Elite Opposition-Based Selfish Herd Optimizer

Shengqi Jiang, Yongquan Zhou, Dengyun Wang, Sen Zhang

Elite Opposition-based Selfish Herd Optimizer

Shengqi Jiang [1], Yongquan Zhou[1,2], Dengyun Wang[1], Sen Zhang[1]

[1]College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China
[2]Guangxi High School Key Laboratory of Complex System and Computational Intelligence, Nanning 530006, China
yongquanzhou@126.com

Abstract: Selfish herd optimizer (SHO) is a new metaheuristic optimization algorithm for solving global optimization problems. In this paper, an elite opposition-based Selfish herd optimizer (EOSHO) has been applied to functions. Elite opposition-based learning is a commonly used strategy to improve the performance of metaheuristic algorithms. Elite opposition-based learning enhances the search space of the algorithm and the exploration of the algorithm. An elite opposition-based Selfish herd optimizer is validated by 7 benchmark functions. The results show that the proposed algorithm is able to obtain the more precise solution, and it also has a high degree of stability.

Keywords: Elite opposition-based learning; selfish herd optimizer; metaheuristic algorithm

1 Introduction

Metaheuristic optimization algorithm is a simulation of various biological behaviors in nature and has been concerned by researchers for many years. Many intelligent optimization algorithms for bionic groups are proposed. Such as particle swarm optimization(PSO)[1], ant colony optimization(ACO)[2], bat algorithm(BA)[3], grey wolf optimizer(GWO)[4], firefly algorithm(FA)[5], cuckoo search (CS)[6], flower pollination algorithm(FPA) [7] et al.

Selfish herd optimizer (SHO) is based on the simulation of the widely observed selfish herd behavior manifested by individuals within a herd of animals subjected to some form of predation risk has been proposed by Fernando Fausto and Erik Cuevas.et.al [8]. The algorithm is inspired by the Hamilton's selfish herd theory [9]. In this paper, an elite opposition-based selfish herd optimizer (EOSHO) has been applied to functions optimization. Improve the search ability of the population by doing elite opposition-based learning to the selfish herd group. EOSHO is validated by 7 benchmark functions. The results show that the proposed algorithm is able to obtain precise solution, and it also has a high degree of stability.

The remainder of the paper is organized as follows: Section 2 briefly introduces the original selfish herd optimizer； This is followed in Section 3 by new elite opposition-based selfish herd optimizer (EOSHO); simulation experiments and results analysis are described in Section 4. Finally, conclusion and future works can be found and discussed in Section 5.

2 Selfish Herd Optimizer (SHO)

2.1 Initializing the population

The algorithm begins by initialize the set $\mathbf{A}$ of N individual positions. SHO models two different groups of search agents: a group of prey and a group of predators. As such, the number of prey (N_h) and the number of predators (N_p) are calculated by the follow equations:

$$N_h = floor(N \cdot rand(0.7,0.9)) \tag{1}$$

$$N_p = N - N_h \tag{2}$$

According to the theory of selfish groups, each animal has its own survival value. Therefore, we distribute the value of each individual's survival as follows：

$$SV_i = \frac{f(\mathbf{a}_i) - f_{best}}{f_{best} - f_{worst}} \tag{3}$$

where $f(\mathbf{a}_i)$ is the fitness value that is obtained by the evaluation that evaluates $\mathbf{a}_i$ with regard to the objective function $f(\bullet)$. The values of f_{best} and f_{worst} are the value of the best and worst position in the population.

2.2 Herd movement operator

According to the selfish herd theory, we can get the gravity coefficient between the prey group member i and the prey group member j, are defined as follows:

$$\psi_{\mathbf{h}_i,\mathbf{h}_j} = SV_{\mathbf{h}_j} \cdot e^{-\left\|\mathbf{h}_i - \mathbf{h}_j\right\|^2} \tag{4}$$

The prey will avoid the predator, so there will be exclusion between them. The repulsion factor is shown as follows:

$$\varphi_{\mathbf{h}_i,\mathbf{p}_M} = \text{-}SV_{\mathbf{p}_M} \cdot e^{-\left\|\mathbf{h}_i - \mathbf{P}_M\right\|^2} \tag{5}$$

The leader's position in the next generation is updated as follows:

$$\mathbf{h}_L^k = \left(\mathbf{h}_i^k \in \mathbf{H}^k \middle| SV_{\mathbf{h}_i^k} = \max_{j\in\{1,2,\cdots,N_h\}} \left(SV_{\mathbf{h}_j^k} \right) \right) \tag{6}$$

$$\mathbf{h}_L^{k+1} = \begin{cases} \mathbf{h}_L^k + 2 \cdot \alpha \cdot \varphi_{\mathbf{h}_L,\mathbf{p}_M}^k \cdot \left(\mathbf{p}_M^k - \mathbf{h}_L^k \right) & if \quad SV_{\mathbf{h}_L^k} = 1 \\ \mathbf{h}_L^k + 2 \cdot \alpha \cdot \psi_{\mathbf{h}_L,\mathbf{x}_{best}}^k \cdot \left(\mathbf{x}_{best}^k - \mathbf{h}_L^k \right) & if \quad SV_{\mathbf{h}_L^k} < 1 \end{cases} \tag{7}$$

In the herd, followers and deserters in the next generation of location updates are as follows:

$$\mathbf{h}_i^{k+1} = \begin{cases} \mathbf{h}_i^k + \mathbf{f}_i^k & if \quad SV_{\mathbf{h}_i^k} < rand(0,1) \\ \mathbf{h}_i^k + \mathbf{d}_i^k & if \quad SV_{\mathbf{h}_i^k} > rand(0,1) \end{cases} \tag{8}$$

$$\mathbf{f}_i^k = \begin{cases} 2 \cdot \left(\beta \cdot \psi_{\mathbf{h}_i, \mathbf{h}_L}^k \cdot \left(\mathbf{h}_L^k - \mathbf{h}_i^k \right) + \gamma \cdot \psi_{\mathbf{h}_i, \mathbf{h}_{c_i}}^k \cdot \left(\mathbf{h}_{c_i}^k - \mathbf{h}_i^k \right) \right) & if \quad SV_{\mathbf{h}_i^k} > SV_{\mathbf{h}_\mu^k} \\ 2 \cdot \delta \cdot \psi_{\mathbf{h}_i, \mathbf{h}_M^k}^k \cdot \left(\mathbf{h}_M^k - \mathbf{h}_i^k \right) & if \quad SV_{\mathbf{h}_i^k} < SV_{\mathbf{h}_\mu^k} \end{cases} \tag{9}$$

where $SV_{\mathbf{h}_\mu^k}$ represents the mean survival value of the herd's aggregation.

$$\mathbf{d}_i^k = 2 \cdot \left(\beta \cdot \psi_{\mathbf{h}_i, \mathbf{x}_{best}}^k \cdot \left(\mathbf{x}_{best}^k - \mathbf{h}_i^k \right) + \gamma \cdot \left(1 - SV_{\mathbf{h}_i^k} \right) \cdot \mathbf{r} \right) \tag{10}$$

where **r** denotes unit vector pointing to a random direction within the given *n*-dimensional solution space.

2.3 Predators movement operators

A calculation of the probability $P_{\mathbf{p}_i, \mathbf{h}_j}$ that a prey can catch up with each herd, as follows:

$$P_{\mathbf{p}_i, \mathbf{h}_j} = \frac{\omega_{\mathbf{p}_i, \mathbf{h}_j}}{\sum_{m=1}^{N_h} \omega_{\mathbf{p}_i, \mathbf{h}_m}} \tag{11}$$

$$\omega_{\mathbf{p}_i, \mathbf{h}_j} = \left(1 - SV_{\mathbf{h}_j} \right) \cdot e^{-\left\| \mathbf{p}_i - \mathbf{h}_j \right\|^2} \tag{12}$$

The predator's position updating formula can be obtained as follows:

$$\mathbf{p}_i^{k+1} = \mathbf{p}_i^k + 2 \cdot \rho \cdot (\mathbf{h}_r^k - \mathbf{p}_i^k) \tag{13}$$

2.4 Predation phase

For each predator, we can define a set of threatened prey as follows:

$$T_{\mathbf{p}_i} = \left\{ \mathbf{h}_j \in \mathbf{H} \middle| SV_{\mathbf{h}_j} < SV_{\mathbf{p}_i}, \left\| \mathbf{p}_i - \mathbf{h}_j \right\| \le R, \mathbf{h}_j \notin \mathbf{K} \right\} \tag{14}$$

When multiple preys enter the range of the predator's attack radius, the predator will choose to kill prey based on the probability of roulette.

2.5 Restoration phase

Mating operations can not produce new individuals in the set of individuals, depends on each mating candidate students go, as follows:

$$\Omega_{\mathbf{h}_j} = \frac{SV_{\mathbf{h}_j}}{\sum_{\mathbf{h} \in \mathbf{M}} SV_{\mathbf{h}_m}} \tag{15}$$

In order to obtain a new individual, we should first consider the random acquisition of n individuals in a set of mating candidates. Replacing the hunted individual with a new individual.

$$\mathbf{h}_{new} = mix\left([\mathbf{h}_{r_1,1}, \mathbf{h}_{r_2,2}, \cdots, \mathbf{h}_{r_n,n}]\right) \tag{16}$$

Specific implementation steps of the Standard Selfish herd optimizer (SHO) can be summarized in the pseudo code shown in **Algorithm** 1.

Algorithm 1. SHO pseudo-code

1. **BEGIN**
2. Initialize each animal populations;
3. Define the number of herd members and predators within **A** by equation (1) and (2);
4. **While** (t < *iterMax*)
5. Calculate the fitness and survival values of each number by equation (4);
6. Update the position of herd numbers by equation (7) and (8);
7. Update the position of herd numbers by equation (13);
8. Define the predator will choose to kill prey based on the probability of roulette by equation (14);
9. Define the random acquisition of n individuals in a set of mating candidates by equation (16);
10. **End While**
11. Memorize the best solution achieved;
12. **END**

3 Elite opposition-based selfish herd optimizer (EOSHO)

Opposition-based learning is a technique proposed by Tizhoosh [10], has been applied to a variety of optimization algorithms [11, 12, 13]. The main purpose of it is to find out the candidate solutions that are closer to the global optimal solution. The elite opposition-based learning has been proved that the inverse candidate solution has a greater chance of approaching the global optimal solution than the forward candidate solution. The elite opposition-based learning has successfully improved the performance of many optimization algorithms. At the same time, it has been successfully applied to many research fields, such as reinforcement learning, morphological algorithm window memory and image processing using opposite fuzzy sets.

In this paper, we apply the elite opposition-based learning to the movement of a group of **H** (the group of prey): select the best fitness value of individuals in **H** as elite prey, hoping that this elite individual can guide the movement of the whole **H** group. To explain the concept of elite opposition-based learning, we introduce an example: We assume that the elite individual in **H** is $\mathbf{H}_e = (\mathbf{h}_{e,1}, \mathbf{h}_{e,2}, ..., \mathbf{h}_{e,n})$ .Then the elite opposition-based candidate solutions for

other individual $\mathbf{H}_i = (\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, ..., \mathbf{h}_{i,n})$ in $\mathbf{H}$ can be defined as $\mathbf{H}_i^{'} = (\mathbf{h}_{e,1}^{'}, \mathbf{h}_{e,2}^{'}, ..., \mathbf{h}_{e,n}^{'})$. And we can get a equation:

$$\begin{aligned} &\mathbf{h}_{i,j}^{'} = \alpha \cdot (da_j + db_j) - \mathbf{x}_{e,j} \\ &i = 1,2,\ldots,N;\ j = 1,2,\ldots,n \end{aligned} \tag{17}$$

Where N is the size of $\mathbf{H}$, n is the dimension of $\mathbf{H}$, $\alpha \in (0,1)$. da_j and db_j is the dynamic boundary of the j dimension of an individual. We define the dynamic boundary da_j and db_j as follows:

$$\begin{aligned} da_j &= \min(x_{i,j}) \\ db_j &= \max(x_{i,j}) \end{aligned} \tag{18}$$

In order to prevent the elite opposition-based learning point from jumping out of the dynamic boundary, we set that when the elite opposition-based learning point jumps out of the dynamic boundary range, we will reset the reverse point, which is as follows:

$$\mathbf{H}_i^{'} = rand(da_j, db_j) \quad if \quad \mathbf{H}_i^{'} < da_j \quad or \quad \mathbf{H}_i^{'} > db_j \tag{19}$$

By this method, the algorithm in the global search process is enhanced and the diversity of the population is improved.

4 Simulation experiments and result analysis

In this section, we used 6 standard test functions [14, 15] to test to get the performance of EOSHO. The rest of this section is organized as follows: the experimental setup is given in Section 4.1, and the performance of each algorithm is compared to section 4.2.The space dimension, scope, optimal value and the iterations of 6 functions are shown in **Table** 1.

Table 1 Benchmark test functions.

Benchmark Test Functions	Dim	Range	$f_{\min}$
$f_1 = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$f_3(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
$f_4(x) = \sum_{i=1}^{n} x_i^4 + random(0,1)$	30	[-1.28,1.28]	0
$f_5(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[-5.12,5.12]	0

$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2} - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)\right) + 20 + e$	3 0	[-32,32]	0

4.1 Experimental setup

All of the algorithms was programmed in MATLAB R2016a, numerical experiment was set up on AMD Athlont (tm) II*4640 processor and 2 GB memory.

4.2 Comparison of each algorithm performance

The proposed elite opposition-based selfish herd optimizer compared with swarm intelligence optimization algorithms, such as CS [6], PSO [1], MVO [16], ABC [17], SHO [7], we compare their optimal performance by means of mean and standard deviation respectively. The set values of the control parameters of the algorithm are given in **Table** 4.

For the standard reference function in **Table** 1, the comparison of the test results is shown in **Table** 3-4. In this paper, the population size is 50, the maximum number of iterations is 1000, and the results have been obtained in 30 trials. The Best, Mean and Std. represent the optimal fitness value, the average fitness value and the standard deviation. We compared the results of EOSHO and other algorithms to test the function, and listed the results ranking of EOSHO on the right side of the **Table** 2.

Table 2 the initial parameters of algorithms

Algorithm	Parameter	Value
CS	The probability of discovery (pa)	0.25
PSO	Cognitive constant (C_1)	1.5
	Social constant (C_2)	2
	Inertia constant (ω)	1
MVO	The probability of the existence of the wormhole (Wep)	linearly decreased from 0.2 to 1
ABC	Limit	D

where D denotes dimension of the problem, trial denotes internal information in ABC.

According to the results of **Table** 3. It can be noted that the single unimodal functions is suitable for the benchmark development. Compared with the original SHO algorithm, the calculation accuracy of the EOSHO algorithm has been greatly improved. For $f_1 \sim f_3$ the optimal fitness value, mean fitness value and standard deviation of EOSHO algorithm ate better than that of the other five algorithms, ranking first in all the algorithms. Fig 1-6 shows EOSHO and other algorithm convergence and the anova tests of the global minimum plots. As can be seen from Fig 1, EOSHO

can obtain higher optimization precision. The results show that for the 3 test functions, the EOSHO algorithm can provide good results and have smaller variance, which means that EOSHO has more advantages than MVO, PSO, CS, ABC and SHO algorithm in solving the problem of single unimodal function. EOSHO in the convergence speed and accuracy is relatively fast. Overall, the EOSHO algorithm accelerates the convergence speed and enhances the calculation accuracy.

Table 3 Simulation results for f_1 ~ f_3 in low-dimension

Bench-mark functions	Result	Algorithm						Rank
		MVO	PSO	CS	ABC	SHO	EOSHO	
f_1 (D=30)	Best	0.1091	9.9E-53	0.05397	3.7E-08	5.2E-11	3.9E-257	
	Mean	0.17877	6.9E-39	0.08540	5.1E-07	2.9E-10	3.7E-235	1
	Std	0.039	3.8E-38	0.01535	5.0E-07	3.1E-10	0	
f_2 (D=30)	Best	0.1091	1.2E-05	1.12153	1.1E-05	5.3E-11	1.5E-134	
	Mean	0.17877	0.01125	2.27540	3.1E-05	2.9E-10	6.5E-122	1
	Std	0.039	0.03003	0.66911	1.8E-05	3.2E-10	3.4E-121	
f_3 (D=30)	Best	0.0031	0.0039	0.17700	0.39195	1.4E-05	3.14E-06	
	Mean	0.01188	0.01164	0.39681	0.72350	0.00030	9.18E-05	1
	Std	0.00597	0.00367	0.11289	0.20553	0.00041	6.73E-05	

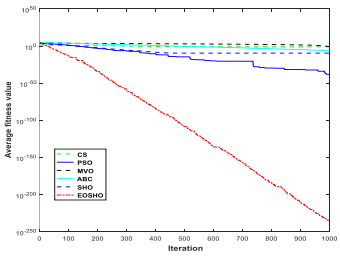

Fig.1 The convergence curves of f_1

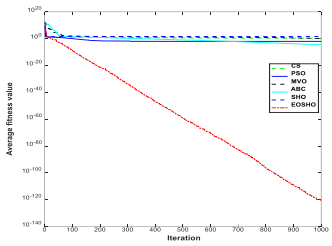

Fig.2 The convergence curves of f_2

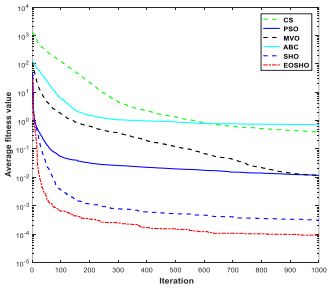

Fig.3 The convergence curves of f_3

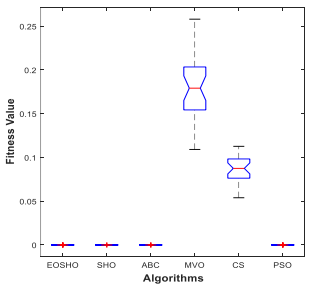

Fig.4 Standard deviation for f_1

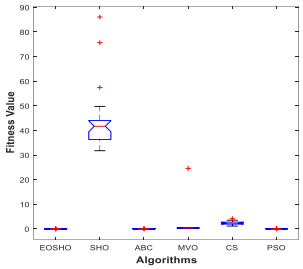

Fig.5 Standard deviation for f_2

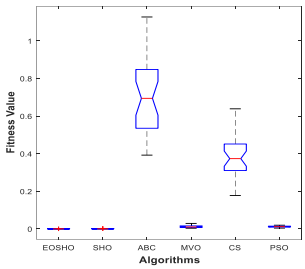

Fig.6 Standard deviation for f_3

According to the results of **Table** 4, the performance of the EOSHO algorithm in the multimodal function is much better than that of the other comparison algorithms.

Table 4 Simulation results for $f_4 \sim f_5$ in low-dimension

Benchmark functions	Result	Algorithm						Rank
		MVO	PSO	CS	ABC	SHO	EOSHO	
f_4 (D=30)	Best	51.8575	21.8890	30.7915	1.0967	12.9344	0	1
	Mean	99.9845	48.8524	44.6965	5.5057	31.9771	0	
	Std	25.8392	14.7027	8.0742	3.1272	10.3539	0	
f_5 (D=30)	Best	0.0998	8.3E-14	2.1600	0.0001	4.7E-14	8.9E-16	1
	Mean	0.6275	0.5604	2.9345	0.0006	0.3424	8.8E-16	
	Std	0.8018	0.6797	0.4305	0.00081	0.8058	0	

Compared with the unimodal functions, the multimodal functions have many local optimal solutions, and its number increases exponentially with the dimension. This makes them suitable for benchmarking the search capabilities of the algorithms. The results show that the advantages of EOSHO algorithm in exploration are also very competitive. For $f_4 \sim f_5$, compared with the result obtained by SHO algorithm, the calculation accuracy of optimal value of EOSHO algorithm is improved. For $f_4 \sim f_5$ The ranking of optimal fitness value for EOSHO algorithm is first, which indicates that EOSHO algorithm has been substantially improved.

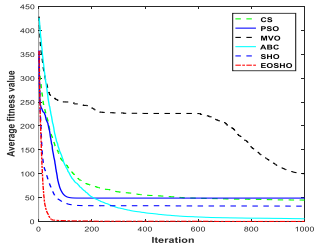

Fig.9 The convergence curves of f_5

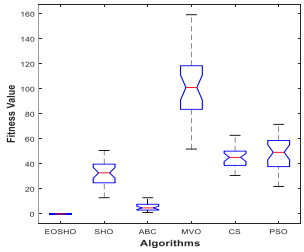

Fig.10 Standard deviation for f_5

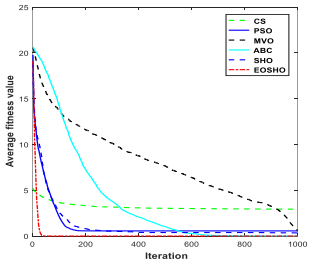

Fig.11 The convergence curves of f_6

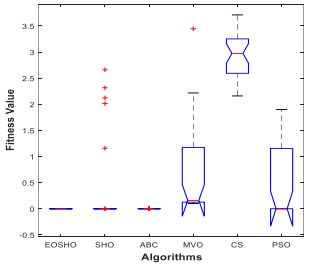

Fig.12 Standard deviation for f_6

According to Fig 9 and Fig 11, we can see that EOSHO has faster convergence speed and higher optimization accuracy. According to Fig 10 and Fig 12, we can get that EOSHO has strong stability. It can be seen that EOSHO has higher convergence accuracy and stronger robustness. On the whole, the SHO algorithm and elite opposition-based strategy improves accuracy of the SHO algorithm, which is helpful to find the global optimal solution.

5 Conclusions and future works

In this paper, to improve the convergence speed and calculation accuracy of the SHO algorithm, we add the elite opposition-based learning strategy to prey movement operators. The EOSHO algorithm is proposed, which can better balance exploration and exploitation, the elite opposition-based learning strategy increases the diversity of population to avoid the search stagnation. The EOSHO algorithm is testing for 7 benchmark functions. The optimization performance of the EOSHO algorithm has been greatly improved compared with the basic SHO algorithm. The optimal fitness valued of EOSHO algorithm is relatively small in the six algorithms.

For EOSHO, there are various idea that still deserve in the future study, Firstly, there exists many N-P hard problems in literature, such as planar graph coloring problem, radial basis probabilistic neural networks; Secondly, it is suggested to apply it to more engineering examples.

Acknowledgements

This work is supported by National Science Foundation of China under Grants No. 61463007, 61563008, and by Project of Guangxi Natural Science Foundation under Grant No. 2016GXNSFAA380264.

References

[1] Kennedy J, Eberhart R. Particle swarm optimization. IEEE International Conference on Neural Networks, 1995. Proceedings. IEEE, 2002:1942-1948 vol.4.

[2] Dorigo, M, Birattari, M, Stutzle, T. Ant colony optimization. Computational Intelligence Magazine, IEEE, 2004, 1(4):28-39.

[3] XinShe Yang. A New Metaheuristic Bat-Inspired Algorithm. Computer Knowledge and Technology, 2010, 284:65-74.

[4] Mirjalili S, Mirjalili S M, Lewis A. Grey Wolf Optimizer. Advances in Engineering Software, 2014, 69(3):46-61.

[5] Yang, Xin-She. Firefly Algorithm. Engineering Optimization. 2010:221-230.

[6] Yang X S, Deb S. Cuckoo Search via Levy Flights. Mathematics, 2010:210 - 214.

[7] Yang X S. Flower Pollination Algorithm for Global Optimization[C]// International Conference on Unconventional Computation and Natural Computation. Springer-Verlag, 2013:240-249.

[8] Fausto F, Cuevas E, Valdivia A, et al. A global optimization algorithm inspired in the behavior of selfish herds[J]. Biosystems, 2017, 160:39-55.

[9] Hamilton, W.D., 1971. Geometry ot the selfish herd. J. Theor. Biol. 31 (2), 295–311.

[10] Tizhoosh H R. Opposition-Based Learning: A New Scheme for Machine Intelligence. Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on. IEEE, 2005:695-701.

[11] Rahnamayan S, Tizhoosh H R, Salama M M A. Opposition-Based Differential Evolution. Advances in Differential Evolution. Springer Berlin Heidelberg, 2008:64-79..

[12] Zhao R, Luo Q, Zhou Y. Elite Opposition-Based Social Spider Optimization Algorithm for Global Function Optimization[J]. Algorithms, 2017, 10(1):9.

[13] Zhou Y, Wang R, Luo Q. Elite opposition-based flower pollination algorithm[J]. Neurocomputing, 2016, 188:294-310.

[14] Tang K, Li X, Suganthan P N, et al. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization[J]. Nature Inspired Computation & Applications Laboratory, 2013.

[15] N. Hansen, A. Auger, S. Finck, R. Ros, Real-Parameter Black-Box Optimization Benchmarking 2009 Experimental Setup, Institute National de Recherche en Informatique et en Automatique (INRIA), 2009, Rapports de Recherche RR-6828, /http://hal.inria.fr/inria-00362649/en/S.

[16] MartÍ V, Robledo L M. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization[J]. Neural Computing & Applications, 2016, 27(2):495-513.

[17] Dervis Karaboga, Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization, 2007, 39(3):459-471.