Chaotic Moth Swarm Algorithm

Ugur Guvenc, Serhat Duman, and Yunus Hınıslıoglu
Department of Electrical and Electronics Engineering
Duzce University
Duzce, Turkey
Emails: {ugurguvenc, serhatduman, yunushinislioglu} @duzce.edu.tr

Abstract—**Moth Swarm Algorithm (MSA) is one of the newest developed nature-inspired heuristics for optimization problem. Nevertheless MSA has a drawback which is slow convergence. Chaos is incorporated into MSA to eliminate this drawback. In this paper, ten chaotic maps have been embedded into MSA to find the best numbers of prospectors for increase the exploitation of the best promising solutions. The proposed method is applied to solve the well-known seven benchmark test functions. Simulation results show that chaotic maps can improve the performance of the original MSA in terms of the convergence speed. At the same time, sinusoidal map is the best map for improving the performance of MSA significantly.**

Keywords—chaotic maps; optimization; moth swarm algorithm

I. INTRODUCTION

Optimization is a method which finds the best solution/solutions for a specific problem. The real aim for solving an optimization problem is finding the global (best) solution without stuck into local solutions. However, when the problem becomes more complex, it is hard to avoid local solutions. To handle this situation, new optimization techniques are being developed [1]. There are several methods that have been used to solve optimization problems. Particle Swarm Optimization (PSO) [2], Artificial Bee Colony (ABC) algorithm [3-6], Ant Colony Optimization (ACO) [7], Moth Flame Optimization algorithm [1] etc. are the nature based algorithms that have been developed. Besides, there are also combined algorithms which enhance the performance of the algorithms such as PSO-local search [8-10], Hybrid ABC [11-13] etc. Nowadays, optimization algorithms are still being improved to solve problems. At 2016, Mohammed et al. introduced a new technique called Moth Swarm Algorithm (MSA) which is based on a group of moths' behavior for orientation and searching food in the dark [14]. Moths decide which light source is going to be the reference by using the light intensity. They originally use the moon as a light source to find their path. However if there is an artificial light source involved there is going to be a useless motion around the artificial light source. In this study, we embedded chaotic maps into MSA's transverse orientation step and the results are more promising respect to Moth Swarm Algorithm.

This proposed method has been used to solve the seven benchmark test functions. It can be seen from the simulation results that all ten chaotic maps can speed the convergence with respect to the original MSA. Besides, sinusoidal map gives the best promising results for improving the performance of MSA.

II. MOTH SWARM ALGORITHM

Moths and butterflies have a large population among all insects. They are nocturnal animals which hide from predator animals during daylight and they fly and search food in the night. They basically fly in a direct path to maintain the angle with a celestial far-distant light source e.g., moonlight. Unfortunately, they cannot separate a natural light source from an artificial one and since the human made artificial light source is closer than the moon, they start to orientate around the artificial source with a spiral shape path [14].

In previously proposed algorithm, the position of the light source is the solution of the optimization problem and the light intensity of the source is the fitness of the solution. Three groups of moths are used to model the algorithm [14].

First group is called pathfinders. They are a small group in the population which can explore the new areas in the optimization space by using the principle of First In-Last Out. They find the best position of the light source among the other sources and lead other members in the population to this position [14].

Second one is called prospectors which ramble into a random spiral path around the light source that has been marked by the pathfinders.

The last group is called onlookers. They fly to the best solution in pursuant of the prospectors.

In Moth Swarm Algorithm, each moth is being charged with the light intensity of its corresponding source. The best fitnesses are assigned to the position of the pathfinders and the second and last fitnesses are stationed to the prospectors and onlookers respectively [14].

In the proposed algorithm, there are four main phases which are being processed.

In the first one, called Initialization phase, the positions of each moth are randomly created as (1) for d-dimensional problem and n number of population.

$$\begin{aligned} x_{ij} &= rand[0,1]\cdot\left(x_j^{\max} - x_j^{\min}\right) + x_j^{\min} \\ &\forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,d\} \end{aligned} \tag{1}$$

where $x_j^{\max}$ and $x_j^{\min}$ are the upper and lower limits, respectively.

Henceforward, calculated fitnesses are sorted best to worst. The best fitnesses are chosen to be pathfinders and the second best and worst fitnesses are chosen to be the prospectors and onlookers, respectively.

In the second phase, called Reconnaissance Phase, the positions of the pathfinders are updated in five steps.

In proposed diversity index for crossover points step, first, for *t* iteration, the normalized dispersal degree σ_j^t of individuals in the jth dimension is measured as follows:

$$\sigma_j^t = \frac{\sqrt{\frac{1}{n_p}\sum_{i=1}^{n_p}\left(x_{ij}^t - \overline{x_j^t}\right)^2}}{\overline{x_j^t}} \tag{2}$$

where $\overline{x_j^t} = \frac{1}{n_p}\sum_{i=1}^{n_p} x_{ij}^t$, and n_p is the number of pathfinders moths. Then, variation coefficient may be formulated as follows:

$$\mu^t = \frac{1}{d}\sum_{j=1}^{d}\sigma_j^t \tag{3}$$

where, μ^t called variation coefficient, is a measure for relative dispersion.

In the second step, Lévy flights, the property of the Lévy flight which is the ability to travel over large scale distances using different size of steps based on α-stable distribution is used. In the third step, called difference vectors Lévy mutation, the previously proposed algorithm creates the sub-trail vector $\overrightarrow{v_p}$ with using the host vector $\overrightarrow{x_p}$ and donor vectors. In the fourth step, proposed adaptive crossover operation based on population diversity, basically, each pathfinder updates their positions. Finally, at Selection Strategy step, after the update process completed, the comparison process between the fitnesses begins (further information at [14]).

In the third phase, called Transverse Orientation Phase, prospector number is decreasing as follows:

$$n_f = round\left(\left(n - n_p\right)\times\left(1 - \frac{t}{T}\right)\right) \tag{4}$$

where t is the iteration number, T is total number of iteration, is the number of prospectors and n_p is the number of pathfinders. In the following section, we redesign the formula with chaotic maps.

III. Chaotic Maps For MSA

Recently, chaos, which is a kind of dynamic behavior of nonlinear systems, has aroused much concern in different fields of sciences such as chaos control, pattern recognition and optimization theory [15]. In this section, ten chaotic maps are used and listed as follows [16]:

A. Chebyshev [16]:

$$x_{i+1} = \cos\left(i\cos^{-1}(x_i)\right) \tag{5}$$

The range of this map lies in the interval of (-1,1)

B. Circle [16]:

$$x_{i+1} = \mathrm{mod}\left(x_i + b - \left(\frac{a}{2\pi}\right)\sin(2\pi x_i), 1\right) \qquad a = 0.5 \quad b = 0.2 \tag{6}$$

The range of this map lies in the interval of (0,1)

C. Gauss/mouse [16]:

$$x_{i+1} = \begin{cases} 1 & x_i = 0 \\ \frac{1}{\mathrm{mod}(x_i, 1)} & otherwise \end{cases} \tag{7}$$

The range of this map lies in the interval of (0,1)

D. Iterative [16]:

$$x_{i+1} = \sin\left(\frac{a\pi}{x_i}\right), \qquad \mathrm{a} = 0.7 \tag{8}$$

The range of this map lies in the interval of (-1,1)

E. Logistic [16]:

$$x_{i+1} = a x_i (1 - x_i) \qquad a = 4 \tag{9}$$

The range of this map lies in the interval of (0,1)

F. Piecewise [16]:

$$x_{i+1} = \begin{cases} \frac{x_i}{P} & 0 \le x_i \le P \\ \frac{x_i - P}{0.5 - P} & P \le x_i \le 0.5 \\ \frac{1 - P - x_i}{0.5 - P} & 0.5 \le x_i \le 1 - P \\ \frac{1 - x_i}{P} & 1 - P \le x_i \le 1 \end{cases} \qquad P = 0.4 \tag{10}$$

The range of this map lies in the interval of (0,1)

G. Sine [16]:

$$x_{i+1} = \frac{a}{4}\sin(\pi x_i) \qquad a = 4 \tag{11}$$

The range of this map lies in the interval of (0,1)

H. Singer [16]:

$$x_{i+1} = \mu\begin{pmatrix}7.68x_i - 23.31x_i^2 + \\ 28.75x_i^3 - 13.302875x_i^4\end{pmatrix} \quad \mu = 2.3 \qquad (12)$$

The range of this map lies in the interval of (0,1)

I. Sinusoidal [16]:

$$x_{i+1} = ax_i^2 \sin(\pi x_i) \qquad a = 2.3 \qquad (13)$$

The range of this map lies in the interval of (0,1)

J. Tent [16]:

$$x_{i+1} = \begin{cases} \dfrac{x_i}{0.7} & x_i < 0.7 \\ \dfrac{10}{3}(1 - x_i) & x_i \geq 0.7 \end{cases} \qquad (14)$$

The range of this map lies in the interval of (0,1)

It has to be noted that all chaotic maps start from $x_0 = 0.7$.

IV. Embedding Chaotic Maps Into MSA

In the transverse orientation step, the number of prospectors decreases as follows:

$$n_f = round\left((n - n_p) \times (1 - x)\right) \qquad x = \frac{t}{T} \qquad (15)$$

where t is the iteration number, T is number of total iteration, n_f is the number of prospectors and n_p is the number of pathfinders. The main purpose of the embedding process is to systematically change of the x parameter of MSA in (15) with embedding different chaotic maps so that we can have a good balance between exploration and exploitation in course of iteration. In MSA algorithm, the x is linearly decreased along with the iteration number. It means that the algorithm either explores or exploits. The value of x changes chaotically while it decreases during iteration so that it can also have exploration in the final steps of iterations.

In ref [16], the range value of normalization is defined by the author as follows:

$$V(t) = MAX - \frac{t}{T}(MAX - MIN) \qquad (16)$$

where MAX and MIN are selected as 0.5 and 1e-10 which is determined by depending on user experience.

If we normalize the C_i from [min,max] to [0,V(t)], $C_i^{norm}(t)$ will be:

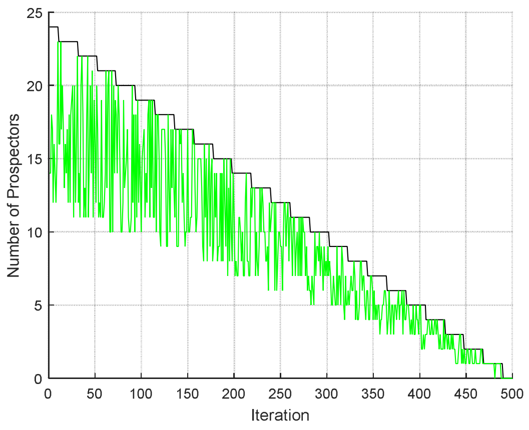


Fig. 1. Decreasing of prospectors with and without chaos

$$C_i^{norm}(t) = \frac{(C_i(t) - \min) \times (V(t) - 0)}{(\max - \min)} + 0$$
$$C_i^{norm}(t) = \frac{(C_i(t) - \min) \times (V(t))}{(\max - \min)} \qquad (17)$$

where the min and max are described as the chaotic boundaries in chaotic maps. When we reformulate the original formula with different chaotic maps as in (18), it can be seen in Fig. 1 that the convergence process becomes faster than the original MSA.

$$n_f = round\left((n - n_p) \times (1 - x)\right) \qquad x = \frac{t}{T} + C_i^{norm} \qquad (18)$$

V. Experimental Results

The proposed hybrid method, Chaotic Moth Swarm Algorithm (CMSA) is tested on the seven well-known benchmark functions which are common use unimodal test functions and the execution times of the benchmark functions for 30 run are given in the Table I [17]. The functions are listed in Table III where *Dim* represents the number of variables, and the *Range* is the range of the variables. MSA has several parameters which defined in Table II. The results are obtained and listed in Table IV. The results are averaged over 30 independent runs and the best means are stated in bold type. Consider, CMSA1 to CMSA10 are Chebyshev, Circle, Gauss/mouse, Iterative, Logistic, Piecewise, Sine, Singer, Sinusoidal, and Tent respectively [16].

TABLE I. Execution times of the benchmark functions

Benchmark function	Execution time for 30 run (sec.)	Execution time for one run (sec.)
F1	299.898378	9.9966126
F2	372.583906	12.419464
F3	644.196395	21.473213
F4	314.514308	10.483810
F5	338.309498	11.276983
F6	298.341361	9.944712
F7	336.632590	11.221086

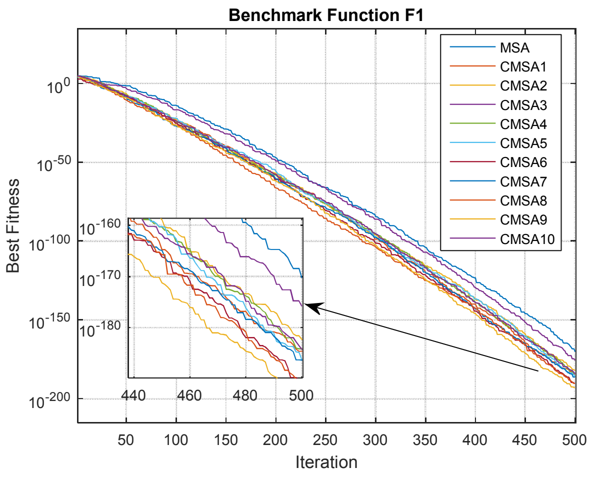
TABLE II. INITIAL PARAMETERS OF MSA

Parameter	Value
Search agents	30
Maximum Iterations	500
Number of Pathfinders	6
Chaos Value	0.5
Stopping criteria	Maximum Iteration

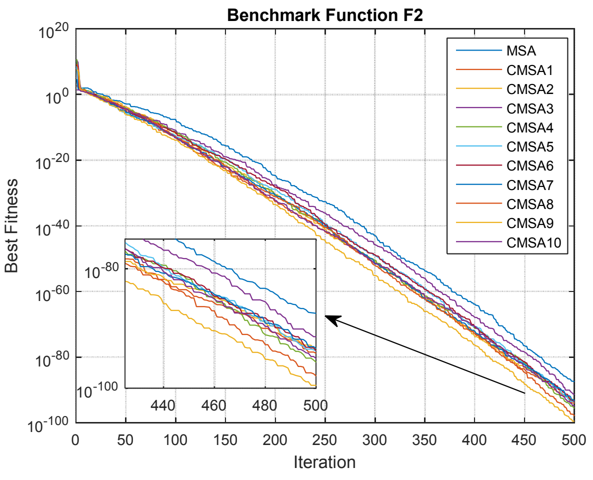
It can be seen from the Table IV that CMSA9 has the best performance for F_1, F_2, F_3, F_6 and F_7 in the way of bests of results. Besides, for F_4 and F_5, CMSA9 has the second best performance in the way of bests of results even though CMSA1 and CMSA7 have the best performance respectively. Therefore, we can say that CMSA9 provides the best performance for almost every function. As a result, original method MSA can be improved by using different chaotic maps for the different functions.

If we look from the way of means, CMSA9 still provides the best performance for almost all functions except F_5, and F_6. In Function 5, the average result of 30 run can give the best performance but if the iteration number increases, at least one chaotic map will provides the best results for the F_5.

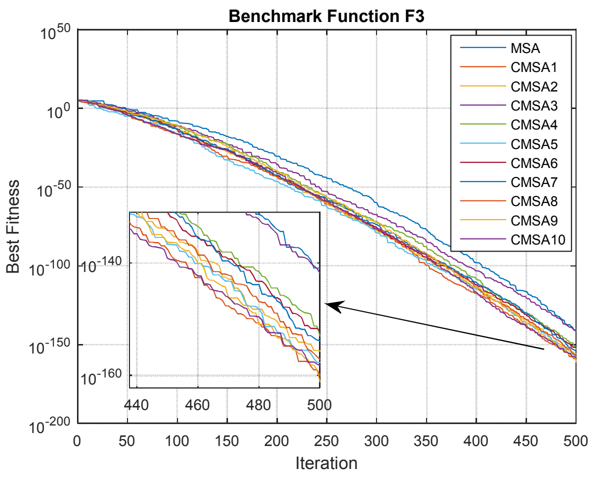
We rank the results in Table V to see the comparison easily. Table V shows us that the MSA method does not provide the best performance for almost any function and this means that all chaotic maps improve the results for the benchmark functions as compared with MSA. Besides, the superiority of the sinusoidal map is definite in Table IV if the probabilities of having best rank are taken. The graphical comparisons of the benchmark functions F1, F2, F3, and F4 are given in Fig. 2.



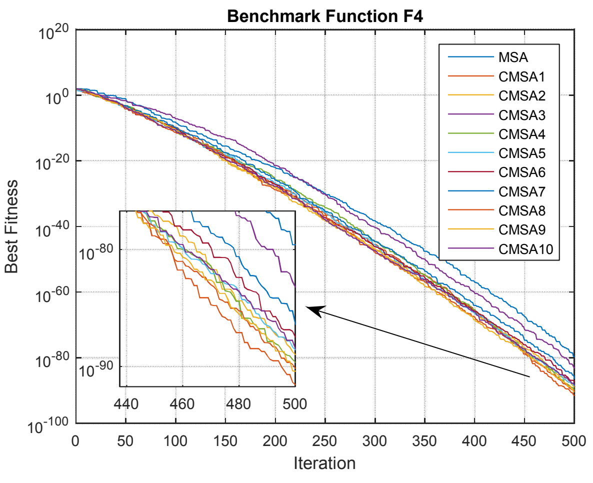
(a)



(b)



(c)



(d)

Fig. 2. Graphical comparison of benchmark functions (a) F1, (b) F2, (c) F3, (d) F4.

TABLE III. BENCHMARK FUNCTIONS

Test functions	Dim	Range
$F_1(x)=\sum_{i=1}^{n} x_i^2$	30	[-100,100]
$F_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	[-10,10]
$F_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[-100,100]
$F_4(x)=\max_i\left\{\lvert x_i\rvert, 1\le i\le n\right\}$	30	[-100,100]
$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$	30	[-30,30]
$F_6(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	30	[-100,100]
$F_7(x)=\sum_{i=1}^{n} ix_i^4+random\,[0,1)$	30	[-1.28,1.28]

TABLE IV. BEST, MEAN AND STD RESULTS OF THE MSA AND CHAOTIC MSA

		F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**
MSA	Best	6.9101e-171	3.6579e-88	1.2076e-141	2.5692e-80	27.2163	0.013145	9.4739e-05
	Mean	3.1631e-148	4.8044e-78	9.3484e-120	1.941e-70	**28.0869**	0.034683	0.00047052
	Std	1.6288e-147	2.6003e-77	5.1068e-119	1.0458e-69	0.419438	0.010143	0.00031246
CMSA1	Best	2.326e-185	7.3079e-95	1.0404e-160	**3.0196e-92**	27.5463	0.016046	5.2421e-05
	Mean	2.2309e-168	8.2413e-89	3.4193e-134	1.2213e-80	28.2023	0.033331	0.00055357
	Std	0	2.276e-88	1.8728e-133	4.3544e-80	0.392413	0.011411	0.00054761
CMSA2	Best	5.004e-183	4.6934e-94	2.207e-156	1.1289e-89	27.6804	0.013867	3.3494e-05
	Mean	1.3642e-166	2.1339e-85	7.8933e-139	3.1749e-80	28.4052	0.033161	0.00044904
	Std	0	1.1376e-84	3.5635e-138	8.4981e-80	**0.311492**	0.012163	0.00034604
CMSA3	Best	1.8917e-176	4.454e-92	1.3756e-142	6.4053e-84	27.2182	0.0095539	4.3426e-05
	Mean	1.3904e-159	1.9431e-81	6.7926e-129	7.8397e-75	28.0884	0.030616	0.00051871
	Std	5.9215e-159	1.0052e-80	2.6295e-128	4.2527e-74	0.455884	0.012126	0.00037633
CMSA4	Best	6.9997e-185	2.2401e-96	9.7024e-154	2.3567e-90	27.3607	0.016414	5.434e-05
	Mean	1.4148e-171	3.1706e-86	5.7876e-136	6.1436e-78	28.2287	0.031186	0.00040974
	Std	0	1.4949e-85	2.9685e-135	3.2939e-77	0.364644	0.010777	0.00032063
CMSA5	Best	5.9291e-187	1.7523e-94	2.1823e-158	2.8237e-89	27.5278	0.0096128	1.2787e-05
	Mean	2.1703e-172	9.3095e-87	4.2406e-133	2.1659e-81	28.2418	**0.02883**	0.00036893
	Std	0	4.6122e-86	2.3219e-132	7.688e-81	0.428507	0.010979	0.00029716
CMSA6	Best	4.2222e-191	2.0112e-94	8.4009e-153	3.5404e-88	27.5672	0.012108	3.5262e-05
	Mean	1.0293e-167	9.8665e-88	2.4544e-138	7.977e-81	28.2023	0.033271	0.0004362
	Std	0	3.7315e-87	1.1288e-137	4.2307e-80	0.402631	0.012538	0.00035996
CMSA7	Best	4.3615e-187	4.336e-94	1.5204e-154	4.3281e-87	**26.8889**	0.011059	4.005e-05
	Mean	4.784e-169	6.7842e-86	6.8988e-133	8.9598e-80	28.2053	0.033968	0.00046189
	Std	0	2.3044e-85	3.7786e-132	4.707e-79	0.490371	0.011672	0.00033261
CMSA8	Best	4.1802e-191	1.0704e-98	7.6235e-158	7.2287e-91	27.5585	0.010023	3.9711e-05
	Mean	2.5072e-173	5.6717e-89	7.6857e-142	2.9078e-81	28.3587	0.032627	0.00048203
	Std	0	3.0204e-88	**4.0289e-141**	1.3797e-80	0.359342	0.012184	0.00049309
CMSA9	Best	**1.2059e-193**	**3.0913e-100**	**7.3954e-162**	4.1051e-91	26.9038	**0.0091995**	**7.3912e-06**
	Mean	**6.1558e-176**	**4.923e-89**	**6.3678e-142**	**1.951e-83**	28.1535	0.032172	**0.00036023**
	Std	0	**1.9354e-88**	3.2956e-140	**1.0016e-82**	0.453409	0.0098094	0.0002347
CMSA10	Best	5.8462e-185	1.0322e-95	6.1699e-159	3.5867e-89	27.2593	0.012898	1.9235e-05
	Mean	3.8448e-172	1.5912e-87	9.2971e-138	2.4645e-79	28.2705	0.030998	0.00049199
	Std	0	5.8094e-87	4.9851e-137	1.3461e-78	0.41477	0.011384	0.00038501

TABLE V. THE RANK OF THE RESULTS

BEST								MEAN							
Methods	F1	F2	F3	F4	F5	F6	F7	Methods	F1	F2	F3	F4	F5	F6	F7
MSA	11	11	11	11	3	8	11	MSA	11	11	11	11	1	11	7
CMSA1	6	5	2	1	8	10	9	CMSA1	7	3	7	5	4	9	11
CMSA2	9	9	6	5	11	9	4	CMSA2	9	9	3	6	11	7	5
CMSA3	10	10	10	10	4	2	8	CMSA3	10	10	10	10	2	2	10
CMSA4	8	3	8	4	6	11	10	CMSA4	5	7	6	9	7	4	3
CMSA5	5	6	4	6	7	3	2	CMSA5	3	6	8	2	8	1	2
CMSA6	3	7	9	8	10	6	5	CMSA6	8	4	4	4	5	8	4
CMSA7	4	8	7	9	1	5	7	CMSA7	6	8	9	7	6	10	6
CMSA8	2	2	5	3	9	4	6	CMSA8	2	2	2	3	10	6	8
CMSA9	**1**	**1**	**1**	**2**	**2**	**1**	**1**	**CMSA9**	**1**	**1**	**1**	**1**	**3**	**5**	**1**
CMSA10	7	4	3	7	5	7	3	CMSA10	4	5	5	8	9	3	9

VI. CONCLUSION

In this work, ten chaotic maps are embedded into the Moth Swarm Algorithm (MSA) which is a new nature-based optimization algorithm. The purpose of this study is to eliminate the slow convergence problem of MSA with using different mathematical approach such as chaotic maps. The methods are tested in seven benchmark functions to decide whether the results are appreciable or not. In testing, simulation runs 30 times and the results were taken as the best and mean of this 30 run. The results prove that almost all chaotic maps speed the convergence respect to MSA.

REFERENCES

[1] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," Knowledge-Based Systems 89 (2015) pp. 228-249.

[2] J. Kennedy, R. Eberhart, "Particle swarm optimization," Neural Networks (1995) pp. 1942-1948.

[3] D. Karaboga, B. Akay, "A comparative study of Artificial Bee Colony algorithm," Applied Mathematics and Computation 214 (2009) 108–132.

[4] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-s. Pan, "Multi-strategy ensemble artificial bee colony algorithm," Information Sciences 279 (2014) 587–603.

[5] S. S. Jadon, J. C. Bansal, R. Tiwari, H. Sharma, "Accelerating Artificial Bee Colony algorithm with adaptive local Search," Memetic Comp. (2015) 7:215–230.

[6] W.-F. Gao, L.-L. Huang, S.-Y. Liu, C. Dai, "Artificial Bee Colony Algorithm Based on Information Learning," 2015.

[7] M. Dorigo, V. Maniezzo, A. Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Trans. Syst. Man Cybern. B Cybern. (1996) pp. 29-41.

[8] *K.* Premalatha, A.M. Natarajan, "A new approach for data clustering based on PSO with local search," Computer and Information Science (2008).

[9] J. J. Lang, P. N. Suganthan, "Dynamic multi-swarm Particle Swarm optimizer with local search," Proceedings of IEEE Swarm Intelligence Symposium (2005) pp. 124-129.

[10] G. Wu, D. Qiu, Y. Yu, W. Pedrycz, M. Ma, H. Li, "Superior solution guided particles warm optimization combined with local search techniques," Expert Systems with Applications (2014) pp. 7536-7548.

[11] C. Ozturk, D. Karaboga, "Hybrid artificial bee colony algorithm for neural network training" Evolutionary Computation (CEC) 2011 IEEE Congress on (2011) pp. 84-88.

[12] D. Karaboga, B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," Journal of Global Optimization (2007) pp. 459-471.

[13] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," Artificial Intelligence Review (2014) pp. 21-57.

[14] A.-A. A. Mohamed, Y. S. Mohamed, A. A.M. El-Gaafary, A. M. Hemeida, "Optimal power flow using moth swarm algorithm," Electric Power Systems Research (2017) pp. 190-206.

[15] S. Talatahari, B. Farahmand Azar, R. Sheikholeslami, and A. Gandomi, "Imperialist competitive algorithm combined with chaos for global optimization," Communications in Nonlinear Science and Numerical Simulation, vol. 17, no. 3, pp. 1312–1319, 2012.

[16] S. Mirjalili, A. H. Gandomi, "Chaotic gravitational constants for the gravitational search algorithm," Applied Soft Computing (2017) pp. 407-419.

[17] S. Duman, N. Yorukeren, I. H. Altas, "A Novel Modified Hybrid PSOGSA based on Fuzzy Logic for Non-convex Economic Dispatch Problem with Valve-point Effect," International Journal of Electrical Power & Energy Systems (2015) pp. 121-135.