Expert Systems With Applications 90 (2017) 484–500

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An improved Opposition-Based Sine Cosine Algorithm for global optimization

Mohamed Abd Elaziz[a,b], Diego Oliva[c,d], Shengwu Xiong[a,e,*]

[a] School of Computer Science and Technology, Wuhan University of Technology, Wuhan, China
[b] Department of Mathematics, Faculty of Science, Zagazig University, Zagazig, Egypt
[c] Departamento de Ciencias Computacionales, Universidad de Guadalajara, CUCEI Av. Revolucion 1500, Guadalajara, Jal, Mexico
[d] Cybernetics Institute, Tomsk Polytechnic University, Lenin Avenue 30, Tomsk, Russian Federation
[e] Hubei Collaborative Innovation Center of Basic Education Information Technology Services, Hubei University of Education, Wuhan, China

ARTICLE INFO

Article history:
Received 5 May 2017
Revised 21 July 2017
Accepted 22 July 2017
Available online 15 August 2017

Keywords:
Sine Cosine Algorithms (SCA)
Opposition-Based Learning (OBL)
Metaheuristic (MH)
Engineering problems

ABSTRACT

Real life optimization problems require techniques that properly explore the search spaces to obtain the best solutions. In this sense, it is common that traditional optimization algorithms fail in local optimal values. The Sine Cosine Algorithms (SCA) has been recently proposed; it is a global optimization approach based on two trigonometric functions. SCA uses the sine and cosine functions to modify a set of candidate solutions; such operators create a balance between exploration and exploitation of the search space. However, like other similar approaches, SCA tends to be stuck into sub-optimal regions that it is reflected in the computational effort required to find the best values. This situation occurs due that the operators used for exploration do not work well to analyze the search space. This paper presents an improved version of SCA that considers the opposition based learning (OBL) as a mechanism for a better exploration of the search space generating more accurate solutions. OBL is a machine learning strategy commonly used to increase the performance of metaheuristic algorithms. OBL considers the opposite position of a solution in the search space. Based on the objective function value, the OBL selects the best element between the original solution and its opposite position; this task increases the accuracy of the optimization process. The hybridization of concepts from different fields is crucial in intelligent and expert systems; it helps to combine the advantages of algorithms to generate more efficient approaches. The proposed method is an example of this combination; it has been tested over several benchmark functions and engineering problems. Such results support the efficacy of the proposed approach to find the optimal solutions in complex search spaces.

1. Introduction

Optimization is present in several fields of science and engineering, this is a process where the best solution of a specific problem is found using a search mechanism. In recent years a group of optimization approaches called metaheuristic has taken the attention of the scientific community. Metaheuristic Algorithms (MA) mimic a natural process to find the optimal solution. MA performs a stochastic search of the best parameters in an optimization problem. The main idea of these methods is the collective behavior that exists between the candidate solutions. These search agents interchange information about their positions in the search space. Using different operators that depend on the metaphor of each algorithm, the search agents are displaced to new positions where the probability to find optimal solutions is increased. The goal is to have a good balance between the exploration of the entire search space and the exploitation of the prominent regions. Several MA techniques have been developed in the last years, this fact is related to the no-free-lunch theorem that states that not all the optimization algorithms can be applied to the same problem (Wolpert & Macreadym, 1997). In other, words it is necessary to find the best MA that can be adapted to the real life problems to be solved. There have been proposed a different classification of MA, however, they commonly are divided into swarm algorithms and evolutionary algorithms (Mirjalili, 2015b). The main difference between them is that evolutionary algorithms use operators that imitate the process of mutation and crossover from the genetic theory (Mirjalili, 2015b).

On the other hand, swarm techniques simulate different behaviors from nature. For example, Particle Swarm Optimization

* Corresponding author.
E-mail addresses: abd_el_aziz_m@yahoo.com, meahmed@zu.edu.eg (M. Abd Elaziz), diego.oliva@cucei.udg.mx (D. Oliva), xiongsw@whut.edu.cn (S. Xiong).

http://dx.doi.org/10.1016/j.eswa.2017.07.043
0957-4174/

(PSO) is inspired by bird flocking and fish schooling (Kennedy & Eberhart, 1995). Artificial Bee Colony is another interesting MA where the operators are bees that are searching for food sources (Karaboga, 2005). New swarm approaches as Grey Wolf Optimizer (GWO) or Whale Optimization Algorithm (WOA) have been recently proposed (Mirjalili, Mirjalili, & Lewis, 2014), (Mirjalili & Lewis, 2016a). These methods mimic the hunting behavior of wolves and whales respectively. A considerable amount of literature has been published on MA. In the state-of-the-art, they have been proposed several approaches that simulate different processes from nature. For example, the Crow Search Algorithm (CSA) that emulates the behavior of crows to hide and stole food (Askarzadeh, 2016). Meanwhile, the Wind Driven Optimization (WDO) is based on the motion of the wind in the atmosphere (Bayraktar, Komurcu, Bossard, & Werner, 2013). Another interesting and popular approach is the Flower Pollination Algorithm (FPA) that is inspired in the process of transfer pollen between flowers (Yang, 2012). In this context, the Tree-Seed Algorithm has been developed by (Kiran, 2015), it is based on the relations between trees and their seeds. In (Cuevas, Dıaz Cortés, & Oliva Navarro, 2016) the Social Spider Optimization (SSO) is introduced as an alternative approach for global optimization. Another recently proposed approach is the Stochastic Fractal Search (Salimi, 2015), different to other MA this algorithm has not inspired in nature, and it considers a mathematic concept called the fractal to optimize complex problems. In this context, an interesting MA proposed in 2016 is the Sine Cosine Algorithm (SCA), it was introduced as an alternative for global optimization (Mirjalili, 2015b). The SCA uses the mathematical functions sine and cosine to perform the exploitation and exploration of the search space. The optimization process of SCA considers two elements of the set of candidate solutions. One of the selected elements affects the next position that the other element will take. In other words, the next position of one of the elements could be inside of a neighbor area of the other candidate solution or outside of this radio. The sine and cosine functions are used to compute the new positions using some variables that permit select one of both mathematical operators (sine or cosine). The SCA has been tested over a big amount of benchmark function showing good performance in comparison with similar approaches (Mirjalili, 2015b). In the same, context SCA has also been applied to the design of airfoil in order to verify its capabilities over real problems (Mirjalili, 2015b). Recently SCA has also been applied for in different problems like binarization of handwritten Arabic text (Mudhsh, Xiong, Abd ElAziz, Hassanien, & Duan, 2017) and for solving the unit commitment problem in energy production (Kaur S, 2016). Moreover and interesting implementation of SCA for detection of galaxies using image retrieval is presented in (Abd ElAziz, Selim, & Xiong, 2017). In this context, SCA has also been modified to solve multi-objective optimization problems (Tawhid & Savsani, 2017). Such methods are based on the standard version of SCA. The main drawback of SCA is that like other MA, its accuracy and convergence are affected by the calibration and randomness of some internal parameters this fact is similar to other MA. Some values should be selected according to the problem to be solved, and other settings are modified in the iterative process.

In the related literature, it has been presented an improved version SCA with an elitism strategy, and it is applied for feature selection in machine learning (Sindhu, Ngadiran, & Yacob, 2017). This version also includes a modified method for update the solutions. The disadvantage of this approach is that it includes an extra parameter that should be tuned, and it is desired that MA has less human interaction. Here is important to mention that since the SCA was introduced in 2015 there exist a few amount of applications, and researchers still looking for problems in which the features of SCA can be useful.

Metaheuristic algorithms are not perfect, some of them have several problems that affect their accuracy and performance. In order to avoid these situations, the Opposition-Based Learning (OBL) has been introduced, the OBL takes a candidate solution and generates their opposite position in search space (Tizhoosh, 2005). Using a single rule OBL verifies if the opposite value or the candidate solution has the best objective function value. Depending on the algorithm this process could be applied at initialization or when an operator modifies the set of feasible solutions. OBL has demonstrated is efficacy improving several MA. In (Bulbul, Pradhan, Roy, & Pal, 2015), the authors proposed the OBL Krill Herd (KH) algorithm for economic load dispatch problem. The Firefly (FF) algorithm has also been modified using OBL for numerical optimization problems (Verma, Aggarwal, & Patodi, 2016). The OBL has also increased the convergence speed of MA, for example, the Electromagnetism-Like Optimization (Cuevas, Oliva, Zaldivar, Cisneros, & Pajares, 2012). The Opposition-Based rule has also been used to estimate parameters in control engineering using the Shuffled Frog Leaping (SFL) algorithm (Ahandani & Alavi-Rad, 2015). The use of OBL has also been extended to multi-objective optimization (Ma et al., 2014). All of these works show that OBL is an interesting mechanism to achieve better results in optimization problems.

The aim of this paper is to introduce a modified version of SCA called Opposition-Based Sine Cosine Algorithm (OBSCA). The use of OBL in combination with the optimization features of SCA improves substantially the accuracy and performance of the standard SCA. Such improvement the affronts the disadvantages of the standard SCA, preserving the good optimization capabilities. The proposed OBSCA has been experimental tested over an extensive set of mathematical benchmark problems. Moreover, in order to prove that OBSCA is able to solve real-life optimization problems, it was tested over benchmark engineering problems. Comparisons with other similar approaches indicate that the OBSCA can provide better results in terms of accuracy and efficacy. The experiments and comparisons are supported by different metrics and statistical validations.

The rest of the paper is organized as follows. Section 2 introduces Preliminaries over, the standard SCA and Opposition-based Learning. The proposed OBSCA is introduced in Section 3. Meanwhile, in Section 4 are presented the experiments and comparisons. Finally Section 5 presents the conclusions.

2. Preliminaries

2.1. Sine Cosine Algorithm

The sine cosine algorithm is a new metaheuristic algorithm (Mirjalili, 2015b), the solutions are updated based on the sine or cosine function as in equations (1) or (2), respectively:

$$X_i = X_i + r_1 \times \sin(r_2) \times |r_3 P_i - X_i| \tag{1}$$

$$X_i = X_i + r_1 \times \cos(r_2) \times |r_3 P_i - X_i| \tag{2}$$

In general, from the previous two functions are combined into one function as in the following equation (Mirjalili, 2015b):

$$X_i = \begin{cases} X_i + r_1 \times \sin(r_2) \times |r_3 P_i - X_i| & if\ r_4 < 0.5 \\ X_i + r_1 \times \cos(r_2) \times |r_3 P_i - X_i| & if\ r_4 \geq 0.5 \end{cases} \tag{3}$$

Where P_i is the destination solution, X_i is the current solution, $|.|$ indicates the absolute value. r_1, r_2, r_3 and r_4 are random variables.

The parameter r_1 is random variable which responsible for determine the area of the next solution, this area may be either outside space between X_i and P_i or inside them. In (Mirjalili, 2015b) the authors update the parameter r_1 using (4) to balance exploration and exploitation.

$$r_1 = a - t\frac{a}{T} \tag{4}$$

where a is a constant, T is the maximum number of iterations and t is the current iteration.

The r_2 is random variable which used to find the direction of the movement of the next solution (i.e if it towards or outwards P_i). Also, the r_3 is random variable which gives random weights for P_i in order to stochastically emphasize ($r_3 > 1$) or deemphasize ($r_3 < 1$) the effect of desalination in defining the distance. The r_4 is used to switch between the sine and cosine functions as in Eq. (13). The steps of the SCA algorithm are given in Algorithm 1.

Algorithm 1 Sine Cosine Algorithm.

1: Initialize a set of search agents (solutions) (X)
2: **repeat**
3: Evaluate each of the search agents by the objective function
4: Update the best solution obtained so far ($P = X^*$)
5: Update $r_1, r_2, r_3,$ and r_4
6: Update the position of search agents using Eq. (13)
7: **until** ($t <$ maximum number of iterations)
8: Return the best solution obtained so far as the global optimum

2.2. Opposition-based learning

The Opposition-based Learning (OBL), which used to improve the convergence of metaheuristic methods to find the global solution of the optimization problem. In general, the metaheuristic methods start by generating initial population (that contains random solutions) as attempt to find the optimum solution(s), these initial solutions are generated randomly or based on prior knowledge such as specify the domain search or other criteria.

However, in the case of absence of this knowledge, these methods could not converge to an optimal solution, since they work randomly in the search space. Also, these approaches are time-consuming because of they depend on the distance between the initial solutions and the optimal solution. To solve this problem, the OBL provides a strategy to search for a solution in the opposite direction to the current solution, therefore, the solution becomes closer to optimal solution and the convergence becomes faster.

2.2.1. Opposite number

The opposite of real number $x \in [l, u]$ is given by $\bar{x}$ (Tizhoosh, 2005):

$$\bar{x} = u + l - x \tag{5}$$

where l and u are the lowest and upper bound of search space, respectively. In the multidimensional space the definition of x can be generalized as in (Tizhoosh, 2005). Suppose $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in R^n$, where $x_1, x_2, \ldots, x_n \in R$ and $x_j \in [l_j, u_j]$. The opposite point $\bar{\mathbf{x}} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]$ is defined by:

$$\bar{x}_j = u_j + l_j - x_j, \quad j = 1, 2, \ldots, n \tag{6}$$

2.2.2. Opposition-based optimization

In this optimization strategy, the opposite point $\bar{\mathbf{x}}$ is replaced with it is corresponding solution $\mathbf{x}$ based on the fitness function. If $f(\mathbf{x})$ is better than $f(\bar{\mathbf{x}})$ then x not changed, otherwise, $x = \bar{x}$, therefore, the solutions of population are updated based on the better value of $\mathbf{x}$ and $\bar{\mathbf{x}}$ (Cuevas et al., 2012).

An example of the opposition-based optimization is shown in Fig. 1, which represents the optimized function Fig. 1(a) and its corresponding contour plot Fig. 1(b). In Fig. 1(a), there are three random solutions from the initial population x_1, x_2 and x_3. The OBL concepts are applied to these solutions to generate a new $\bar{x}_1, \bar{x}_2$ and $\bar{x}_3$ which construct opposite population. The three best solutions are selected from both populations as solutions which construct a new population. From Fig. 1(a), it can be observed that the new three solutions are x_2, x_3 and $\bar{x}_1$, therefore, we can conclude that, compute the opposite solution for both x_2 and x_3 not improve the optimization. Also, the current solution x_1 will be removed from the next generation.

3. Proposed algorithm

In this section, the proposed algorithm is introduced which improve the performance of traditional SCA. The traditional SCA suffers from some drawbacks such as getting stuck in local optimal solution, slow convergence and time-consuming. These drawbacks are results from the fact that, some solutions are updated toward the best one (solution), at the same time there are some solutions that away from this solution. Therefore, the proposed algorithm avoid these drawbacks by taking the opposite direction into consideration.

Moreover, the OBSCA combines the search capabilities of the standard version of SCA with the OBL to increase the exploration of the search space. In comparison with similar algorithms, the proposed approach has fewer parameters to be tuned, as well as, the incorporation of OBL does not affect the configuration SCA and in contrast, the accuracy of the optimal solution is increased. In this way, it is also possible to reduce the size of the initial population that required to improve the convergence to the optimal solution, since OBSCA can broadly explore the search space. For example, if a problem requires a population of 200 initial solutions, the OBSCA can initialize 100 solutions in the standard way and then compute their opposite solutions (also, 100) using the OBL rule. After that only the best 100 solutions are selected to the optimization in the iterative process. However, the setting of the population in OBSCA can also affect the number of call functions required in the optimization process. Depending on the implementation an evaluation of the objective function represents more computational effort. This fact corresponds directly with No-Free-Lunch (NFL) theorem (Wolpert & Macreadym, 1997), that states that an algorithm cannot be improved without sacrifice any advantage. However, the NFL also mentioned that an optimization algorithm could not be used to solve all the optimization problems. This is the main motivation to develop the OBSCA.

The proposed algorithm improves the SCA through two stages: Firstly, the OBL is used in the initialization of population to enhancement the rate of convergence and avoid stuck in local optimal through searching the solutions in the whole domain of search. Secondly, in the updating stage of the population solution, the OBL is used, also, to check if the update in opposite direction is better than the current updated. The two stages are explained in details in the following subsections.

3.1. Initialization stage

The proposed method is starting by generating a random population X of size N, in which the solution $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$, $i = 1, 2, \ldots, N$, (this in this case of absence the a priori knowledge as mention in Section 2.2.1. Then OBL is used compute the opposite number for each solution, then the opposite population $\bar{X}$ is generated. Based on the two populations X and $\bar{X}$, the best N solution are selected. The steps of this stage are as follows:

1. Initialize the solutions of population X randomly.
2. Compute the opposite population $\bar{X}$ as:

   $$\bar{x}_{ij} = u_i + l_i - x_{ij}, \quad i = 1, 2, .., N \text{ and } j = 1, 2, .., n$$

   where x_{ij} and $\bar{x}_{ij}$ denote the ith point of the jth solution of X and its corresponding $\bar{X}$.
3. Choose the best N solutions from the union of two populations(i.e $X \bigcup \bar{X}$) to construct a new population.

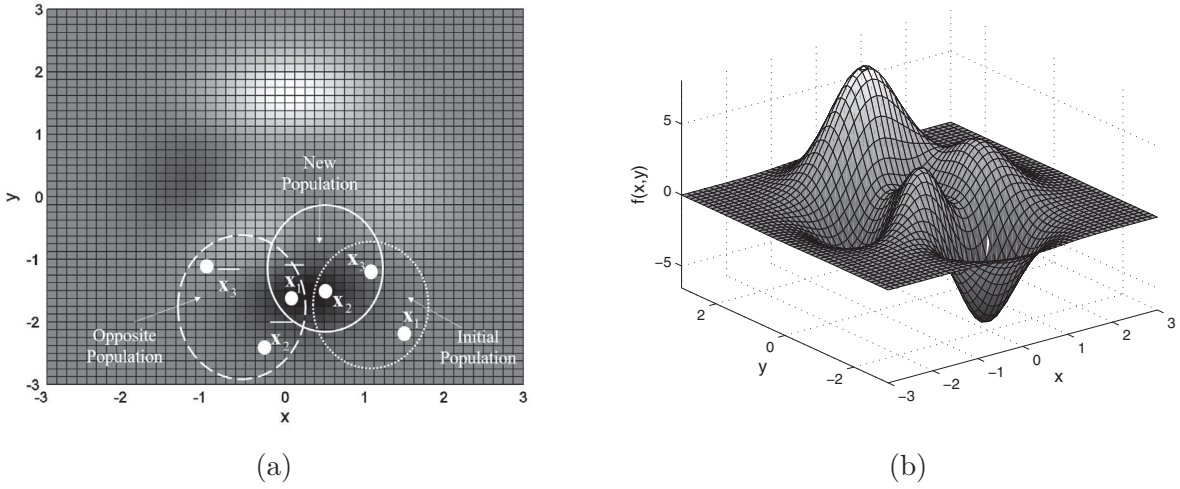

Fig. 1. The example of opposition-based optimization: (a) The contour plot of optimized function. (b) The optimized function.

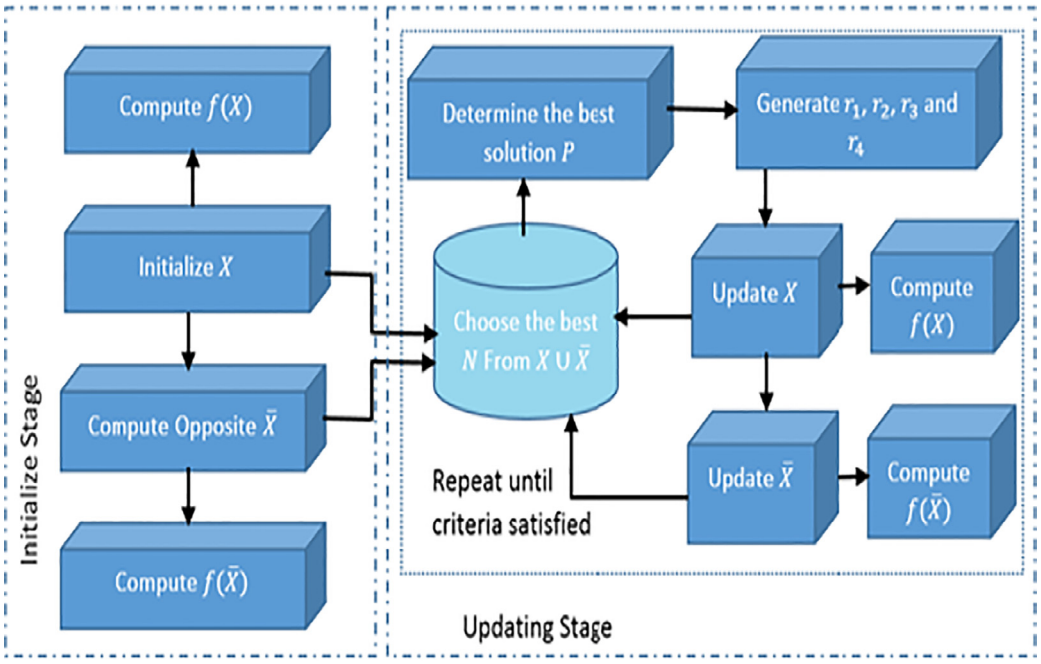

Fig. 2. The proposed OBSCA algorithm.

3.2. Updated stage

The OBSCA after selecting the best N solutions that generated a new population (from the initial stage), the best solution P is determined. The solutions in population X are updated using the SCA and their fitness functions are computed, also the opposite population $\bar{X}$ is computed based on the OBL and the fitness function for each $\bar{x}$ is evaluated. The next step in the OBSCA algorithm is to select the best N solutions from $X \bigcup \bar{X}$ based on their fitness functions. These previous steps are repeated until the stopping conditions are satisfied, the proposed OBSCA algorithm is illustrated in Fig. 2.

4. Experiments and discussion

The proposed method is an interesting alternative to the standard version of SCA for global optimization. The accuracy of the OBSCA is enhanced in comparison with the SCA and other similar approaches for complex problems. The experimental results demonstrate that the accuracy of the optimal solutions and abilities to explore the search space are superior in most of the cases. This fact occurs even with non-traditional benchmark function like the composite problems. Moreover, the application of OBSCA in engineering benchmark problems provides evidence of improvement of traditional SCA. The competitive results of OBSCA might encourage researchers to apply this technique to similar problematic.

4.1. Test functions

A comprehensive set of benchmark functions, including 29 different functions (that are divided into unimodal and multimodal), has been used to evaluate the performance of the proposed algorithm. The definition of the benchmark functions and their global optimum values are listed in Table 1. We referred to benchmark

Table 1
Benchmark of Functions.

ID	Equation	Lower	Upper	Dimension	type
F1	$f(x)=\sum_{i=1}^{n} x_i^2$	-100	100	10	Unimodal
F2	$f(x)=\sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	-10	10	10	Unimodal
F3	$f(x)=\sum_{i=1}^{n} (\sum_{j-1}^{i} x_i)^2$	-100	100	10	Unimodal
F4	$f(x)=max_i\{\lvert x_i\rvert, 1 \leqslant i \leqslant n\}$	-100	100	10	Unimodal
F5	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	-30	30	10	Unimodal
F6	$f(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	-100	100	10	Unimodal
F7	$f(x)=\sum_{i=1}^{n} ix_i^4+random[0,1]$	-1.28	1.28	10	Unimodal
F8	$f(x)=\sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i\rvert})$	-500	500	10	Multimodal
F9	$f(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	-5.12	5.12	10	Multimodal
F10	$f(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	-32	32	10	Multimodal
F11	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	-600	600	10	Multimodal
F12	$f(x)=\frac{\pi}{n}\{10sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$ $u(x_i,a,k,m)=\begin{cases} k(x_i-a)^m, & x_i>a \\ 0, & -a\leqslant x_i\leqslant a \\ k(-x_i-a)^m, & x_i<-a \end{cases}$	-50	50	10	Multimodal
F13	$f(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n} u(x_i,5,100,4)$	-50	50	10	Multimodal
F14	$f(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^{-1}$	-65.536	65.536	2	Multimodal
F15	$f(x)=(\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}])^2$	-5	5	4	Multimodal
F16	$f(x)=(4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-x_2^2+4x_2^4$	-5	5	2	Multimodal
F17	$f(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})cosx_1+10$	-5	5	2	Multimodal
F18	$f(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$ $\times[30+(2x_1-3x_2)\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	-2	2	2	Multimodal
F19	$f(x)=-\sum_{i=1}^{4} c_i\exp(-\sum_{j=1}^{3} a_{ij}(x_j-p_{ij})^2)$	1	3	3	Multimodal
F20	$f(x)=-\sum_{i=1}^{4} c_i\exp(-\sum_{j=1}^{6} a_{ij}(x_j-p_{ij})^2)$	0	1	6	Multimodal
F21	$f(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal
F22	$f(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal
F23	$f(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal
F24	$f_1,f_2,\ldots,f_{10}$ =Sphere Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[1,1,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[5/100,5/100,\ldots,5/100]$	-5	5	15	Composite
F25	$f_1,f_2,\ldots,f_{10}$ =Griewanks Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[1,1,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[5/100,5/100,\ldots,5/100]$	-5	5	15	Composite
F26	$f_1,f_2,\ldots,f_{10}$ =Griewanks Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[1,1,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[1,1,\ldots,1]$	-5	5	15	Composite
F27	f_1,f_2 = Rastrigins Function f_3,f_4 =Weierstrass Function f_5,f_6 =Griewanks Function f_7,f_8 =Ackleys Function f_9,f_{10} =Sphere Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[1,1,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[5/32,5/32,1,1,5/0.5,5/0.5,5/100,5/100,5/100,5/100]$	-5	5	15	Composite
F28	f_1,f_2 = Rastrigins Function f_3,f_4 =Weierstrass Function f_5,f_6 =Griewanks Function f_7,f_8 =Ackleys Function f_9,f_{10} =Sphere Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[1,1,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[1/5,1/5,5/0.5,5/0.5,5/100,5/100,5/32,5/32,5/100,5/100]$	-5	5	15	Composite
F29	f_1,f_2 = Rastrigins Function f_3,f_4 =Weierstrass Function f_5,f_6 =Griewanks Function f_7,f_8 =Ackleys Function f_9,f_{10} =Sphere Function $\sigma_1,\sigma_2,\ldots,\sigma_{10}=[0.1,0.2,\ldots,1]$ $\lambda_1,\lambda_2,\ldots,\lambda_{10}=[0.1\times1/5,0.2\times1/5,\ldots,1\times1/5]$	-5	5	15	Composite

*Fixed-dimension multimodal

functions by character "*F*" and its number such as *F1, F2, F3,... , F29*. The functions from *F1* to *F7* are unimodal optimization problems which have one extreme value point within a given search area. These functions are used to investigate the convergence speed and optimization precision of the algorithms, while the functions from *F8* to *F23* are multimodal optimization functions which have more than one local one extreme value points within a given search area. They are used to evaluate the abilities to jump out of local optimum and seeking the global excellent result. The rest of benchmark functions (*F24* to *F29*) are composite functions introduced by CEC 2005 special session (Suganthan et al., May 2005), (Liang, Suganthan, & Deb, 2005), these functions are a combination of the unimodal and multimodal functions with different shift, rotation, and biasing. The composite functions are similar to other real search spaces and they are used to evaluate the performance of the algorithms to balance between the exploitation and exploration.

The number *N* of agents in the population is fixed to 30 corresponding to the dimension 30 and the maximum number of iterations is 500. The experiments are implemented over "Windows 7 (64bit)" that runs on "CPU Core2 Duo with 4GB ram"; and "Matlab 2014b" software is used. For statistical analysis, all the

Table 2
Comparison between SCA and OBSCA over 11 fitness functions.

	NFC_{SCA}	NFC_{OBSCA}	AR	RSR
F1	1676	807	2.08	0.79
F2	1573	693	2.27	0.80
F3	2418	1330	1.82	0.70
F4	2409	1292	1.87	0.70
F5	-	-	-	-
F6	-	-	-	-
F7	-	-	-	-
F8	-	-	-	-
F9	2025	913	2.22	0.73
F10	2416	904	2.67	0.63
F11	2185	1249	1.75	0.75
Aver.			1.15	

Table 3
The results of test the Influence of Dimensionality.

	dim = 2				dim = 10			
	NFC_{SCA}	NFC_{OBSCA}	AR	RSR	NFC_{SCA}	NFC_{OBSCA}	AR	RSR
F1	67	31	2.13	0.99	1292	427	3.02	0.81
F2	146	48	3.03	0.98	1241	412	3.01	0.82
F3	83	29	2.81	0.99	1800	872	2.07	0.78
F4	192	57	3.35	0.97	1796	883	2.03	0.78
F5	-	-	-	-	-	-	-	-
F6	-	-	-	-	-	-	-	-
F7	-	-	-	-	-	-	-	-
F8	-	-	-	-	-	-	-	-
F9	79	31	2.54	0.99	1602	526	3.04	0.76
F10	185	58	3.21	0.97	1446	561	2.58	0.80
F11	274	88	3.13	0.96	2032	946	2.15	0.73
Aver.			1.93	0.78			1.63	0.50
	dim = 20							
	NFC_{SCA}	NFC_{OBSCA}	AR	RSR				
F1	1936	1020	1.90	0.77				
F2	1790	892	2.01	0.78				
F3	3900	1632	2.39	0.33				
F4	3140	1561	2.01	0.54				
F5	-	-	-	-				
F6	-	-	-	-				
F7	-	-	-	-				
F8	-	-	-	-				
F9	2613	1097	2.38	0.61				
F10	3720	1140	3.26	0.33				
F11	2363	1350	1.75	0.72				
Aver.			1.43	0.37				

algorithms performed 30 independent runs over each benchmark problem.

4.2. Measures of performance

The performance of the algorithms is measured using some statistical metrics computed based on the fitness (Suganthan et al., May 2005):

1) Mean of fitness values:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{7}$$

2) Standard deviation (STD):

$$STD = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(v_i - \mu)^2} \tag{8}$$

where N is the total number of elements evaluated, v_i is a fitness value and μ is calculated by Eq. (7).

3) The Acceleration Rate (AR):

$$AR = \frac{NFC_{SCA}}{NFC_{OBSCA}} \tag{9}$$

where *NFC* is number of function calls (NFCs) and if $AR > 1$ this means that OBSCA is faster. Further, the average acceleration rate

Table 4
The results of test the Influence of Population Size.

	Pop = 50				Pop = 100			
	NFC_{SCA}	NFC_{OBSCA}	AR	RSR	NFC_{SCA}	NFC_{OBSCA}	AR	RSR
F1	4709	4252	1.11	0.80	5339	4274	1.25	0.80
F2	5100	4391	1.16	0.82	5379	4364	1.23	0.81
F3	4653	3597	1.29	0.75	4787	3747	1.28	0.73
F4	4591	3653	1.26	0.77	4765	3498	1.36	0.73
F5	-	-	-	-	-	-	-	-
F6	-	-	-	-	-	-	-	-
F7	-	-	-	-	-	-	-	-
F8	-	-	-	-	-	-	-	-
F9	5097	3953	1.29	0.76	5192	4012	1.29	0.72
F10	4981	4075	1.22	0.79	5155	3968	1.30	0.77
F11	3985	3678	1.08	0.75	4093	3805	1.08	0.78
Aver.			1.20	0.49			1.26	0.48
	Pop = 150				Pop = 200			
	NFC_{SCA}	NFC_{OBSCA}	AR	RSR	NFC_{SCA}	NFC_{OBSCA}	AR	RSR
F1	5408	4387	1.23	0.74	5550	4584	1.21	0.77
F2	5449	4418	1.23	0.76	5761	4640	1.24	0.78
F3	4876	3794	1.29	0.65	5095	3825	1.33	0.60
F4	4906	3506	1.40	0.66	5143	3780	1.36	0.64
F5	-	-	-	-	-	-	-	-
F6	-	-	-	-	-	-	-	-
F7	-	-	-	-	-	-	-	-
F8	-	-	-	-	-	-	-	-
F9	5427	4138	1.31	0.68	5767	4347	1.33	0.73
F10	5468	4058	1.35	0.49	5773	4151	1.39	0.41
F11	4918	3925	1.25	0.73	5484	4036	1.36	0.73
Aver.			1.29	0.43			1.32	0.42

Table 5
The parameters of algorithm and their values.

Algorithm	Parameters	Value
SCA	a	2
HS	HMCR	0.7
	pitch adjusting rate for each generation (PAR)	0.3
	minimum pitch adjusting rate (PAR_{min})	0.3
	maximum pitch adjusting rate (PAR_{max})	0.9
	minimum bandwidth (bw_{min})	0.2
	maximum bandwidth	0.5
ABC	maximum cycle number	1000
	modification rate (MR)	0.8
MFO	b	1
	l	[-1, 1]
SSO	Probabilities of attraction or repulsion (pm)	0.7
	Lower Female Percent	65
	Upper Female Percent	90
MVO	Minimum of Wormhole Existence Probability	0.2
	Maximum of Wormhole Existence Probability	1
PSO	Maximum Inertia weight (w_{max})	0.9
	Minimum Inertia weight (w_{min})	0.4
	Maximum velocity (V_{max})	1.0
	Minimum velocity (V_{min})	-1.0
	Cognitive coefficient (C_1)	1.429
	Cognitive coefficient (C_2)	1.429

AR_{avg} over n test functions are calculated as follows:

$$AR_{avg} = \frac{1}{n}\sum_{i=1}^{n} AR_i \tag{10}$$

4) The number of times, for which the algorithm successfully reaches the *NVTR* for each test function is measured as the Ratio success rate (RSR):

$$RSR = \frac{SR_{SCA}}{SR_{OBSCA}}, \quad SR_{SCA} = \frac{NVTR}{total\ number\ of\ trails} \tag{11}$$

where *NVTR* is number times reached to the value-to-reach (VTR).

4.3. Comparison between SCA and OBSCA

First of all, it is necessary to compare the standard SCA with the proposed OBSCA in terms of convergence speed and robustness. The comparison results of SCA with OBSCA are shown in

Table 6
The average of unimodal and multimodal functions function values for all algorithms.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA	fmin
F1	2.86E-6	1.67E3	5.49E-6	1.82E-74	1.90E-1	1.08E0	8.06E-5	9.78E2	5.43E0	0
F2	5.69E-2	2.88E1	1.49E-1	1.09E-42	2.05E0	1.88E3	5.42E-3	6.33E0	2.37E-2	0
F3	6.36E1	2.30E4	8.41E1	2.05E1	1.14E2	2.49E2	1.73E4	1.66E4	1.02E4	0
F4	2.18E0	6.91E1	2.38E0	3.22E-32	2.09E0	2.31E0	4.06E1	1.56E1	3.63E1	0
F5	5.17E1	2.67E6	5.53E1	2.82E1	4.54E1	2.98E2	4.65E1	2.24E5	6.30E4	0
F6	1.16E-6	3.51E3	8.15E-6	4.70E0	1.91E-1	1.18E0	8.26E-5	9.37E2	1.43E1	0
F7	2.92E-2	3.42E0	2.55E-2	2.13E-4	3.82E-1	3.35E-2	2.93E-1	5.04E1	1.35E-1	0
F8	-6.06E3	−8.47E3	−6.56E3	−3.53E3	−8.90E3	−7.65E3	−1.16E4	−1.13E4	−3.70E3	−415*5
F9	4.58E1	1.74E2	5.21E1	0.00E0	7.53E1	1.31E2	4.73E0	8.56E1	4.90E1	0
F10	1.52E0	1.40E1	1.29E0	8.88E-16	4.85E-1	1.69E0	1.28E-1	6.50E0	1.66E1	0
F11	2.69E-2	1.61E1	2.86E-2	0.00E0	1.04E-2	8.57E-1	1.74E-2	7.38E0	8.88E-1	0
F12	1.56E-1	8.95E0	8.64E-2	5.72E-1	3.27E0	2.13E0	2.35E2	4.69E3	2.89E4	0
F13	7.83E-2	2.11E3	3.75E-1	2.41E0	1.14E-1	1.78E-1	2.38E-4	3.22E5	6.75E4	0
F14	3.40E0	2.54E0	4.17E0	2.64E0	2.98E0	1.49E0	9.98E-1	1.03E0	2.18E0	1
F15	1.88E-3	2.25E-3	2.60E-3	6.58E-4	7.45E-4	4.84E-3	8.65E-4	3.35E-4	1.08E-3	0.0003
F16	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.03E0	−1.0316
F17	3.98E-1	3.98E-1	3.98E-1	3.99E-1	3.98E-1	3.98E-1	3.98E-1	4.00E-1	4.00E-1	0.398
F18	3.00E0	3.00E0	3.00E0	3.00E0	3.00E0	3.00E0	3.00E0	3.10E0	3.00E0	3
F19	−3.00E-1	−3.00E-1	−3.00E-1	−3.00E-1	−2.86E-1	−3.00E-1	−3.00E-1	−3.00E0	−3.00E-1	−3.86
F20	−3.29E0	−3.22E0	−3.27E0	−3.10E0	−3.31E0	−3.29E0	−3.32E0	−3.05E0	−3.04E0	3.32
F21	−6.24E0	−6.24E0	−5.73E0	−9.06E0	−9.49E0	−9.89E0	−9.77E0	−8.64E0	−2.20E0	−10.1532
F22	−8.33E0	−7.65E0	−6.49E0	−9.93E0	−1.04E1	−8.07E0	−1.04E1	−7.35E0	−4.27E0	−10.4028
F23	−8.21E0	−6.51E0	−8.03E0	−1.01E1	−1.05E1	−8.51E0	−1.05E1	−4.84E0	−3.34E0	−10.5363

Table 7
Standard deviation of unimodal and multimodal functions for all algorithms.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA
F1	1.23E-5	3.79E3	2.08E-5	2.90E-11	8.47E-17	2.85E-1	7.03E-5	2.89E3	1.54E1
F2	6.95E-2	2.06E1	2.03E-1	2.44E-12	9.03E-16	9.89E3	2.55E-3	1.15E1	4.35E-2
F3	6.76E1	1.09E4	8.95E1	2.64E0	2.89E-14	1.02E2	4.46E3	1.84E4	6.38E3
F4	1.03E0	7.65E0	9.39E-1	1.19E-1	9.03E-16	1.32E0	6.43E0	8.81E0	1.38E1
F5	3.04E1	1.46E7	4.31E1	1.80E-1	1.45E-14	4.93E2	1.80E1	7.85E5	1.98E5
F6	3.92E-6	4.75E3	2.26E-5	3.32E-1	8.47E-17	2.79E-1	8.15E-5	1.09E3	1.49E1
F7	1.60E-2	5.09E0	1.29E-2	2.87E-3	2.26E-16	1.36E-2	4.91E-2	2.71E1	1.60E-1
F8	1.00E3	1.10E3	7.86E2	2.74E2	5.55E-12	7.94E2	1.93E2	2.29E3	3.03E2
F9	1.35E1	3.96E1	2.28E1	2.08E-9	1.45E-14	3.22E1	1.79E0	4.27E1	4.06E1
F10	8.02E-1	7.57E0	7.53E-1	2.07E0	3.95E-16	5.81E-1	1.35E-1	3.90E0	7.12E0
F11	3.84E-2	3.40E1	2.93E-2	7.00E-2	0.00E0	9.08E-2	8.80E-3	1.00E1	3.13E-1
F12	2.85E-1	5.14E0	1.47E-1	1.80E-1	4.52E-16	1.14E0	2.38E-5	2.48E4	1.07E5
F13	2.00E-1	1.11E4	8.50E-1	1.69E-1	0.00E0	9.44E-2	2.74E-4	8.22E5	1.98E5
F14	2.74E0	2.33E0	3.52E0	3.11E0	4.52E-16	7.26E-1	3.72E-12	1.81E-1	2.49E0
F15	5.04E-3	4.93E-3	6.04E-3	2.83E-4	3.31E-19	7.93E-3	1.71E-4	8.38E-6	3.78E-4
F16	6.45E-16	6.78E-16	6.45E-16	8.51E-6	9.03E-16	4.77E-7	2.05E-11	2.98E-3	4.46E-5
F17	0.00E0	0.00E0	0.00E0	6.55E-4	0.00E0	1.96E-7	9.20E-11	2.10E-3	1.43E-3
F18	1.22E-15	1.48E-15	1.11E-15	6.54E-5	0.00E0	3.05E-6	1.42E-3	1.29E-1	1.56E-4
F19	2.26E-16	2.26E-16	2.26E-16	2.26E-16	0.00E0	2.26E-16	2.26E-16	9.61E-1	2.26E-16
F20	5.35E-2	5.87E-2	5.92E-2	3.94E-2	0.00E0	5.70E-2	7.49E-11	1.02E-1	1.16E-1
F21	3.74E0	3.57E0	3.64E0	1.76E0	3.61E-15	3.14E0	1.95E-2	2.68E0	1.71E0
F22	3.26E0	3.50E0	3.79E0	2.73E-1	5.42E-15	3.20E0	1.36E-2	3.80E0	1.43E0
F23	3.41E0	3.64E0	3.40E0	2.56E-1	0.00E0	3.23E0	1.87E-2	3.25E0	1.78E0

Tables 2 to solve 11 benchmark functions $F1 - F11$ that were previously defined in Table 1. The (AR_{avg}) on 11 test functions is shown in the last row of the table.

OBSCA outperforms SCA on 11 test functions, average acceleration rate (AR_{avg}) is 1.15, which means OBSCA is on average 15% faster than SCA. Both algorithms fail to solve, F_5, F_6, F_7 and F_8.

4.4. Influence of dimensionality

In order to analyze the effect of the problem dimensionality, the same experiments in the previous section are repeated for $D = 2, 10$ and 20 using the previous 11 functions (the other parameters are fixed without any changed). The results for 11 test functions are illustrated in Table 3 using AR and RSR measures.

From this table is possible to conclude that, the OBSCA outperforms the SCA, however, both of them not reach to the optimal value of F_5, F_6, F_7 and F_8 functions before meeting the maximum NFCs. The average AR is equal to 1.93, 1.63 and 1.42 for $D = 2, 10$ and 20 respectively, these values indicate that OBSCA performs faster than SCA. The average of RSR for $D = 2, 10$ and 20 is 0.78, 0.50 and 0.37 respectively. In general, for 11 functions the AR is increased for both algorithms, however, the results of OBSCA are more desirable that results of SCA for all functions overall dimension. In Table 3, the average RSRs has presented which the results indicate that RSR has decreased with increase the dimension for both algorithms. Whenever the dimension equal to the RSR is better than other two dimensions ($D = 10$ and $D = 20$).

Concluded from the previous results that decreasing the overall RSR for SCA and OBSCA was observed because by increasing the problem dimension, both algorithms are sometimes unable to solve the problem before reaching the maximum NFCs. However, as seen, OBSCA performs better for high-dimensional problems.

Table 8
The elapsed time of unimodal and multimodal functions.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA
F1	67.96	11.45	61.65	9.32	129.88	120.88	57.91	306.87	24.89
F2	82.93	12.06	66.93	10.31	120.15	117.87	58.73	307.61	25.84
F3	129.17	51.48	100.73	49.82	166.00	181.94	144.89	348.08	104.82
F4	64.43	13.82	62.22	12.26	123.62	136.59	64.57	4440.41	29.47
F5	81.29	14.87	64.65	13.34	127.96	129.00	64.58	321.51	30.97
F6	74.13	14.85	64.97	14.23	126.90	139.46	61.88	315.50	29.87
F7	67.38	18.46	62.26	21.13	120.49	135.09	70.28	313.56	37.77
F8	73.59	15.41	65.24	18.62	128.56	115.39	95.75	314.19	31.61
F9	76.01	15.30	61.69	17.93	117.74	142.01	96.77	317.01	31.63
F10	77.97	16.46	63.91	19.06	122.05	137.05	97.23	313.30	33.71
F11	82.44	22.97	70.06	17.18	125.13	136.38	102.03	312.77	36.92
F12	98.43	34.80	85.52	30.94	144.01	136.53	123.51	1256.60	59.78
F13	111.08	37.22	81.68	33.48	138.93	134.89	134.40	328.56	64.23
F14	128.71	93.24	119.26	79.16	178.89	76.71	224.65	86.63	157.52
F15	67.95	19.93	64.43	18.62	94.23	29.14	104.68	54.86	32.91
F16	62.91	14.21	55.71	13.36	87.53	19.01	89.82	30.28	25.17
F17	73.24	13.09	58.68	13.77	86.36	20.96	89.44	29.72	27.95
F18	73.31	14.53	55.24	13.52	84.51	20.06	89.87	30.17	25.29
F19	74.15	25.15	61.64	22.25	90.28	26.92	102.80	47.70	38.27
F20	73.36	25.17	63.27	23.46	95.00	35.51	102.53	78.79	37.62
F21	81.66	29.66	62.98	25.50	95.40	31.72	107.58	59.98	43.77
F22	85.12	34.24	66.45	29.38	99.91	34.24	112.17	61.77	47.57
F23	87.92	38.32	70.25	34.66	104.26	38.67	120.38	65.29	58.49

4.5. Influence of population size

In order to illustrate the influence of the population size, the experiments are repeated using the following values of population $Pop = 50, 100, 150$ and 200 for the same eleven functions without changed other parameters. Table 4 shows the results for both algorithms based on different population sizes.

From this table we can observe that the average AR is equal to 1.20, 1.26, 1.29 and 1.32 for $Pop = 50, 100, 150$ and 200 respectively, and this indicate that OBSCA performs faster than SCA. The average RSR for $Pop = 50, 100, 150$ and 200 is 0.49 ,0.48, 0.43 and 0.42 respectively. Like in the Section 4.4 both algorithms not achieve the optimal value for functions F_5, F_6, F_7 and F_8 before meeting the maximum NFCs.

4.6. The comparison between OBSCA and other algorithms

4.6.1. Parameters of algorithms

For a fair comparison, the improved algorithm with classic SCA and seven optimization algorithms, namely, particle swarm optimization (PSO), Opposite-PSO (OBPSO) (Wang, Li, Liu, Li, & Zeng, 2007), Moth-flame Optimization (MFO) (Mirjalili, 2015a), Social-Spider Optimization (SSO) (Abd ElAziz & Hassanien, 2017), (Cuevas, Cienfuegos, ZaldıVar, & PéRez-Cisneros, 2013), Artificial Bee Colony (ABC) (Karaboga, 2005), Multi-Verse Optimizer (MVO) (Mirjalili, Mirjalili, & Hatamlou, 2016) and Harmony Search (HS) (Lee & Geem, 2005). In this study, their parameters are set using the same criteria for all of them to test unimodel, multimodel and composite functions. The parameters settings of these algorithms in our experiments tabulated in Table 5.

4.6.2. Unimodal and multimodal functions

The comparison results for all the algorithms using unimodal and multimodal functions are presented in Tables 6–7 and Figs. 3, 4, 5, 6. Tables 6–7 display the average, the standard division of fitness values and Time (in seconds) obtained by all algorithms over 30 runs, respectively.

From Table 6 we can observed that OBSCA obtained the best mean values in *F1-F5, F7-F11* and *F23*. The PSO outperformed other algorithms in *F12*; whereas, OBPSO yielded the better value in *F6*. The ABC achieve the best value for *F13, F14* and *F20*, also, HS in *F15*. The best mean value of *F18* recorded by OBPSO, MFO and PSO, the *F19* achieved by all algorithm (expect SSO) and for *F21-F22* the OBSCA. It is possible to conclude that the results obtained by SSO and ABC are better than the results that computed by other algorithms.

Table 7 displayed the standard divisions for all algorithms. The SSO is the best algorithm overall all functions, then the ABC algorithm in the second rank (*F8, F10-F14, F20-F23*) followed by OBSCA algorithm (*F1-F5, F7, F9* and *F19*). The PSO achieved the minimum values in function *F18*, whereas, for *F17* the OBPSO, MVO, SSO and PSO are the better.

Table 8 displayed the elapsed time taken by each algorithm. The OBSCA is the best algorithm for *F1-F6, F11-F16* and *F18-F23*, then the MFO algorithm in the second rank (*F7-F10*) and *F17*.

From Tables 6–7 and Figs. 3–6, it is possible to conclude that the OBSCA proved efficiency and effectiveness in finding the global optimal values of optimization problems with fast convergence rate.

4.6.3. Composite functions

In this subsection, the comparison results of the proposed method with the other algorithms are given in Tables 9, 10, 11 and Fig. 7. From these results, it can be see that the proposed OBSCA is outperform than other algorithms, however, there exists some functions the results of the OBSCA is very close the other algorithms such as *F27* and *F29*. As presented in Table 6, the results of the algorithms on composite test functions are very close. This is due to the difficulty of this set of test functions. The results prove that the MVO algorithm provides highly competitive results on the composite test functions as well. This evidences that the MVO algorithm properly balances exploration and exploration during optimization. This is firstly originated from the employed adaptive WEP that assists to smoothly transit between exploration and exploitation phases. This illustrates that OBSCA balances between exploitation and exploration, these results from using the OBL strategy. As mentioned above, the OBL increases the convergence performance of the proposed method through avoiding the stuck local point.

From all the previous results it can be concluded that the proposed OBSCA method has a better ability for exploration and exploitation due to the solutions are updated using the Eq. (13) that

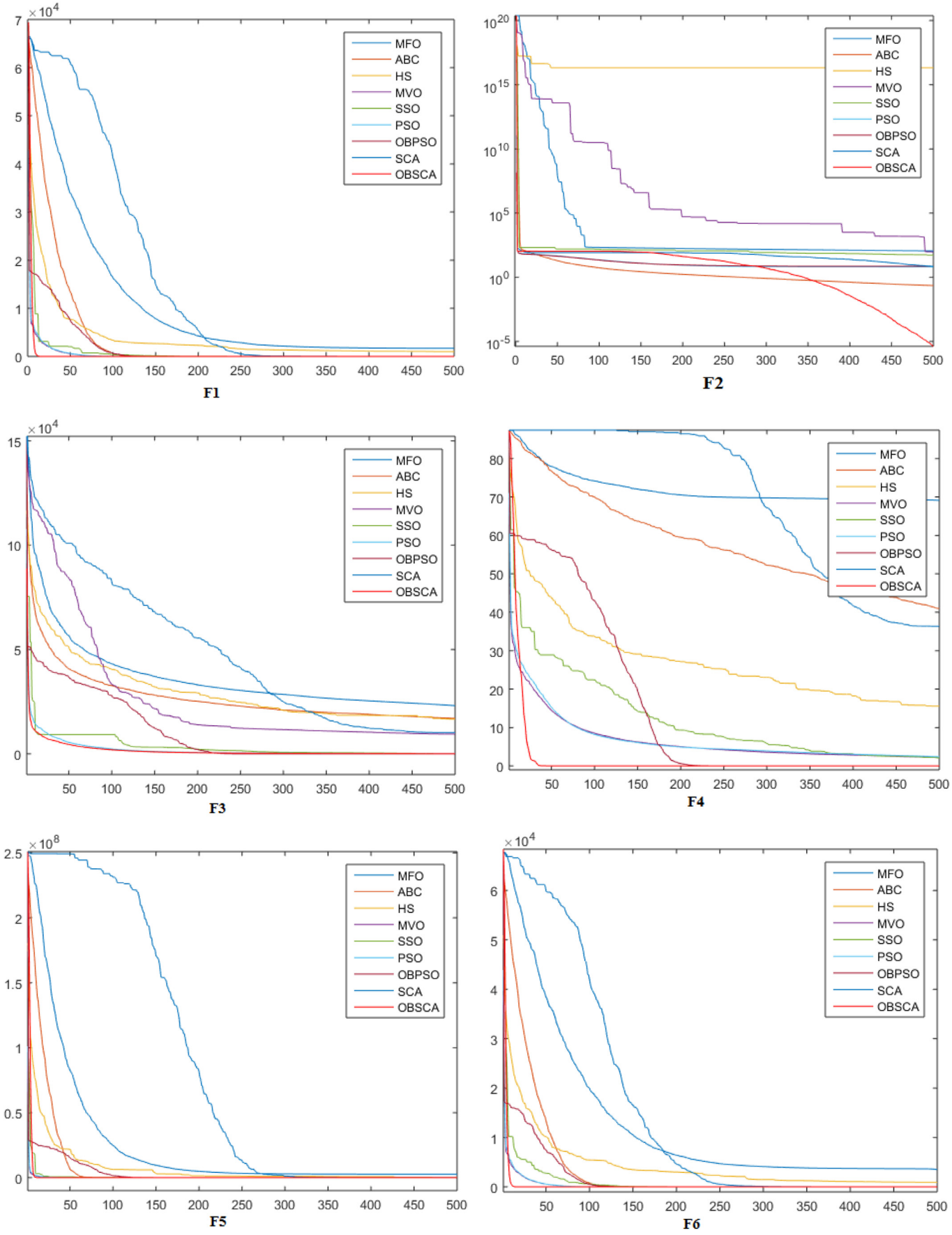

Fig. 3. Convergence curves for fitness function from F1 to F6.

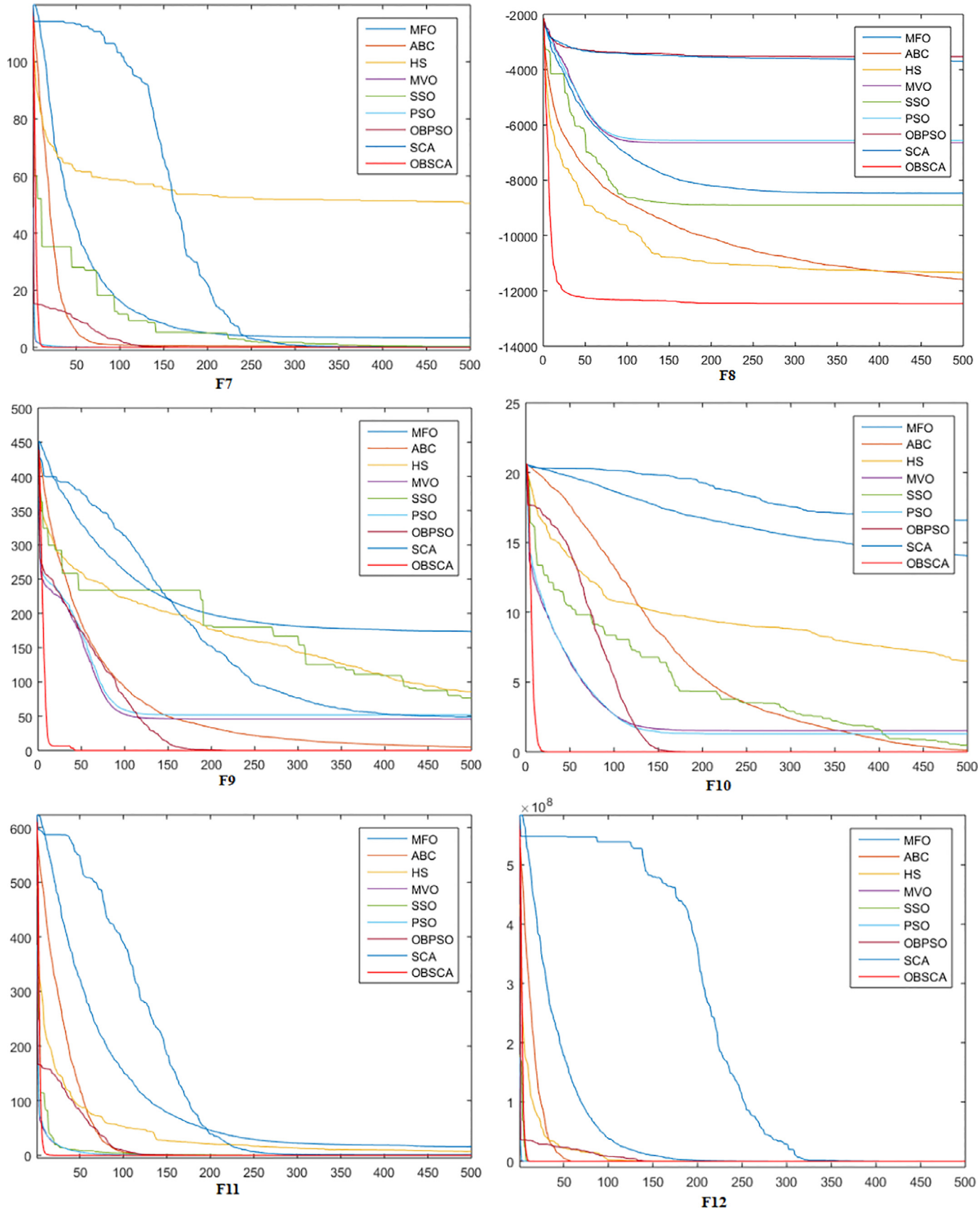

Fig. 4. Convergence curves for fitness function from F7 to F12.

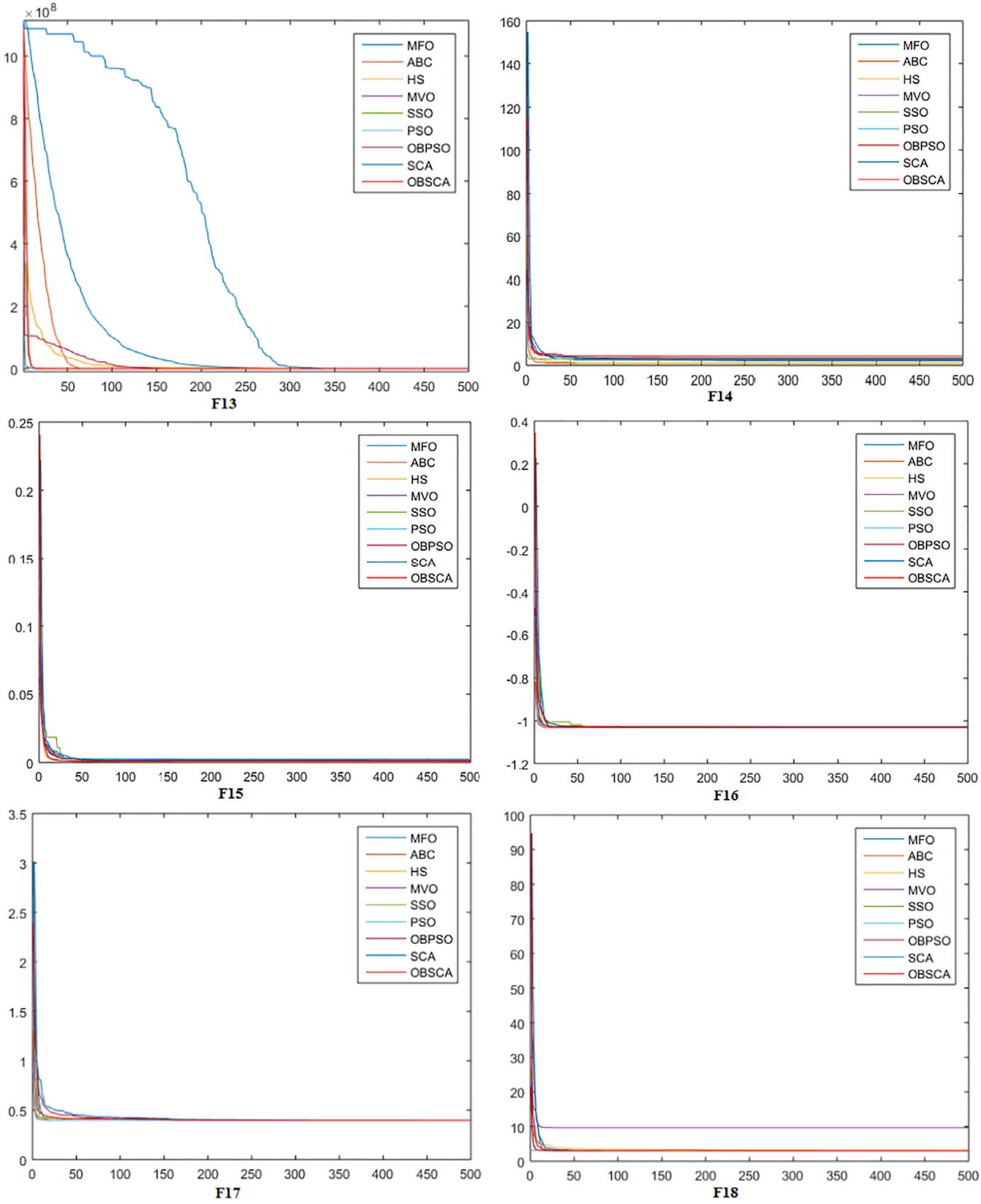

Fig. 5. Convergence curves for fitness function from F13 to F18.

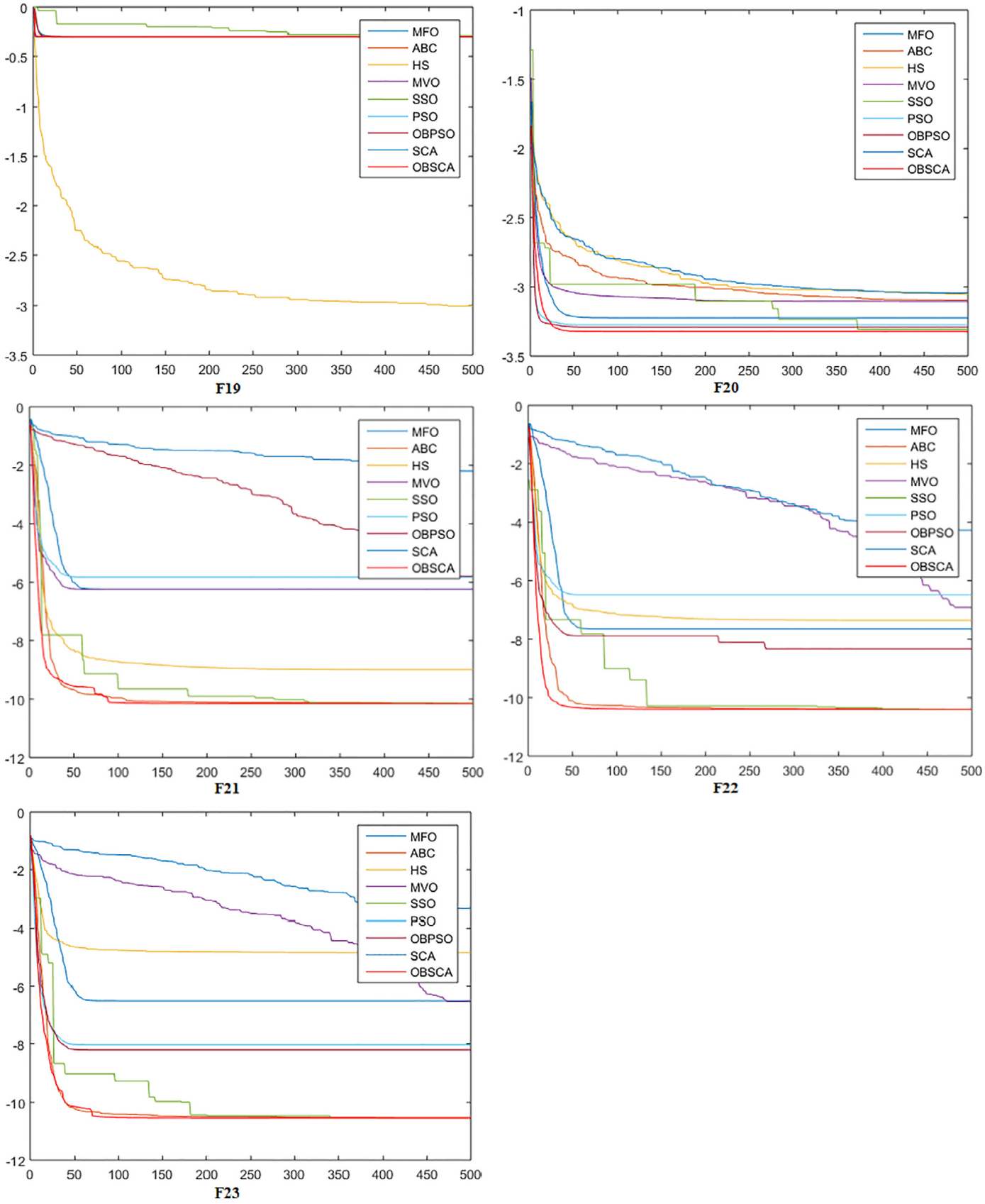

Fig. 6. Convergence curves for fitness function from F19 to F23.

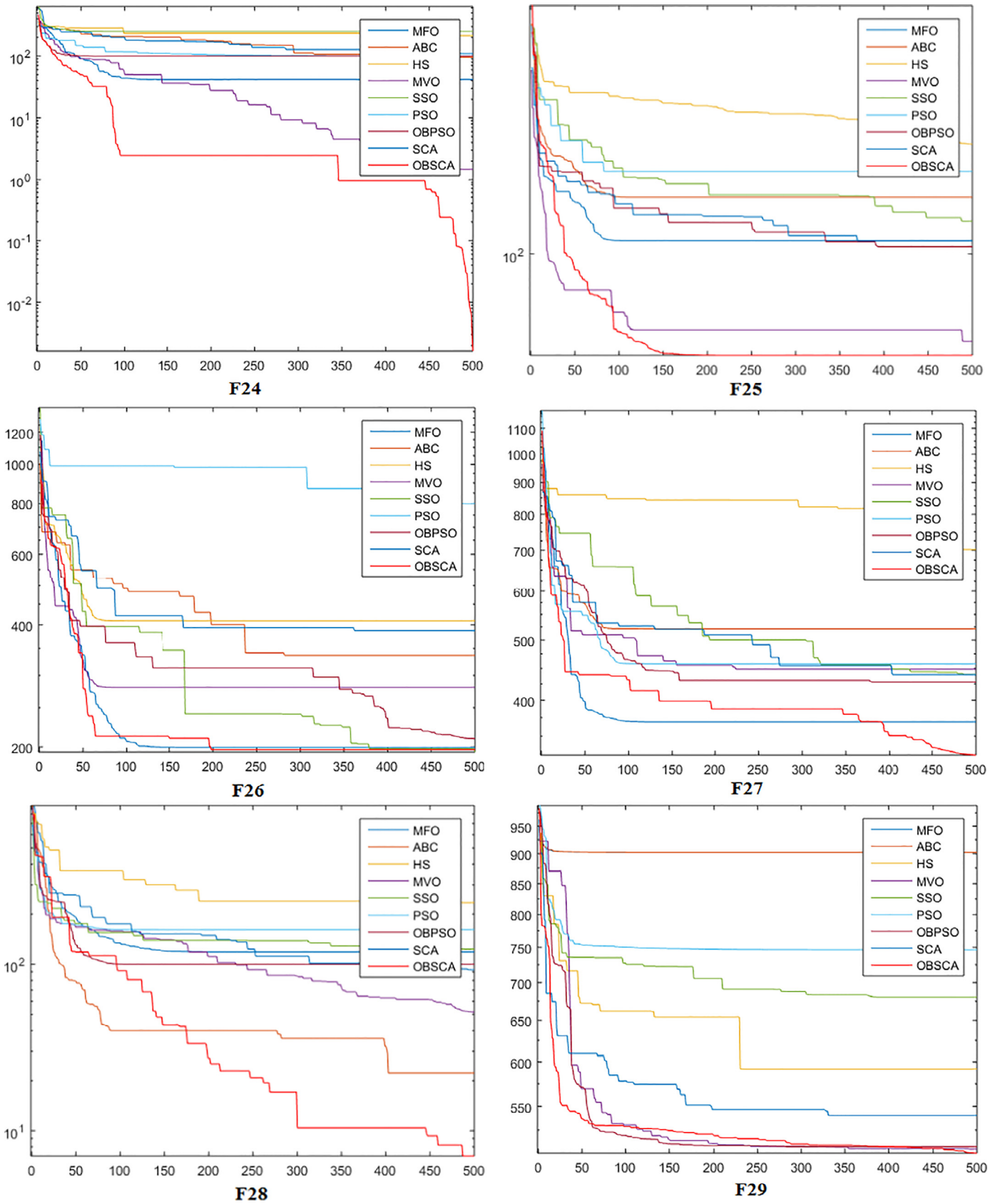

Fig. 7. Convergence curves for fitness function from F24 to F29.

Table 9
The average of fitness composite functions values for all algorithms.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA	fmin
F24	20.00	33.34	100.40	0.9171	80.00	100.17	180.00	102.27	192.13	0
F25	110.77	150.45	169.16	30.95	153.02	133.45	165.46	222.00	149.42	0
F26	240.86	182.23	435.17	89.75	126.60	238.06	294.80	474.28	306.77	0
F27	410.78	369.13	457.92	326.36	440.30	448.79	521.45	726.22	439.91	0
F28	22.22	118.67	161.93	7.04	123.50	51.68	100.00	234.57	92.57	0
F29	505.73	508.09	802.51	501.68	746.62	540.28	902.53	680.35	592.013	0

Table 10
Standard deviation for all algorithms overall composite functions.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA
F24	30.38	45.65	83.66	1e-3	12.54	44.72	0.59	37.03	21.68
F25	15.84	14.50	14.39	11.98	13.76	17.02	14.19	24.51	18.41
F26	38.95	19.76	54.45	8.17	20.30	43.77	21.54	47.82	11.75
F27	111.08	12.17	35.14	33.17	53.16	43.83	24.53	55.69	22.55
F28	10.00	151.93	87.59	28.13	38.36	68.40	8.11	117.19	14.27
F29	21.1183	220.45	50.27	25.19	60.62	39.15	41.01	35.52	31.68

Table 11
The elapsed time of composite functions.

function	OBPSO	MFO	PSO	OBSCA	SSO	MVO	ABC	HS	SCA
F24	0.7574	0.5157	0.5950	0.4016	0.7597	0.5814	1.2302	0.5575	0.6224
F25	0.6646	0.6135	0.6637	1.0371	0.9749	0.8569	1.2980	0.7218	0.7906
F26	73.0920	70.0629	71.5161	68.9574	85.2587	70.5303	141.9261	74.5750	70.1608
F27	0.2637	0.2478	0.2725	0.1970	0.3961	0.2537	1.8434	0.7392	0.2528
F28	82.1025	78.8720	81.2789	70.5688	91.7360	78.0930	158.5330	81.0636	80.3419
F29	244.3086	249.2951	254.1280	150.7757	280.8456	243.4572	485.5812	246.4182	254.0075

gives the proposed method ability to switch between two components, the sine and cosine functions. Also, the OBL strategy gives the OBSCA ability to skip the local point since it takes the opposite direction into consideration. Moreover, the other algorithms such as PSO, MFO, ABC, HS, used only one equation to update the solutions and this increase the probability to stagnation in local point. While, the other algorithms such as MVO, SSO and SCA are used two equations, in general, to update their solutions. However, the last algorithms may be easy to get stuck in local point, since they depend on the current best solution to update the other solutions, this limitation can be avoided by using the OBL strategy. The influence of this strategy appears in the convergence of the OBPSO and OBSCA algorithm, however, the performance of the OBSCA is better than OBPSO since it combined the both strategies (two equations and OBL).

4.7. Complexity

The time complexity of OBSCA is depended on the size of population N, the number of iterations and the sorting algorithm. In this study, the Quicksort (QS) algorithm has been used in which has complexity (O_{QS}) equal to $O(N\log N)$ in the best case; while in the worst case $O(N^2)$. Therefore, the complexity of the proposed method is:

$$O(OBSCA) = O(t_{max}(2N \times n + O_{QS})) \tag{12}$$

Therefore,

$$O(OBSCA) = \begin{cases} O(t_{max}(k(N+\frac{N}{k}) \times n + N^2)) & In\ best\ case\ of\ Quicksort \\ O(t_{max}(k(N+\frac{N}{k}) \times n + N\log N)) & In\ best\ case\ of\ Quicksort \end{cases} \tag{13}$$

where n is the dimension and the $t_{\max}$ is the maximum number of iterations and $k \in [0, 1]$ represents the part of the population to compute its opposite direction.

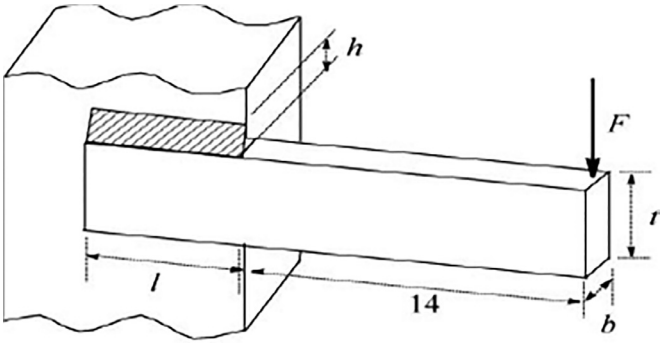

Fig. 8. The parameters of welded beam design problem.

4.8. OBSCA for classical engineering problems

This subsection is analyzed the performance and efficiency of OBSCA by solving three constrained real engineering problems, namely, a welded beam, tension/compression spring, and a pressure vessel. These problems have many inequality constraints, so the OBSCA tries to handle them in processing. We applied the simplest penalty function called death penalty, whereas, swarms are obtained big values if they violate any of the constraints. The number of solutions in all experiments is 30 and the maximum number of iterations is 500.

4.8.1. Welded beam design

The aim of welded beam design problem is to minimize the fabrication cost by determining the optimal value of four variables namely, length of attached part of bar (l), thickness of weld (h), the height of the bar (t) and thickness of the bar (b) (as in Fig. 8). These variables must be satisfied seven constraints. The mathematical definition of this problem as follows:

$$Consider \quad \vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b],$$
$$Minimize \quad f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),$$

Table 12
The optimization results of OBSCA and other algorithms for solving the welded beam design problem.

Algorithm	Optimal values for variables				Optimal cost
	h	*l*	*t*	*b*	
OBSCA	0.230824	3.069152	8.988479	0.208795	1.722315
WOA (Mirjalili & Lewis, 2016b)	0.205396	3.484293	9.037426	0.206276	1.730499
RO	0.203687	3.528467	9.004233	0.207241	1.735344
MVO (Mirjalili et al., 2016)	0.205463	3.473193	9.044502	0.205695	1.72645
CPSO (He & Wang, 2007a)	0.202369	3.544214	9.04821	0.205723	1.72802
GSA (Mirjalili et al., 2016)	0.182129	3.856979	10	0.202376	1.87995
GA (Deb, 1991)	0.2489	6.173	8.1789	0.2533	2.43
HS (Lee & Geem, 2005)	0.2442	6.2231	8.2915	0.24	2.3807
SIMPLEX (Ragsdell & Phillips, 1976)	0.2792	5.6256	7.7512	0.2796	2.5307
DAVID (Ragsdell & Phillips, 1976)	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX (Ragsdell & Phillips, 1976)	0.2444	6.2189	8.2915	0.2444	2.3815
CSCA (Huang et al., 2007)	0.203137	3.542998	9.033498	0.206179	1.733461
CBO	0.205722	3.47041	9.037276	0.205735	1.724663

$$
\begin{aligned}
&\textit{Subject to} \quad g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leqslant 0,\\
&g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leqslant 0,\\
&g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leqslant 0,\\
&g_4(\vec{x}) = x_1 - x_4 \leqslant 0,\\
&g_5(\vec{x}) = P - P_c(\vec{x}) \leqslant 0,\\
&g_6(\vec{x}) = 0.125 - x_1 \leqslant 0,\\
&g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leqslant 0\\
&\textit{Variables range} \quad 0.1 \leqslant x_1, x_4 \leqslant 2,\\
&0.1 \leqslant x_2, x_3 \leqslant 10,
\end{aligned}
$$

$$
\begin{aligned}
&\textit{where} \quad \tau(\vec{x})\\
&= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tau' = \frac{p}{\sqrt{2x_1x_2}}, \quad \tau'' = \frac{MR}{J},\\
&M = P\left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},\\
&P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
&J = 2\left\{\sqrt{2x_1x_2}\left[\ \frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2\ \right]\right\}, \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \quad \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}\\
&P = 6000\ lb, L = 14\ in., \quad \delta_{max} = 0.25\ in.,\\
&E = 30 \times 1^6\ psi, \quad G = 12 \times 10^6\ psi,\\
&\tau_{max} = 13600\ psi, \quad \sigma_{max} = 30000\ psi
\end{aligned} \tag{14}
$$

The OBSCA was applied for solving this problem and compared it with twelve optimization algorithms which are reported in previews works; these are Davidon-Fletcher-Powell (DAVID) (Ragsdell & Phillips, 1976), Co-evolutionary Differential Evolution (CSCA) (Huang, Wang, & He, 2007), Coevolutionary Particle Swarm Optimization (CPSO) (Krohling & dos Santos Coelho, 2006), Co-evolutionary Differential Evolution (CSCA) (Huang et al., 2007),Harmony Search (HS) (Lee & Geem, 2005), Genatic Algorithm (GA) (Deb, 1991), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016b), Griffith and Stewart's successive linear approximation (APPROX) (Ragsdell & Phillips, 1976), MVO (Mirjalili et al., 2016), Ray Optimization (RO) (Kaveh & Khayatazad, 2012) and Simplex method (SIMPLEX) (Ragsdell & Phillips, 1976), as shown in Table 12. From Table 12 we can conclude that the results of OBSCA is better than all other algorithms.

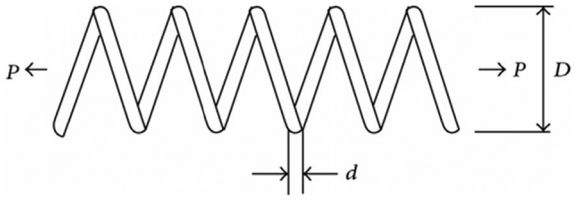

Fig. 9. The parameters of tension/compression spring problem.

4.8.2. Tension/compression spring design

The aim of this section is to analyze the performance of OBSCA to solve the tension/compression spring problem which aims to minimize the weight of tension/compression spring. In this problem, there are three parameters namely, mean coil diameter (D), the number of active coils (N) and wire diameter (d). This problem is illustrated in Fig. 9 and it is defined as follows:

$$
\begin{aligned}
&\textit{Consider} \quad \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N],\\
&\textit{Minimize} \quad f(\vec{x}) = (x_3 + 2)x_2x_1^2,\\
&\textit{Subject to} \quad g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leqslant 0,\\
&\qquad g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leqslant 0,\\
&\qquad g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0,\\
&\qquad g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0,\\
&\textit{Variables range} \quad 0.05 \leqslant x_1 \leqslant 2, 0.25 \leqslant x_2 \leqslant 1.30, 2.00 \leqslant x_3 \leqslant 15
\end{aligned} \tag{15}
$$

The OBSCA is compared eleven optimization algorithms which are reported in related works such as MVO, WOA (Mirjalili & Lewis, 2016b), CSCA (Huang et al., 2007), CPSO (He & Wang, 2007a), Evolution Strategy (ES) (Mezura-Montes & Coello, 2008), GA (Coello, 2000), Simple Evolution Strategy (SES) (Mezura-Montes, Coello, & Landa-Becerra, 2003), Ray-Saini method (RAY & SAINI, 2001), and mathematical method by Belegundu and Arora (Belegundu & Arora, 1985) as shown in Table 13. The OBSCA outperforms only the original MVO, SES, Ray-Saini method and Belegundu-Arora method.

4.8.3. Pressure vessel design problem

Fig. 10 illustrate Pressure vessel design problem. The aim of this problem is to minimize the whole cost of the cylindrical pressure vessel. The optimization operation tries to determine the op-

Table 13
The optimization results of OBSCA and other literatures for tension/compression spring design problems.

Algorithm	Optimal values for variables			Optimal cost
	d	*D*	*N*	
OBSCA	0.05230	0.31728	12.54854	0.012625
WOA (Mirjalili & Lewis, 2016b)	0.051207	0.345215	12.004032	0.0126763
MVO	0.05251	0.37602	10.33513	0.012790
GSA	0.050276	0.323680	13.525410	0.0127022
CSCA (Huang et al., 2007)	0.051609	0.354714	11.410831	0.0126702
CPSO (He & Wang, 2007a)	0.051728	0.357644	11.244543	0.0126747
ES (Mezura-Montes & Coello, 2008)	0.051643	0.355360	11.397926	0.012698
GA (Coello, 2000)	0.051480	0.351661	11.632201	0.01270478
Belegundu-Arora method (Belegundu & Arora, 1985)	0.0500	0.3177	14.026	0.012730
SES (Mezura-Montes et al., 2003)	N/A	N/A	N/A	0.012732
Ray-Saini method (RAY & SAINI, 2001)	0.321532	0.050417	13.979915	0.013060
RO (Kaveh & Khayatazad, 2012)	0.051370	0.349096	11.76279	0.0126788

Table 14
The optimization results of OBSCA and other literatures for pressure vessel design problem.

Algorithm	Optimal values for variables				Optimal cost
	T_s	T_h	*R*	*L*	
OBSCA	1.2500	0.0625	59.1593	70.8437	5833.9892
WOA (Mirjalili & Lewis, 2016b)	0.812500	0.437500	42.0982699	176.638998	6059.7410
MVO (Mirjalili et al., 2016)	0.8125	0.4375	42.090738	176.73869	6060.8066
PSO-SCA (Liu et al., 2010)	0.8125	0.4375	42.098446	176.6366	6059.71433
HPSO (He & Wang, 2007b)	0.8125	0.4375	42.0984	176.6366	6059.7143
ACO (Kaveh & Talatahari, 2010)	0.812500	0.437500	42.098353	176.637751	6059.7258
CSCA (Huang et al., 2007)	0.8125	0.4375	42.098411	176.63769	6059.7340
ES (Mezura-Montes & Coello, 2008)	0.8125	0.4375	42.098087	176.640518	6059.74560
GA (Coello & Montes, 2002)	0.81250	0.43750	42.097398	176.65405	6059.94634
CPSO (He & Wang, 2007a)	0.8125	0.4375	42.091266	176.7465	6061.0777
GA (Deb, 1997)	0.9375	0.5000	48.3290	112.6790	6410.3811
Lagrangian Multiplier (Kannan & Kramer, 1994)	1.125	0.625	58.291	43.690	7198.200
Branch-bound (Sandgren, 1990)	1.125	0.625	48.97	106.72	7982.5
GSA (Rashedi et al., 2009)	1.125	0.625	55.9886598	84.4542025	8538.8359

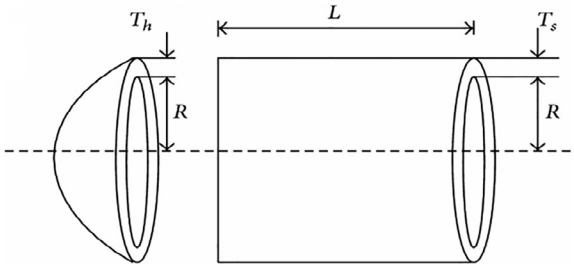

Fig. 10. The parameters of pressure vessel design problem.

timal value of four variables (namely, the thickness of the head (T_h), thickness of the shell (T_s), the length of the cylindrical section without considering the head (L) and the inner radius (R)) that satisfied four constraints as (16):

$$\begin{aligned}
&\textit{Consider} \quad \vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L],\\
&\textit{Minimize} \quad 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3,\\
&\textit{Subject to} \quad g_1(\vec{x}) = -x_1 + 0.0193x_3 \leqslant 0,\\
&g_2(\vec{x}) = -x_3 + 0.00954x_3 \leqslant 0,\\
&g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0,\\
&g_4(\vec{x}) = x_4 - 240 \leqslant 0,\\
&\textit{Variables range} \quad 0 \leqslant x_1, x_2 \leqslant 99, \quad 10 \leqslant x_3, x_4 \leqslant 200, \qquad (16)
\end{aligned}$$

The results obtained by the proposed approach over this problem are evaluated by comparing it with thirteen other optimization algorithms which are reported in previews works, namely, MVO (Mirjalili et al., 2016), PSO-SCA (Liu, Cai, & Wang, 2010), HPSO (He & Wang, 2007b), CPSO (He & Wang, 2007a), ACO (Kaveh & Talatahari, 2010), CSCA (Huang et al., 2007), ES (Mezura-Montes & Coello, 2008), GA (Coello & Montes, 2002) (Deb, 1997), Lagrangian Multiplier (Kannan & Kramer, 1994), Branch-bound (Sandgren, 1990), and GSA (Mirjalili et al., 2016), as shown in Table 14. From Table 14 we can conclude that the results of OBSCA is better than all other algorithms.

5. Conclusions and future works

The Opposition-Based Learning (OBL) has more attention in recent years and it is used to improve the performance of metaheuristic algorithms. Which has an important characteristic that used to search in opposite direction to the current solution and these made the metaheuristic algorithms to search in the whole search space. In this paper, we illustrated the influence of OBL to improve the accuracy of SCA algorithm for solving global optimization and engineering problems. The results of the proposed algorithm (which called OBSCA) have been compared with SCA and other metaheuristic algorithms. The comparisons have shown that the OBSCA is better than traditional SCA to find a globally optimal solution and provides a high performance in term of convergence. Also, from these comparisons, we can conclude that the OBSCA is better than other metaheuristic algorithms in both convergence and time complexity. However, MFO algorithm is better in some functions only from OBSCA in time complexity but MFO not convergence to global solution in these functions, therefore, OBSCA is better than MFO.

Moreover, the OBSCA has been applied for solving real-world engineering problems, that are commonly used to test this kind of techniques. The results of OBSCA in solving engineering problems

are compared with several state-of-the-art approaches and proved encouraging results.

In the future work, we attempt to apply the proposed method in many applications such as data mining and image processing through considering it as 1) Feature selection by converting it to binary version, 2) Image segmentation method through using the Kapur function or Otsu function. 3) Also, the proposed method can be extended to work as the multi-objective optimization algorithm. 4) The proposed method may be hybrid with other metaheuristic algorithms to increase the efficiency of it. 5) Modified the OBSCA for constrained optimization problems. 6) Apply the OBSCA in renewable energy problems.

Acknowledgment

This work was in part supported by national key Research & Development Program of China (NO. 2016YFD0101903), Nature Science Foundation of Hubei Province (Grant No. 2015CFA059), Science & Technology Pillar Program of Hubei Province (Grant No. 2014BAA146), Science & Technology Cooperation Program of Henan Province (NO. 152106000048) and Hubei Collaborative Innovation Center of Basic Education Information technology Services.

References

Abd ElAziz, M., & Hassanien, A. E. (2017). An improved social spider optimization algorithm based on rough sets for solving minimum number attribute reduction problem. *Neural Computing & Applications*.

Abd ElAziz, M., Selim, I. M., & Xiong, S. (2017). Automatic detection of galaxy type from datasets of galaxies image based on image retrieval approach.. *Scientific Reports, 7*, 4463.

Ahandani, M. A., & Alavi-Rad, H. (2015). Opposition-based learning in shuffled frog leaping: An application for parameter identification. *Information Sciences (Ny), 291*, 19–42.

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & Structures*, 1–12.

Bayraktar, Z., Komurcu, M., Bossard, J. A., & Werner, D. H. (2013). The wind driven optimization technique and its application in electromagnetics. *IEEE transactions on antennas and propagation, 61*, 2745–2757.

Belegundu, A. D., & Arora, J. S. (1985). A study of mathematical programming methods for structural optimization. part ii: Numerical results. *International journal for numerical methods in engineering, 21*, 1601–1623.

Bulbul, S. M. A., Pradhan, M., Roy, P. K., & Pal, T. (2015). Opposition-based krill herd algorithm applied to economic load dispatch problem. *Ain Shams Engineering Journal*.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*, 113–127.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*, 193–203.

Cuevas, E., Cienfuegos, M., ZaldıVar, D., & PéRez-Cisneros, M. (2013). A swarm optimization algorithm inspired in the behavior of the social-spider. *Expert Systems with Applications: An International Journal, 40*, 6374–6384.

Cuevas, E., Dıaz Cortés, M. A., & Oliva Navarro, D. A. (2016). Social-spider algorithm for constrained optimization. *Stud Comput Intell, Advances of Evolutionary Computation: Methods and Operators*, 175–202.

Cuevas, E., Oliva, D., Zaldivar, D., Cisneros, M. A. P., & Pajares, G. (2012). Opposition-based electromagnetism-like for global optimization. *Int. J. Innov. Comput. Inf. Control, 8*, 8181–8198.

Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal, 29*, 2013–2015.

Deb, K. (1997). Geneas: A robust optimal design technique for mechanical component design. In *Evolutionary Algorithms in Engineering Applications* (pp. 497–514). Springer.

He, Q., & Wang, L. (2007a). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*, 89–99.

He, Q., & Wang, L. (2007b). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation, 186*, 1407–1422.

Huang, F.-z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and computation, 186*, 340–356.

Kannan, B., & Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of mechanical design, 116*, 405–411.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical report-tr06, Erciyes university, engineering faculty, computer engineering department*.

Kaur S, P. S. (2016). Int j sci eng technol res. *A Novel Sine Cosine Algorithm for the solution of Unit Unit PROBLEM FORMULATION COMMITMENT, 5*, 3298–3310.

Kaveh, A., & Khayatazad, M. (2012). A new meta-heuristic method: Ray optimization. *Computers & Structures, 112–113*, 283–294.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations, 27*, 155–182.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *Neural Networks 1995. Proceedings., IEEE Int. Conf., 4*, 1942–1948.

Krohling, R. A., & dos Santos Coelho, L. (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*, 1407–1416.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering, 194*, 3902–3933.

Liang, J., Suganthan, P., & Deb, K. (2005). Novel composition test functions for numerical global optimization. In *Proceedings 2005 IEEE swarm Intelligence Symposium, 2005. SIS 2005* (pp. 68–75).

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*, 629–640.

Ma, X., Liu, F., Qi, Y., Gong, M., Yin, M., Li, L., … Wu, J. (2014). Moea/d with opposition-based learning for multiobjective optimization problem. *Neurocomputing, 146*, 48–64.

Mezura-Montes, E., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37*, 443–473.

Mezura-Montes, E., Coello, C. C., & Landa-Becerra, R. (2003). Engineering optimization using simple evolutionary algorithm. In *IEEE. Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on* (pp. 149–156).

Mirjalili, S. (2015a). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.

Mirjalili, S. (2015b). Sca: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Syst.*, 1–14.

Mirjalili, S., & Lewis, A. (2016a). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., & Lewis, A. (2016b). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495–513.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Kiran, MustafaServet (2015). TSA: Tree-seed algorithm for continuous optimization. *Expert Systems with Applications, 42*, 6686–6698.

Mudhsh, M., Xiong, S., Abd ElAziz, M., Hassanien, A. E., & Duan, P. (2017). Hybrid swarm optimization for document image binarization based on otsu function. *CASA2017*.

Ragsdell, K., & Phillips, D. (1976). Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry, 98*, 1021–1025.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information sciences, 179*, 2232–2248.

RAY, T., & SAINI, P. (2001). Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Engineering Optimization, 33*, 735–748.

Salimi, H. (2015). Stochastic fractal search: A powerful metaheuristic algorithm. *Knowledge-Based Syst, 75*, 1–18.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design, 112*, 223–229.

Sindhu, R., Ngadiran, R., Yacob, Y. M., et al. (2017). Sine-cosine algorithm for feature selection with elitism strategy and new updating mechanism. *Neural Comput Applications*.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (May 2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *Nanyang Tech. Univ., Singapore and KanGAL, Kanpur Genetic Algorithms Lab., IIT, Kanpur, India, Tech. Rep., Rep. No., 2005*.

Tawhid, M. A., & Savsani, V. (2017). Multi-objective sine-cosine algorithm (mo-sca) for multi-objective engineering design problems.. *Neural Comput Appl*, 1–15.

Tizhoosh, H. (2005). 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, 1*, 695–701.

Verma, O. P., Aggarwal, D., & Patodi, T. (2016). Opposition and dimensional based modified firefly algorithm. *Expert Systems with Applications, 44*, 168–176.

Wang, H., Li, H., Liu, Y., Li, C., & Zeng, S. (2007). Opposition-based particle swarm algorithm with cauchy mutation. In *IEEE Congress on Evolutionary Computation, 25-28 September, Singapore* (pp. 4750–4756).

Wolpert, D. H., & Macreadym, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*, 67–82.

Yang, X. S. (2012). Flower pollination algorithm for global optimization.. *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics), 7445 LNCS*, 240–249.