Periodicals of Engineering and Natural Scinces ISSN 2303-4521
Vol.5, No.2, October 2018, pp. 1~9
Available online at: http://pen.ius.edu.ba

SCSO: A novel sine-cosine based swarm optimization algorithm for numerical function optimization

Türker TUNCER[1]
[1] Department of Digital Forensics Engineering, Technology Faculty, Firat University, Elazig, Turkey

Article Info

Article history:

Received Feb 5, 2018
Revised June 4, 2018
Accepted August 28, 2018

Keyword:

Sine-cosine optimization
Swarm Optimization
Swarm inspired evolutionary algorithm
Numerical Functions Optimization

ABSTRACT

Many swarm optimization algorithms have been presented in the literature and these algorithms are generally nature-inspired algorithms. In this paper a novel sine-cosine based particle swarm optimization (SCSO) is presented. In SCSO, firstly particles are generated randomly in the search space. Personal best value and velocity of the particles are calculated and by using sine, cosine and difference valuee. Calculated velocity is used for updating particles. The proposed algorithm is basic algorithm and approximately 30 rows MATLAB codes are used to implement the proposed algorithm. This short code surprisingly has high optimization capability. In order to evaluate performance and prove success of this algorithm, 14 well known numerical functions was used and the results illustrate that the proposed algorithm is successful in numerical functions optimization.

Corresponding Author:

Türker TUNCER,
Departement of Digital Forensics Engineering,
Firat University,
23119, Elazig, TURKEY.
Email: turkertuncer@firat.edu.tr

1. Introduction

Optimization is the process of searching a global optimum solution of a problem in a finite search space. Optimization algorithms consist of two sub-classes and these are gradient based optimization and meta-heuristic optimization algorithms. In the real-world applications, some problems cannot be solved by using mathematically approaches. In order to solve these problems, meta-heuristic optimization algorithms have been used Meta-heuristic algorithms do not require gradient knowledge and they call fitness (objective) function repeatedly in order to find global minimum. These algorithms try to narrow the search space and find an effective solution. In the past two decade, researchers proposed many optimization algorithms and optimization has become hot-topic research area and the well-known swarm optimization algorithm is Particle Swarm Optimization (PSO). PSO is proposed by Kenedy et al. [1] in 1995 and this algorithm is nature-inspired heuristic optimization algorithm. In the PSO, social behavior of individuals of fish and bird swarms were mathematically modelled. Besides the PSO, many optimization algorithms such as artificial neural network [2,3], ant colony optimization (ACO) [4], moth-flame optimization (MFO) algorithm [5], artificial bee colony (ABC) algorithm[6] ,firefly algorithm [7], sine-cosine algorithm (SCA) [8], genetic algorithm (GA) [9], bat algorithm (BA) [10], differential evaluation (DE) [11], biogeography-based optimization (BBO) [12] , harmony search (HS) [13], gravitational search algorithm (GSA) [14], krill herd algorithm (KH) [15], etc were proposed in the literature. The main aim of these algorithms to find global optima value but some of

them trapped local optima values. In order to obtain more successful results, chaotic maps were used to calculate velocity of these algorithms [16]. Can and Alatas [17] presented performance comprisons of current nature-inspired metaheuristic optimization algorithms. He used 7 benchmark functions to evalueate Ant Lion Optimization (ALO) [18], Dragonfly Algorithm (DA) [19], Grey Wolf Optimization (GWO) [20], MFO [5], Multi-Verse Optimizer (MVO) [21], SCA [8] and Whale Optimization Algorithm (WOA) [22] methods. Also, optimization techniques can be solved many real world problems [23]. A hybrid method which used GWO and SCA together is presented by Singh and Singh to achieve more successful results and this method called as hybrid grey wolf optimizer sine cosine algorithme (HGWOSCA) [24]. We need a successful, basically implemented and effective optimization technique. Thus, a novel SCSO algorithm is proposed. In order to implement this algorithm, only approximately 30 rows MATLAB code has been used. The characteristics of the proposed algorithm given as follows. SCSO is a basic and effective swarm optimization algorithm and it uses sine and cosine to find global optima. We used 14 numerical benchmark functions to evaluate performance of SCSO. The obtained results and comparisons showed that, SCSO is successful search algorithm for numerical function optimization.

The organization of this article given as follows. In section 2, the proposed sine-cosine swarm optimization algorithm is mentioned, in section 3, numerical functions and the obtained results are given and finally conclusions and recommendations is presented in the section 4.

2. The proposed sine-cosine based swarm optimization algorithm: SCSO

A novel sine-cosine based swarm optimization method is presented in this paper. SCSO is modified version of the SCA and it is a heuristic search method [8]. SCSO consists of initial value generation, local optima selection, particles updating and best value selection. Firstly, particles are generated randomly in the search space. Then, personal best value are computed by using objective function and particles are updated sine-cosine based particle updating equation. In the SCSO, particles are used for searching global optima. In this section, steps, pseudo code and MATLAB code of the SCSO are presented. We give MATLAB code of this method for researchers in this paper. Researchers can use MATLAB code of SCSO in order to solve their problems. The steps of the SCSO is given in below.

Step 1: Generate initial particles randomly in the search space.

$$p_i = rand(0,1) \text{ x } (UB - LB) + LB$$

Where p_i is particle, UB is upper bound, LB is lower bound.

Step2: Evaluate each particles by using objective function and calculate personal best (pbest) value.

Step 3: Generate r randomly. Range of r is [0,1].

Step 4: Calculate velocity.

$$v = \begin{cases} \sin(2\pi r) \text{ x } (p_{best} - p_i), r < 0.5 \\ \cos(2\pi r) \text{ x } (p_{best} - p_i), r \geq 0.5 \end{cases}$$

Step 5: Update all particles by using velocity.

$$p_i = p_i + v$$

Step 6: Evaluate all updated particles by using objective function.

Step 7: If particles exceed lower bound or upper bound, generate new particles in range of lower bound and upper bound randomly.
Step 8: Update pbest.
Step 9: Repeat steps 4 and 8 until the global optima is found or maximum iterations is reached.
The pseudo code of the proposed algorithm is shown in Figure 1.

1: Initialize parameters (LB is lower bound, UB is upper bound, MI is maximum iteration, pn is number of particles)
2: Randomly generate position of each particle in the swarm.
3: Evaluate each particle by using fitness function and find $\mathbf{p_{best}}$ in the swarm
4: **for** iter=1:MI **do**
5: **for** i=1:pn **do**
6: $r = rand[0,1]$
7: **if** r<0.5 **then**
8: $v = \sin(2\pi r)$ x $(p_{best} - p_i)$
9: **else**
10: $v = \cos(2\pi r)$ x $(p_{best} - p_i)$
11: **End if**
12: $p_i = p_i + v$
13: **if** p_i < LB or p_i>UB **then**
14: Randomly assigned position to p_i in range of the search space
15: **End if**
16: Evaluate particle by using fitness function
17: **if** fitness value of p_i better than pbest **then**
18: p_i is pbest
19: **End if**
20: **if** fitness value of pbest is gbest **then**
21: break;
22: **End if**
23: **End for iter**
24: **End for k**

Figure 1. Pseudo code of the SCSO algorithm.

To implement of the SCSO and validate results by researchers, MATLAB code of the SCSO is given in this paper and it is shown in Fig. 2.

```
for dim=1:40
LB=-100; UB=100; pnum=30; % Lower bound, Upper Bound and
number of particles
p(:,1)=rand(1,pnum)*(UB-LB)+LB; % Generate Population
Randomly
iter=500;
fit=fitness(p(:,1)); % Calculate fitness values of particles
[mini ind]=min((fit));
pbest=fitness(p(ind,1)); % Find personal best value
tic
for k=1:iter
for i=1:pnum
    r=rand();
    if(r<.5)
        p(i,1)=p(i,1)+sin(2*pi*r)*((p(ind,1)-p(i,1)));
    else
        p(i,1)=p(i,1)+cos(2*pi*r)*((p(ind,1)-p(i,1)));
    end
        if(p(i,1)<LB || p(i,1)>UB)
            p(i,1)=rand*(UB-LB)+LB;
        end
    t=(fitness(p(i,1)));
    if(t<pbest)
        ind=i;
        pbest=t;  % Update personal best value
    end
    if(fitness(pbest)==0)
        break;
    end
end
curves(k)=pbest;
end
toc
results(dim)=min(curves);
end
```

Figure 2. The MATLAB code of the SCSO.

3. Experimental Results and Discussions

In order to evaluate the effectiveness of the SCSO, 14 well known numerical benchmark functions are used and these functions are evaluated into two subgroups which are unimodal and multimodal. In this paper, 7 unimodal and 7 multimodal benchmark functions were used. These numerical benchmark functions are adopted widely used optimization techniques. These numerical benchmark functions are given.
Table 1. The widely used numerical benchmark functions [16-25].

Table 1. Numerical Benchmark Functions

Group	Name	Test Function	Space	Global opt.
Unimodal	Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$	$[0]^n$
	Schwefel's 2.22	$f_2(x) \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$	$[0]^n$

	Schwefel's 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	[-100, 100]^n	[0]^n
	Schwefel's 2.21	$f_4(x) = \max(\lvert x_i \rvert)$	[-100, 100]^n	[0]^n
	Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30, 30]^n	[0]^n
	Step	$f_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	[-100, 100]^n	[0.5]^n
	Noise	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	[-1.28, 1.28]^n	[0]^n
Multimodal	Rastrigin	$f_8(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12, 5.12]^n	[0]^n
	Ackley	$f_9(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	[-32, 32]^n	[0]^n
	Griewank	$f_{10}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[-600, 600]^n	[0]^n
	Generalized Penalized I	$f_{11}(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)[1 + 10\sin^2(\pi y_{i+1}) + (y_n + 1)^2]\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	[-50, 50]^n	[0]^n
	Generalized Penalized II	$f_{12}(x) = \frac{1}{10}\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + \sin^2(3\pi x_i)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \le x_i \le a \\ k(-x_i - a)^m, x_i < -a \end{cases}$	[-50, 50]^n	[0]^n
	Alphine I	$f_{13}(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	[-10, 10]^n	[0]^n
	Alphine II	$f_{14}(x) = \prod_{i=1}^{n} \lvert \sin(x_i)\sqrt{x_i} \rvert$	[0, 10]^n	[0]^n

The proposed SCSO algorithm applied on these functions and the obtained results are given in Table 2 with various parameters.

Table 2. Performance of SCSO with variable iteration numbers and particles and Dim=30

Fun	Criteria	Max Iteration (MI) and particles (P)					
		MI=100 P=10	MI=100 P=30	MI=500 P=10	MI=500 P=30	MI=1000 P=10	MI=1000 P=30
f_1	Mean	5.0324	9.753e-11	3.3292	2.6607e-13	0.2590	9.6535e-14
	SD	19.3002	3.895e-10	10.5428	1.1919e-12	0.6824	4.9319e-13
f_2	Mean	0.3769	2.9041e-06	0.2393	3.8972e-08	0.2781	6.0557e-09
	SD	1.0765	1.5906e-05	0.8736	1.5758e-07	0.6490	2.8264e-08
f_3	Mean	10.4067	6.1038e-04	4.9515	7.9080e-07	0.4060	1.0797e-09
	SD	52.9323	0.0033	17.7746	4.3164e-06	1.1109	4.3009e-09
f_4	Mean	1.0942	1.3181e-05	1.3364	4.0232e-07	0.9944	1.1874e-07
	SD	3.7501	6.2305e-05	4.4977	1.9943e-06	2.6318	6.4536e-07
f_5	Mean	0	0	0	0	0	0
	SD	0	0	0	0	0	0
f_6	Mean	1.2908	1.2014e-06	1.0004	1.2992e-09	0.6121	9.5653e-10
	SD	5.3255	6.5782e-06	4.2928	7.1158e-09	2.1661	5.1086e-09
f_7	Mean	0.0128	0.0058	0.0101	0.0056	0.0097	0.0037
	SD	0.0224	0.0083	0.0221	0.0135	0.0131	0.0076
f_8	Mean	0.5652	0.1057	0.4434	0.0879	0.2710	0.0332
	SD	0.5909	0.3034	0.8202	0.2727	0.4326	0.1817
f_9	Mean	0.8497	0.0081	0.6921	1.8260e-04	0.3905	2.7394e-06
	SD	2.1508	0.0439	1.4869	9.4229e-04	0.9603	1.1228e-05
f_{10}	Mean	0.3330	0.0120	0.2183	0.0118	0.1512	0.0059
	SD	0.8058	0.0212	0.3199	0.0147	0.3173	0.0102
f_{11}	Mean	1.8580	0.5185	1.2724	0.3110	1.0538	0.2124
	SD	1.6031	1.1788	1.5254	0.9490	1.5088	0.7882
f_{12}	Mean	0.1961	7.5574e-04	0.0502	2.0843e-05	0.0465	5.4108e-15
	SD	0.4982	0.0035	0.1467	1.1416e-04	0.1472	2.0563e-14
f_{13}	Mean	0.0146	0.0018	0.0095	2.0160e-04	0.0091	2.7552e-07
	SD	0.0589	0.0100	0.0314	0.0011	0.0312	1.1563e-06
f_{14}	Mean	0.0484	0.0131	0.0468	0.0105	0.0199	0.0036
	SD	0.1201	0.0327	0.0682	0.0372	0.0462	0.0112

These numerical benchmark functions which are listed in Table 1 are applied to widely used swarm optimization algorithms in order to obtain comparisons. Moth-flame optimization (MFO) [5] algorithm, artificial bee colony (ABC) algorithm [6], sine-cosine algorithm (SCA) [8], biogeography-based optimization (BBO) [12] and krill herd algorithm (KH) [15] and hybrid grey wolf optimizer sine cosine algorithm (HGWOSCA) [24] are used to obtain comparisons. Population, maximum iteration, dim of each object and other parameters of these algorithms are listed in Table 3.

Table 3. Parameters of the algorithms [16]

Algorithms	Population	Number of maximum iteration	Dim of each object	Other
MFO	40	500	30	t is random [-2,1]
ABC	40	500	30	Limit=200
SCA	40	500	30	r_1, r_2, r_3 and r_4 randomly generated.
BBO	40	500	30	Mu=0.005, m=0.8
KH	40	500	30	N^{max}=0.01 V_f=0.02 D^{max}=0.005
HGWOSCA	40	500	30	r1, r2, r3 and r4 randomly generated.
SCSO	40	500	30	r randomly generated

These algorithms and SCSO were applied to the first 12 numerical benchmark functions listed in Table 1 and performance comparisons are listed in Table 4.

Table 4. Performance comparison results.

f	Criteria	MFO	ABC	SCA	BBO	KH	HGWOSCA	SCSO
f_1	Mean	2×10^{3}	3.15×10^{-5}	1.11×10^{1}	7.17×10^{0}	5.19×10^{-1}	3.72×10^{3}	**3.10×10^{-38}**
	S.D.	4.22×10^{3}	3.47×10^{-5}	1.83×10^{1}	4.79×10^{0}	2.76×10^{-1}	1.24×10^{4}	**1.18×10^{-37}**
f_2	Mean	3.51×10^{1}	4.42×10^{-3}	1.62×10^{-2}	6.82×10^{-1}	3.84×10^{0}	1.14×10^{11}	**5.01×10^{-10}**
	S.D.	2.27×10^{1}	1.40×10^{-3}	2.12×10^{-2}	2.28×10^{-1}	2.11×10^{0}	8.07×10^{11}	**2.74×10^{-9}**
f_3	Mean	1.86×10^{4}	1.78×10^{4}	2.10×10^{4}	6.85×10^{3}	3.31×10^{2}	2.98×10^{4}	**1.62×10^{-20}**
	S.D.	1.40×10^{4}	2.92×10^{3}	1.12×10^{4}	1.67×10^{3}	$1.41\ 10^{2}$	3.72×10^{4}	**8.91×10^{-20}**
f_4	Mean	6.76×10^{1}	4.35×10^{1}	7.37×10^{1}	1.20×10^{0}	6.09×10^{0}	3.07×10^{1}	**4.83×10^{-12}**
	S.D.	9.49×10^{0}	5.05×10^{0}	2.11×10^{1}	4.50×10^{0}	1.46×10^{0}	2.04×10^{1}	**1.62×10^{-11}**
f_5	Mean	9.59×10^{2}	4.36×10^{1}	1.70×10^{6}	8.49×10^{2}	8.96×10^{0}	5.18×10^{6}	**0**
	S.D.	1.04×10^{3}	4.41×10^{1}	4.86×10^{6}	5.24×10^{2}	1.08×10^{2}	2.91×10^{7}	**0**
f_6	Mean	1.01×10^{3}	5.82×10^{-5}	3.69×10^{1}	1.48×10^{1}	4.61×10^{-1}	5.36×10^{3}	**2.65×10^{-21}**
	S.D.	3.19×10^{3}	9.74×10^{-5}	8.84×10^{1}	2.59×10^{1}	1.56×10^{-1}	1.44×10^{4}	**1.44×10^{-20}**
f_7	Mean	6.31×10^{0}	3.03×10^{-1}	6.65×10^{-1}	8.31×10^{-2}	7.76×10^{-2}	6.07×10^{0}	**2.30×10^{-3}**
	S.D.	8.85×10^{0}	8×10^{-2}	1.81×10^{0}	4.53×10^{-2}	3.17×10^{-2}	2.08×10^{1}	**4.70×10^{-3}**
f_8	Mean	1.59×10^{2}	5.09×10^{0}	5.91×10^{1}	3.21×10^{2}	1.45×10^{1}	1.51×10^{1}	**8.40×10^{-3}**
	S.D.	2.66×10^{1}	2.32×10^{0}	3.65×10^{1}	6.47×10^{0}	1.07×10^{1}	5.97×10^{1}	**4.51×10^{-2}**
f_9	Mean	8.86×10^{-8}	1.54×10^{-1}	1.15×10^{1}	1.69×10^{0}	5.52×10^{0}	3.18×10^{0}	**1.73×10^{-14}**
	S.D.	6.38×10^{-8}	1.43×10^{-1}	9.95×10^{0}	4.70×10^{-1}	1.40×10^{0}	6.03×10^{0}	**8.81×10^{-14}**
f_{10}	Mean	8.59×10^{-1}	2.60×10^{-2}	1.04×10^{0}	1.08×10^{0}	1.48×10^{-1}	6.22×10^{0}	**9.10×10^{-3}**
	S.D.	1.44×10^{-1}	2.86×10^{-2}	2.94×10^{-1}	7.83×10^{-2}	4.04×10^{-2}	4.61×10^{1}	**1.31×10^{-2}**
f_{11}	Mean	4.69×10^{0}	**1.93×10^{-5}**	4.17×10^{6}	3.42×10^{-1}	3.41×10^{0}	2.30×10^{7}	1.04×10^{-1}
	S.D.	1.81×10^{0}	**4.33×10^{-5}**	1.38×10^{7}	4.44×10^{-1}	1.20×10^{0}	1.03×10^{8}	5.67×10^{-1}
f_{12}	Mean	1.10×10^{1}	1.01×10^{-5}	1.51×10^{7}	6.52×10^{-1}	3.85×10^{-2}	5.83×10^{7}	**9.78×10^{-12}**
	S.D.	9.65×10^{0}	1.04×10^{-5}	2.87×10^{7}	2.18×10^{-1}	1.41×10^{-2}	1.93×10^{8}	**5.25×10^{-11}**

In this section, numerical results are demonstrated clearly according to Ref. [25]. Experimental results of the SCSA were presented with variable parameters and these results were compared with the widely used meta-

heuristic optimization algorithms. To obtain comparisons, 12 numerical benchmark functions are used and best results were achieved in 11 of them. The experimental results clearly demonstrated that the SCSO has high minimization ability and is resulted successfully in terms of numerical function optimization.

4. Conclusions and Recommendations

In the literature, many nature-inspired swarm optimization algorithms but a few mathematical based swarm optimization algorithms have been presented. In this paper, a novel mathematical based swarm optimization algorithm is presented and this algorithm uses sine, cosine and step value. The proposed algorithms is called as SCSO. To calculate step value, bounds of search space and number of particles are utilized. This algorithm is a modified version of SCA[8] and the experiments clearly demonstrated that the SCSO have more successful results than SCA [8]. The proposed SCSO algorithm consists of randomly generate initial particles, finding local optima value, updating particles and searching global optima value. This algorithm used very simple mathematical model. In order to evaluate performance of the proposed SCSO and obtain experiments, 14 widely used numerical benchmark functions were used with various parameters. Also, 12 of these were utilized to obtain comparisons and SCSO achieved the best values in 11 of the 12 benchmark functions. The experiments and comparisons clearly illustrated that the SCSO algorithm is successful meta-heuristic optimization algorithm for numerical function optimization.
In the future work, the presented SCSO will be applied to real-world optimization problems such as deep learning, artificial intelligence, image segmentation, etc. and various mathematical models will be used for proposing novel swarm optimization algorithms.

References

[1] R. Ebenhart, Kennedy, Particle swarm optimization, in: *Proceeding IEEE Inter Conference on Neural Networks*, 4, Perth, Australia, Piscat-away, 1995, pp. 1942–1948.
[2] Y.Y. Lin, J.Y. Chang, C.T. Lin, Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network, *IEEE Trans. Neural Netw. Learn. Syst.* 24 (2) (2013) 310–321.
[3] C. Karakuzu, F. Karakaya, M.A. Çavusoğlu, FPGA implementation of neuro-fuzzy system with improved PSO learning, Neural Netw. 79 (2016) 128–140.
[4] K.L. Du, M.N.S Swamy, Ant colony optimization, in: Search and Optimization by Metaheuristics, Springer International Publishing, 2016, pp. 191–199.
[5] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl. Based Syst. 89 (2015) 228–249.
[6] G. Li, P. Niu, Y. Ma, et al., Tuning extreme learning machine by an improved artificial bee colony to model and optimize the boiler efficiency, Knowl. Based Syst. 67 (2014) 278–289.
[7] X.S. Yang, Multiobjective firefly algorithm for continuous optimization, Eng. Comput. 29 (2) (2013) 175–184.
[8] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl. Based Syst. 96 (2016) 120–133.
[9] T. Vidal, T.G. Crainic, M. Gendreau, et al., A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, Comput. Oper. Res. 40 (1) (2013) 475–489.
[10] X.S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng. Comput. 29 (5) (2012) 464–483.
[11] P. Civicioglu, E. Besdok, A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms, Artif. Intell. Rev. (2013) 1–32.
[12] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimization with chaos, Neural Comput. Appl. 25 (5) (2014) 1077–1097.
[13] B. Alatas, Chaotic harmony search algorithms, Appl. Math. Comput. 216 (9) (2010) 2687–2699.
[14] F.Y. Ju, W.C. Hong, Application of seasonal SVR with chaotic gravitational search algorithm in electricity forecasting, Appl. Math. Model. 37 (23) (2013) 9643–9651.
[15] G.G. Wang, L. Guo, A.H. Gandomi, et al., Chaotic krill herd algorithm, Inf. Sci. 274 (2014) 17–34.

[16] K. Chen, F. Zhou, A. Liu, Chaotic dynamic weight particle swarm optimization for numerical function optimization, Knowledge-Based Systems 139 (2018) 23–40.
[17] U. Can, B. Alatas, Performance Comparisons of Current Metaheuristic Algorithms on Unconstrained Optimization Problems, Periodicals of Engineering and Natural Scinces, Vol.5, No.3, November 2017, pp. 328-340.
[18] S. Mirjalili, The ant lion optimizer, Advances in Engineering Software, 83, (2015), 80-98.
[19] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving singleobjective, discrete, and multi-objective problems. Neural Computing and Applications, 27(4), (2016), 1053-1073.
[20] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer. Advances in Engineering Software, 69, (2014), 46-61.
[21] S. Mirjalili, S.M. Mirjalili, A. Hatamlou, Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Computing and Applications, 27(2), (2016), 495-513.
[22] B. Alatas, E. Akin, A.B. Ozer, Chaos Embedded Particle Swarm Optimization Algorithms, Chaos, Solitons & Fractals, vol. 40, (2009), 1715-1734.
[23] M.C. Canoğlu, B. Kurtuluş, Determination of the Dam Axis Permeability for the Design and the Optimization of Grout Curtain: An Example from Orhanlar Dam (Kütahya-Pazarlar), Periodicals of Engineering And Natural Sciences Vol. 5 No. 1 (2017), 37-43.
[24] N. Singh, S.B. Singh, A novel hybrid GWO-SCA approach for optimization problems, Engineering Science and Technology, an International Journal 20 (2017) 1586–1601.
[25] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[26] B. Durakovic, Design of Experiments Application, Concepts, Examples: State of the Art, Periodical of Engineering and Natural Sciences, Vol. 5, No. 3, pp. 421-439 (2017).

BIBLIOGRAPHY OF AUTHORS (11 PT) (Optional)

Türker Tuncer, was born in Elazig, Turkey, in 1986. He received the B.S. degree from the Firat University, Technical Education Faculty, Department of Electronics and Computer Education in 2009, M.S. degree in telecommunication science from the Firat University in 2011. and PHD degree department of software engineering at Firat University in 2016. He works as research assistant Digital Forensics Engineering, Firat University. His research interests include watermarking, data hiding, image authentication, perceptual hashing, image processing and evolutionary programming.