7, no. 3, pp. 289–304, 2003.

[14] M. S. Tavazoei, M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," Applied Mathematics and Computation, vol. 187, no. 2, pp. 1076-1085, 2007.

[15] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," Expert Systems with Applications, vol. 37, no. 8, pp. 5682-5687, 2010.

[16] H. Gao, Y. Zhang, S. Liang, D. A. Li, "New chaotic algorithm for image encryption," Chaos Solitons Fractals, vol. 29, pp. 393–399, 2006.

[17] B. Li, W. Jiang, "Optimizing complex functions by chaos search," Cybernetics and Systems, vol. 29, no. 4, pp. 409–419, 1998.

[18] L. S. Coelho, V. C. Mariani, "Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization," Expert Systems with Applications, vol. 34, pp. 1905–1913, 2008.

[19] B. Alatas, E. Akin, O. Bedri, "Chaos embedded particle swarm optimization algorithms," Chaos Soliton Fractals, vol. 40, pp. 1715-1734, 2009.

[20] R. M. May, "Simple mathematical models with very complicated dynamics," Nature, vol. 261, pp. 459-475, 1976.

[21] H. Peitgen, H. Jurgens, D. Saupe. Chaos and Fractals. Springer-Verlag, Berlin, 1992.

[22] W. M. Zheng, "Kneading plane of the circle map," Chaos Solitons Fractals, vol. 4, no. 7, pp. 1221-1233, 1994.

[23] S. L. Lievbovitch, T. I. Thot, "A model of ion channel kinetics using deterministic chaotic rather than stochastic processes," Journal of Theoretical Biology, vol. 148, pp. 243–267, 1991.

[24] G. M. Zaslavskii, "The simplest case of a strange attractor," Physics Letters A, vol. 69, no. 3, pp. 145–147, 1978.

[25] V. I. Arnold, A. Avez. Ergodic Problems of Classical Mechanics. 1st ed. Benjamin, New York, 1968.

[26] A. Konso, M. Ghebleh,"A novel image encryption algorithm based on a 3D chaotic map," Communications in Nonlinear Science and Numerical Simulation, vol. 17, pp. 2943-2959, 2012.

[27] M. Henon, "A two-dimensional mapping with a strange attractor," Communications in Mathematical Physics, vol. 50, pp. 69–77, 1976.
[28] J. L. Kaplan, J. A. Yorke. "Chaotic behavior of multidimensional difference equations," Lecture Notes in Mathematics, vol. 730, pp. 204–227, 1979.
[29] N. Singh, A. Sinha, "Optical image encryption using fractional fourier transform and chaos," Optics and Lasers in Engineering, vol. 46, pp. 117-123, 2008.
[30] T. Back, H. P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," Evolution Computation, vol. 1, no. 1, pp. 1-23, 1993.
[31] A. Törn, A. Zilinskas, Global Optimization. Springer-Verlag, Berlin, 1989.
[32] Y. W. Leung, Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," IEEE Transactions on Evolutionary Computation, vol. 5, no. 1, pp. 41-53, 2001.
[33] X. Yao, Y. Liu, G. Lin, "Evolutionary programming made faster," IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 82-102, 1999.

BIOGRAPHIES

B. Durmuş was born in Pazaryeri, Turkey, in 1978. He received the B.Sc. and M.Sc. degrees in 2000 and 2003 from the Department of Electrical Electronics Engineering, Dumlupınar University, Kütahya, Turkey respectively. He received Ph.D. degree from Sakarya University. He is working as Assistance Professor in Department of Electrical Electronics Engineering, Dumlupınar University. His areas of research include optimization techniques and computational intelligence.
E-mail : **burhanettin.durmus@dpu.edu.tr**

S. Özyön was born in Ayaş, Turkey, in 1981. He received the B.Sc. degree in electrical electronics engineering from Dumlupınar University, Kütahya, Turkey, in 2005 and M.Sc. degree from the Department of Electrical Electronics Engineering, Dumlupınar University, Kütahya, Turkey in 2009. He is working as research assistant in Department of Electrical Electronics Engineering, Dumlupınar University. His areas of research include analysis of power systems, economic operation of power systems, power distribution systems, renewable energy systems and optimization techniques.
E-mail : **serdar.ozyon@dpu.edu.tr**

H. Temurtaş was born in Mersin, Turkey, in 1967. He received the B.Sc. degree in electrical electronics engineering from Middle East Technical University, Ankara, Turkey in 1993 and M.Sc. degree from the Department of Electrical Electronics Engineering, Dumlupınar University, Kütahya, Turkey in 1996, Ph.D. degree in Electrical and Electronic Engineering from Sakarya University, Sakarya, Turkey in 2004. He is working as Assistance Professor in Department of Computer Engineering, Dumlupınar University. His areas of research include programming languages, control algorithms and optimization techniques.
E-mail : **hasan.temurtas@dpu.edu.tr**