Soft Computing (2020) 24:14051–14072
https://doi.org/10.1007/s00500-020-04781-3

METHODOLOGIES AND APPLICATION

An improved brainstorm optimization using chaotic opposite-based learning with disruption operator for global optimization and feature selection

Diego Oliva[1] · **Mohamed Abd Elaziz**[2]

Published online: 26 February 2020

Abstract
Optimization has increased its use in different domains for accurately solving challenging problems. Complex optimization problems require the use of methods that possess the capabilities to properly explore the search spaces. The traditional algorithms commonly tend to fail in suboptimal values during the optimization process; this fact affects the quality of the solutions. This situation occurs for different reasons, but the lack of diversity due to the use of exploitation operators is the most common. Brainstorm optimization is an alternative method based on the social strategy to generate new innovative ideas in work groups. In brainstorm optimization, each solution representing an idea and brainstorm process is performed using clustering algorithms. However, brainstorm optimization is not able to thoroughly explore the search space, and its diversity is reduced. It does not possess any mechanism to escape from suboptimal solutions. Besides, the computational effort is also increased in the iterative process. This paper presents a modified version of brainstorm optimization that improves its performance. In the proposed algorithm, chaotic maps and opposition-based learning are applied to initialize the solutions for a given problem. Moreover, in the optimization process, the positions of the initial population are updated using the disruptor operator. After updating the population, opposition-based learning is used again to analyze the opposite solutions. The combination of chaotic maps, opposition-based learning and disruption operator improve the exploration ability of brainstorm optimization by increasing the diversity of the population. The proposed method has been evaluated using a set of benchmark functions, and it has been also used for feature selection in data mining. The results show the high efficacy of the proposed method to determine the optimal solutions of the tested functions.

Keywords Brainstorm optimization (BSO) · Opposition-based learning (OBL) · Swarm algorithms (SA) · Evolutionary algorithms (EA) · Feature selection

1 Introduction

Optimization is introduced in different fields like engineering or computer sciences; this process consists in determining the best solution for a complex problem exploring a search space (Abualigah 2019; Abualigah and Hanandeh 2015; Deng et al. 2017a). Accuracy is always expected from optimization methods; however, depending on the implementation, they require more (or less) accurate solutions (Abualigah et al. 2018a, b; Zhao et al. 2019; Deng et al. 2019; Abualigah and Khader 2017; Deng et al. 2017b; Abualigah et al. 2018c). Optimization can be divided into multi-objective optimization (MO) and global optimization (GO). In MO, it is necessary to find a set of solutions for a group of objective functions; meanwhile, in GO only a single-objective function is used, and one solution is found. This paper focuses on GO, where a considerable amount of methods has been proposed inspired by different natural processes. For example, the genetic algorithms (GAs) (Goldberg 1989) that are a part of the evolutionary algorithms (EAs) or the particle

Communicated by V. Loia.

✉ Diego Oliva
diego.oliva@cucei.udg.mx

Mohamed Abd Elaziz
abd_el_aziz_m@yahoo.com

1 Departamento de Ciencias Computacionales, Universidad de Guadalajara, CUCEI Av. Revolucion 1500, Guadalajara, Jal, Mexico

2 Department of Mathematics, Faculty of Science, Zagazig University, Zagazig, Egypt

swarm optimization (PSO) (Kennedy and Eberhart 1995) that defines the swarm algorithms (SAs). EAs and SAs aim to generate intelligent approaches based on behaviors from nature, in general, a set single agents is used, and these agents can communicate to each other sending information about the region of the search where they are located. This process can be divided into exploration and exploitation, and a good balance between them is reflected in the efficiency of the optimization methods. Most of the EAs and SAs are a good alternative for GO. However, only a small group of algorithms explore the intelligence of humans. Some examples are the human behavior-based optimization (HBBO) (Ahmadi 2017) that mimics the interaction of societies or the brainstorm optimization (BSO) (Shi 2011) that is based on the process of brainstorming to generate new ideas. Both HBBO and BSO are part of the swarm algorithms.

On the other hand, human beings consider the most intelligent species on the planet, as humans, we can work in groups in solving any problem. To provide solutions, we can learn, experiment, explore, think and collaborate with other humans. Also, we have the creativity and the capability to innovate. In this context, the brainstorm is a strategy for generating innovative solutions for a specific problem, in which a group of human experts from different areas interacts to find new ideas; a moderator helps to select the best ideas according to the criteria that provide a good solution. The selected ideas are now taken as a base to generate new ones. The BSO is a relative new swarm optimization algorithm that possesses different operators for imitating the brainstorming process (Shi 2011). The most important steps are the convergence and the divergence operators, they are used to find the best solution, and they also perform the exploitation and the exploration of the search space. In the convergence, the elements of the population are concentrated in a specific section of the search space; meanwhile, in the divergence, the population is distributed to explore different regions. However, the main advantage of BSO is the use of clustering in the iterative process. In this way, BSO combines the data mining techniques and swarm intelligence to create an optimization algorithm. The clustering step is used to separate the population into groups that are used to exploit prominent regions in the domain of the problem. The BSO has been used in economic dispatch (Jadhav et al. 2012), and in optimal power flow solution (Krishnanand et al. 2013). Meanwhile, another interesting application for neural networks has also been introduced in Cao et al. (2015).

BSO as other swarm or evolutionary approaches does not provide the best solutions for all the optimization problems, and the computational effort affects its performance; in this context BSO is also open to being adapted and modified (Zhan et al. 2013). Some modifications have been done in the clustering step, for example, in (Chen et al. 2015), authors proposed the affinity propagation clustering. Meanwhile, in Shi (2015), the elitist rules are proposed to separate the BSO population into two groups depending on the fitness, and this modification avoids the use of more complex clustering methods. Another typical situation in BSO is the initialization of new solutions along the iterative process. To affront such situations, there are generated new alternatives to improve the BSO, for example the dynamic adjustment of the internal parameters (Zhou et al. 2012). The initialization of new solutions using a batch-mode has also been presented by Chen et al. (2014). Such approaches increase the performance of BSO providing better results than the standard algorithm. However, based on the No-Free-Lunch (NFL), not all the optimization methods are able to provide excellent results for the same problem (Wolpert and Macready 1997). Moreover, the NFL also assumes that an optimization method cannot be enhanced without sacrifice any advantage.

Considering the above, the drawbacks that BSO possesses can be summarized in the lack of exploration that depends directly on the internal configuration of the algorithm. The configuration of the control parameters of BSO is not an easy task, and it also depends on the problem to be solved. Moreover, the exploitation is also affected by the way in which, the clusters are created. Such problems are the main motivation of this paper to propose an improved version of BSO. In this sense, there are used three methods at different stages of the algorithm, and such methods are (1) the chaotic maps, (2) the opposition-based learning and (3) disruption operator. The chaotic maps (CM) are a part of the chaos theory that is defined as an erratic properties in nonlinear systems. The CM are kind of particles traveling through a nonlinear dynamic system. Such particles do not follow any regular path, for that reason the chaotic distribution has a high level of randomness. In practice, the optimization takes advantage of the features of the CM since different implementations support the evidence that the diversity of solutions and the convergence are improved using them (Yang et al. 2007). On the other hand, the opposition-based learning (OBL) is a machine learning rule proposed by Tizhoosh (2005) for improving the search capabilities of optimization algorithms. The OBL takes the population of solutions and computes an opposite population, and then, using the objective function value the best elements between the two populations are selected. Such elements are used to create a new set of solution that possesses the best position in the search space. OBL can be used in different sections of the algorithm, depending on the implementation, it could be applied in the initialization or after a modification of the set of solutions. The OBL has demonstrated its ability to improve several SA (Cuevas et al. 2012; Abd ElAziz et al. 2017). In the same context, the disruption operator (DO) has been proposed to enhance the exploitation and the exploration ability of SA and EA. The DO is extracted from astrophysics and was introduced by Harwit (2006), where the disruption in astronomy is the sudden inward fall of a group of particles by the gravity. This phenomena occurs when all other forces are not able to provide enough pressure to counter the gravity and maintain the equilibrium (Harwit 2006). The use of DO for optimization algorithms was first proposed for enhancing the performance of the gravitational search algorithm (GSA) (Sarafrazi et al. 2011). Moreover, it attracts the attention of researchers, and its use has been extended for different approaches like the Bare-bones particle swarm optimization (Liu et al. 2014).

The algorithm proposed in this paper is called the opposition chaotic BSO with disruption (OCBSOD). In this method, the initialization is performed using a chaotic map to compute the initial solutions; then, the OBL generates the opposite positions in the search space. The best particles are selected and used in the iterative process. To update the position of the elements of the population, the DO is used with the BSO operators; then, the OBL is applied to improve the exploration ability of the search domain. The aim of this paper can be summarized in the use of different strategies to enhance the performance of BSO. The OCBSOD has been tested over a set of optimization problems with different complexity. The experimental results and the comparison with other recent methods provide the evidence of the effectiveness of the proposed approach to optimize mathematical functions. Moreover, to verify the performance of the proposed hybrid algorithm in real problems, it has been used to solve the problem of feature selection (FS). FS is a method used to extract the most representative features from a large set of data. FS is a critical step; it helps to discard redundant and irrelevant features and reduce the dimensionality of the dataset (Ewees et al. 2019; Labani et al. 2018; Tian et al. 2018a, b). The proposed OCBSOD for FS is applied over different datasets taken from the UCI repository (Frank and Asuncion 2010), and they have different complexities. The results obtained in FS with the OCBSOD are superior to other similar approaches from the state of the art. The main contributions of this study can be summarized as:

- Improve the performance of the brainstorm optimization algorithm through providing it with a suitable initial population and maintain its diversity.
- Apply the chaotic maps and opposite-based learning to find the most suitable initial population for BSO.
- Integrate the BSO with disruptor operator to enhance its diversity during the optimization process.
- Evaluate the performance of the proposed OCBSOD to find the solution of a set of global optimization problems.
- Evaluate the ability of the proposed OCBSOD to enhance the classification of different UCI datasets by using it as a feature selection method.

The remainder paper is organized as follows: Sect. 2 introduces the concepts of chaotic maps, opposition-based learning, disruption operator and brainstorm optimization. Section 3 presents the proposed OCBSOD; meanwhile, in Sect. 4 the experimental results and comparisons are provided. Finally, in Sect. 5, some conclusions are discussed.

2 Background

2.1 Chaotic maps: basic

Recently, the improvement in EA's and SA's by using the chaos theory has more attention, for example multi-verse optimizer (Ewees et al. 2019) and brainstorm optimization (Yang and Shi 2015). According to the results of these approaches, an enhancement in the convergence rate and the diversity of these EA's and SA's can be seen. This results from the properties of chaos such as ergodicity that leads to traverse all the local points and searching for the solutions in the whole search domain (Yang and Shi 2015). Also, if the nonlinear system has an infinite number of different periodic responses and represents sensitive dependence on the initial values, then it is considered as a chaotic system. This sensitivity property to the initial values for a chaotic system makes large differences in its output if there exist small differences in the initial values. The third important property is the stochastic/randomness that makes the chaotic system can avoid the local optimal points. Based on these properties, the optimization algorithms that employ chaotic maps potentially increase the searching capabilities. In other words, using chaotic maps an optimal solution can be found faster that using the standard probability distribution.

The chaotic maps are used to model the chaos. A chaotic map (CM) is defined as a dynamical discrete-time function with a continuous value that determines the relationship of the current value (ch_i) of the chaotic system with its following value ch_{i+1}. The mathematical definition of these maps can be formulated as in the following equation:

$$ch_{i+1} = \beta(ch_i), \tag{1}$$

where β represents the transformation mapping function. The Singer map is considered as one of the most popular CMs that was introduced by Aguirregabiria in 2014 (Aguirregabiria 2009), and it can be defined by the following equation:

$$ch_{i+1} = \mu(7.86ch_i - 23.31ch_i^2 + 28.75ch_i^3 - 13.301875ch_i^4), \tag{2}$$

where $ch_i \in (0, 1)$ is the previous chaotic number, while the $\mu \in [0.9, 1.08]$ is a random number.

2.2 Opposition-based learning: basic concept

In (Tizhoosh 2005), the basic knowledge of opposition-based learning (OBL) is introduced to enhance the performance of the algorithm through considering a solution and its opposite at the same time. The classical EAs (and SAs) start by generating a random population which contains the solutions for the tested problem, and they go through updating it toward the optimal solution. Therefore, the computational time may be increased, and also, the convergence rate may be reduced if this initial population is not properly selected; however, this may occur in the case of the absence of prior information. To avoid this problem, it can be determined a strategy for searching in the opposite direction about new solution. The basic concepts of the OBL strategy can be defined by assuming a real number x lies in the interval $[u, l]$; then, its opposite number $(\overline{x})$ is computed by as:

$$\overline{x} = u + l - x \tag{3}$$

In higher dimension, the generalization of an $\overline{x}$ formulated as:

$$\overline{\mathbf{x}} = \mathbf{u} + \mathbf{l} - \mathbf{x} \text{ or } \overline{x_j} = u_i + l_j - x_j, \quad j = 1, 2, \ldots, D, \tag{4}$$

In Eq. (4), $\mathbf{x}$ and $\overline{\mathbf{x}}$ represent the solution vector and its opposite solution in D dimension, respectively.

2.2.1 Opposition-based optimization

The opposition-base optimization method is introduced by considering the $\mathbf{x} \in R^D$ as the solution of the given problem and its fitness function $f(\mathbf{x})$. As well as, based on the definition of the opposite value, $\overline{\mathbf{x}}$ is the opposite to $\mathbf{x}$ and its fitness function $f(\overline{\mathbf{x}})$. Now, the $\overline{\mathbf{x}}$ is selected when its $f(\overline{\mathbf{x}})$ is better than $f(\mathbf{x})$; otherwise, $\mathbf{x}$ will be selected. Thus, by this way, the best solutions from the current solutions and their opposite are kept in the population (Cuevas et al. 2012).

2.3 Disruption operator

The basic definition of the disruption operator D_{op} is given as Sarafrazi et al. (2011):

$$D_{op} = \begin{cases} Dis_{i,j} \times \Psi\left(\frac{-1}{2}, \frac{1}{2}\right) & \text{if } Dis_{i,\text{best}} \geq 1 \\ 1 + Dis_{i,\text{best}} \times \Psi\left(\frac{-10^{-16}}{2}, \frac{10^{-16}}{2}\right) & \text{otherwise} \end{cases} \tag{5}$$

where $\Psi(a, b)$ represents the uniform function used to generate the number in the interval $[a, b]$. The $Dis_{i,j}$ is the Euclidean distance between the ith and jth idea (where j is the nearest neighborhood of the ith idea). From the definition of D_{op}, $D_{op} < 1$ if the two ideas are similar to each other and this refers to the convergence of the ideas.

2.4 Brainstorm optimization

The brainstorm optimization (BSO) is a relative novel algorithm that is inspired by the process of creating new ideas. Brainstorming is commonly used in companies to increase the creativity in work groups. The ideas are formulated by the members of the groups and selected by a moderator considering predefined criteria. In the BSO context, an idea is considered as a candidate solution and is extracted from the search space. The work groups are simulated using clustering, and it is possible to interchange ideas between the "groups." In general terms, the BSO begins by randomly building a set of candidate solutions. These solutions represent positions in the bounded search domain where the objective function is defined. Once the solutions are initialized and evaluated, m clusters are generated. The entire procedure is explained in Algorithm 1, and the main steps are explained in this section.

2.4.1 Clustering

In this process, the elements of the initial population are separated into groups considering similarities that they possess. Along the iterative process, the clusters are maintained, and new solutions (ideas) are assigned to each group. In this context, the ideas with the best objective function value replace the center of the cluster. Different methods can be used in this phase; however, the k-means algorithm is used in the traditional BSO algorithm.

2.4.2 Generating new positions

This step of the BSO is applied to maintain the diversity of the ideas. A new idea is created using one or more elements of the population (or clusters). In the standard BSO, a probability p_{gen} is used to determine if the new solution will be produced by one or two ideas stored in the population. This step is crucial in the optimization process of BSO because if the exploitation of the prominent regions can be enhanced, a new solution is generated by one cluster. Meanwhile, when it is created using two or more clusters, the exploration is increased because the new idea could be far from the elements used for its computation. In Eq. (6), the process of selecting between using one or two clusters is performed considering the probability p_{ot}.

$$x_s = \begin{cases} x_i & \text{if } \rho_2 < p_{ot} \\ r_1 \times x_{i,1} + (1 - r_2) \times x_{i,2} & \text{for two clusters} \end{cases} \tag{6}$$

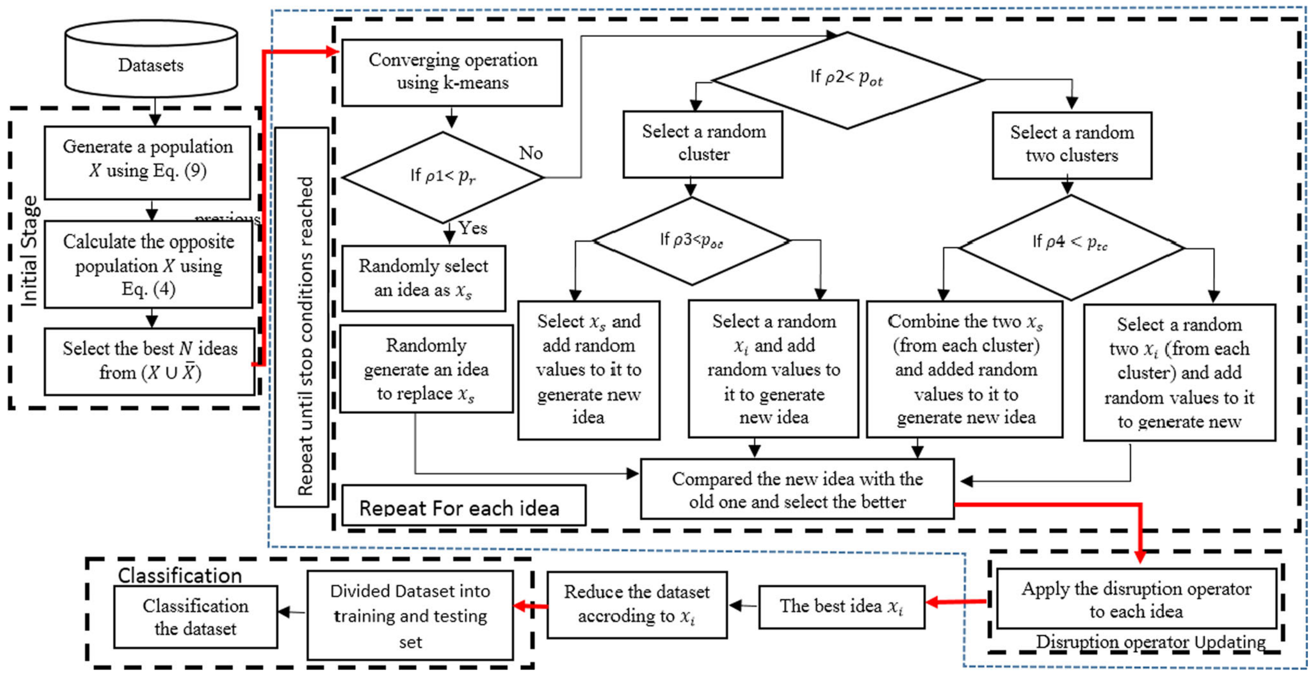

Datasets
Initial Stage
Generate a population X using Eq. (9)
Calculate the opposite population X using Eq. (4)
Select the best N ideas from (X ∪ X̄)
Repeat until stop conditions reached
Converging operation using k-means
If ρ1< p_r
No
Yes
Randomly select an idea as x_s
Randomly generate an idea to replace x_s
If ρ2< p_ot
Select a random cluster
Select a random two clusters
If ρ3<p_oc
If ρ4 < p_tc
Select x_s and add random values to it to generate new idea
Select a random x_i and add random values to it to generate new idea
Combine the two x_s (from each cluster) and added random values to it to generate new idea
Select a random two x_i (from each cluster) and add random values to it to generate new
Compared the new idea with the old one and select the better
Repeat For each idea
Apply the disruption operator to each idea
Disruption operator Updating
The best idea x_i
Reduce the dataset accroding to x_i
Divided Dataset into training and testing set
Classification
Classification the dataset

Fig. 1 Flowchart of the proposed method

From Eq. (6), x_s is the selected idea from the population, x_i is the ith element of the population, and ρ_2, r_1 and r_2 represent a random uniformly numbers. Once x_s is chosen, there is a necessity to compute the new position, and this task is performed using Eq. (7).

$$x_{\text{New}} = x_s + \zeta \times r_3, \tag{7}$$

where x_{New} is the new element that is computed and $r_3 \in [0, 1]$ represents a random value uniformly distributed. Meanwhile, ζ is a variable that controls the speed of convergence, it is updated at each iteration, and its value is computed as follows:

$$\zeta = r_4 \times log\,sig\left(\frac{0.5 \times t_{\max} - t}{k}\right), \tag{8}$$

where $t_{\max}$ represents the maximum number of iterations previously defined, t is the current iteration and the $logsig$ represents a logarithmic sigmoid transfer function. This function is informative to improve the ability of global search and local search at the beginning of the evolution and when the process is approaching the end, respectively. k represents a predefined parameter that used to change the slopes of the $logsig$ function. Finally, $r_4 \in [0, 1]$ represents a random number. The x_{New} is then evaluated in the objective function.

Algorithm 1 Brainstorm Optimization (BSO) algorithm (Shi 2011)

```
1: Initialization: Generate a set (population) of $N$ random $d$-dimensional solution
2: while a stop criterion is not satisfied do
3:     Clustering: Create n clusters considering $m$ elements of the population
4:     Generate new positions: Select a random cluster and compute a new position
5:     Selection: Compare the new solutions with the existing and update considering the position the best element
6:     Evaluation: Evaluate the population in the objective function
7: end while
```

3 The proposed OCBSOD method

The proposed method is called OCBSOD since it combined the chaotic with OBL to improve the initial solutions of BSO algorithm (Shi 2011). Also, the disruption operator is applied to improve the updated solution. The steps of OCBSOD approach are given with more details in the following subsections (see Fig. 1). Then, this section explains such steps for solving the problem of feature selection. Moreover, it is important to mention that some steps (i.e., convert the solution to binary and using the classifier) could be removed for solving mathematical optimization problems.

3.1 Initial stage

The initial solutions are very important for meta-heuristic algorithms because it will affect the convergence rate and the performance of the final solutions. Based on the properties of the chaotic maps (randomness and sensitivity to the initial conditions), they can be used to generate the initial population. As well as, the process of replacing the random initial population with it opposite solution can improve the initial solution and accelerate convergence rate. Therefore, the initial stage of the proposed method combines the chaotic systems and the OBL strategy. The initial stage starts by applying the chaotic *Singer* map to produce a population X of size N as:

$$x_i = ch_i \times (u - l) + u, \quad x_i \in X, \tag{9}$$

where ch_i, u and l represent the chaotic map value generated from Eq. (2), upper bound and lower bound as defined in Eq. (4), respectively. The OBL is then applied to enhance the initial population X by calculating the opposite direction for each idea as defined in Eq. (4). The next step used for the problem of FS in the initial stage is to convert each solution to binary as defined in Eq. (10):

$$x_i(t) = \begin{cases} 1 & \text{if } \frac{1}{1+e^{-x_i(t)}} > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where $\epsilon \in [0, 1]$ is a random threshold and t is the current iteration. Also, the values of x_i which is equal to 1 represent the relevant features and those values that is equal to 0 are the irrelevant features. Then, each $x_i(t)$, at the current iteration t, is evaluated through computing the objective function that defined as (El Aziz and Hassanien 2018).

$$\text{fit}(x_i(t)) = \xi E_{x_i(t)} + (1 - \xi)\left(1 - \frac{|x_i(t)|}{|C|}\right), \tag{11}$$

where $E_{x_i(t)}$ represents the classification error using K-NN classifier; also, the second part of the equation represents the ratio of selected features. $|C|$ represents the total number of features, and $|x_i|$ is the number of selected features. The $\xi \in [0, 1]$ is applied to balance between the two parts of Eq. (11). The solution in the opposite population $\bar{X}$ also assessed by using the same objective function (Eq. (11)).

From X and $\bar{X}$, the best N solutions are chosen which represents the current population. Equations (10)–(11) are used only for FS problem, and they are removed when the proposed method is applied for the global optimization.

3.2 Updating stage

In the updating stage, the ideas are updated by using the traditional BSO algorithm as discussed in Sect. 2.4 and the disruption operator (Sarafrazi et al. 2011).

3.2.1 Updating using traditional BSO

At this step, the convergence operation is performed through clustering the solutions to m clusters using k-means; then, divergence operation is performed through disrupting cluster center and creating solutions, in which the disrupting cluster center chooses a random idea and replaced it with a new random generated idea according to the probability p_r. Meanwhile, in the operation used to create new solutions, and the BSO selects one cluster (if the probability $p_{ot} > \rho_2$) or two clusters (if $p_{ot} < \rho_2$) to generate a new idea. In the case of select one cluster, the cluster center x_s is selected from a random cluster when the probability $p_{oc} > \rho_3$; otherwise, a random idea is selected x_s. In the contrary case, the BSO selects two clusters randomly, and then, based on probability $p_{tc} > \rho_4$ the two clusters' center is combined with x_s or two clusters' center ideas can also be combined with x_s. The next step is to create a new idea x_{new} using x_s, and its fitness value is compared with the fitness value of x_s, and the best of them is preserved.

3.2.2 Updating using disruption operator

After the solutions are updated using the traditional BSO steps, the DO is employed to improve the diversity (as a result the convergence also improved), where the disruption operator for any solution is defined as:

$$X = X \times D_{op}, \tag{12}$$

where D_{op} can be defined as in Eq. (5). Finally, the steps of the previous stages are executed again until the stopping conditions are met.

3.3 Complexity

The computational complexity of the OCBSOD ($O(OCBSOD)$) depends on the following items: (1) the size of the population (N), (2) total number of iterations ($t_{\max}$), (3) the dimension of the given problem and (4) the sorting algorithm. (Here, the Quicksort (QS) is used.) Since the complexity of QS (O_{QS}) is $O(N log N)$ and $O(N^2)$ in the best case and the worst case, respectively, so $O(OCBSOD)$ is:

$$O(OCBSOD) = O((4 \times N \times n + O_{QS}) \times t_{\max}) \tag{13}$$

Therefore,

$$O(OCBSOD) = \begin{cases} O\left(\left(N + \frac{N}{k}\right) + (2N \times D + N^2) \times t_{max}\right) & \text{In worst case of Quicksort} \\ O\left(\left(N + \frac{N}{k}\right) + (2N \times D + N log N) \times t_{max}\right) & \text{In best case of Quicksort} \end{cases} \tag{14}$$

where $k \in [0, 1]$ is the part of X to calculate its $\bar{X}$.

Table 1 Parameters of algorithm and their values

Algorithm	Parameters	Value
BSO	Probability for disrupting elitists	0.5
	Probability for select elitist	0.2
	Probability for select one individual	0.8
	Slope of the s-shape function	25
OCBSOD	The same parameters of BSO and the chaotic Singer map	
HS	HMCR	0.7
	Pitch adjusting rate for each generation	0.3
	Minimum pitch adjusting rate	0.3
	Maximum pitch adjusting rate	0.9
	Minimum bandwidth	0.2
	Maximum bandwidth	0.5
ABC	maximum cycle number	1000
	Modification rate	0.8
MFO	b	1
	l	$[-1, 1]$
SSO	Lower female percent	65
	Upper female percent	90
	Probabilities of attraction or repulsion	0.7
MVO	Maximum of Wormhole Existence Probability	1
	Minimum of Wormhole Existence Probability	0.2
SSA	C1	[0, 1]
	C2	[0, 1]

4 Experiments and discussion

To assess the quality of the OCBSOD method, three experiment series are executed, (1) used it as the global optimization method to find the optimal value of 23 benchmark functions, (2) verify the influence of the chaotic Signer map and (3) apply the proposed algorithm as a FS method. The experiments are implemented over "Windows 7 (64bit)" that runs on "CPU Core2 Duo with 4GB ram", and "Matlab 2014b" is used. All the methods executed 30 independent runs over each tested problem, and for comparisons the maximum number of function evaluation is set to 10^5.

4.1 Experiment series 1: benchmark functions

In this experiment, a set of benchmark functions are used to evaluate the performance of the OCBSOD approach as an alternative global optimization method, in which it is compared with other algorithms that have been established their performance in the related literature; these algorithms are, namely, Moth-flame optimization (MFO) (Mirjalili 2015), BSO, artificial bee colony (ABC) (Karaboga 2005), harmony search (HS) (Lee and Geem 2005), social-spider optimization (SSO) (El Aziz and Hassanien 2018), salp swarm algorithm (SSA) (Mirjalili et al. 2017) and multi-verse optimizer (MVO) (Mirjalili et al. 2016). For free comparison between the algorithms, we fixed the common parameters such as the total number of the solutions $N = 30$ with dimension equal to 30. Also, the maximum number of iterations is 1000. In this study, the parameter values for each algorithm are setting as in Table 1 which set as the same values in the original implementation.

4.1.1 Test functions description

There are 23 benchmark functions that defined in Table 2, also these functions contain seven unimodal functions (*F1* to *F7*) and the rest are multimodal (*F8* to *F*23). The unimodal functions have a single extreme point within a search domain and they are applied to evaluate the convergence rate of the algorithms. In other words, the multimodal functions have more than one local one extreme value points.

4.1.2 Measures of performance

In order to evaluate the performance of the methods, the following measures are used (Suganthan et al. 2005):

Table 2 Tested functions

ID	Equation	Lower	Upper	Dimension	Type
F1	$f(x)=\sum_{i=1}^{n} x_i^2$	– 100	100	10	Unimodal
F2	$f(x)=\sum_{i=1}^{n} \|x_i\| + \Pi_{i=1}^{n}\|x_i\|$	– 10	10	10	Unimodal
F3	$f(x)=\sum_{i=1}^{n}(\sum_{j-1}^{i} x_i)^2$	– 100	100	10	Unimodal
F4	$f(x)=max_i\{\|x_i\|, 1\le i\le n\}$	– 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	$f(x)=\sum_{i=1}^{n} -x_i sin(\sqrt{\|x_i\|})$	– 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	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\Pi_{i=1}^{n}cos(\frac{x_i}{\sqrt{i}})+1$	– 600	600	10	Multimodal
F12	$f(x)=\frac{\pi}{n}\{10sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	– 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

(1) Mean of fitness (μ):

$$\mu=\frac{1}{N_{\text{run}}}\sum_{i=1}^{N_{\text{run}}}\text{fit}_i \tag{15}$$

(2) Standard deviation (STD):

$$\sigma=\sqrt{\frac{1}{N_{\text{run}}-1}\sum_{i=1}^{N_{\text{run}}}(\text{fit}_i-\mu)^2} \tag{16}$$

where N_{run} is the total number of runs, fit_i is a fitness value (i.e., $F1, F2, \ldots, F23$).

(3) The number of times ($NVTR$) the algorithm successfully reaches to the value-to-reach (VTR) for each test function is defined as:

$$SR=\frac{NVTR}{\text{total number of trails}} \tag{17}$$

Table 3 Comparison between the algorithm based on the average of fitness function values

Function	OCBSOD	BSO	SSO	SSA	MVO	MFO	HS	ABC
F1	5.85E−16	2.66E+01	1.35E+01	2.51E+00	3.88E+01	3.20E+04	9.14E+04	7.90E−04
F2	1.70E−07	1.17E−01	1.78E+01	1.18E+01	1.29E+01	6.96E+01	5.13E+15	2.61E−02
F3	4.36E−15	2.12E+04	4.61E+03	4.07E+04	4.68E+04	1.93E+05	5.41E+05	1.77E+05
F4	2.33E−09	6.45E−09	3.65E−02	1.65E−05	4.96E−02	8.24E−01	3.44E−02	6.75E−01
F5	2.91E+01	2.88E+01	3.94E+03	1.51E+03	2.60E+03	4.24E+08	1.33E+06	3.44E+01
F6	7.73E+00	2.03E+00	1.57E+01	2.48E+00	3.96E+01	3.11E+04	1.03E+05	9.69E−04
F7	1.13E−03	2.23E−03	5.58E−01	8.45E−02	1.34E−01	1.67E+00	4.50E+00	1.32E+00
F8	− 3.12E+04	− 3.07E+04	− 2.32E+04	− 2.38E+04	− 2.39E+04	− 2.37E+04	− 1.87E+04	− 3.47E+04
F9	0.00E+00	6.83E+00	4.84E+02	1.62E+02	6.43E+02	7.50E+02	1.03E+03	8.43E+01
F10	9.47E−09	1.35E−01	3.63E+00	6.94E+00	9.85E+00	1.97E+01	1.83E+01	3.59E+00
F11	0.00E+00	2.05E+01	6.01E−01	7.10E−01	1.36E+00	3.41E+02	8.62E+02	4.78E−02
F12	1.97E−03	2.18E+00	7.35E+00	1.74E+01	1.15E+01	1.95E+08	3.94E+08	1.46E−05
F13	8.81E+00	7.57E−02	2.44E+01	1.79E+02	1.30E+02	2.63E+08	8.47E+08	1.05E−03
F14	2.45E+00	2.85E+00	2.98E+00	1.06E+00	1.06E+00	2.35E+00	1.06E+00	9.98E−01
F15	5.63E−04	7.08E−04	6.37E−04	1.54E−03	3.45E−03	1.24E−03	3.23E−04	7.11E−04
F16	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00	− 1.03E+00
F17	3.98E-01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01	3.98E−01
F18	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.06E+00	3.00E+00
F19	− 3.47E−01	− 3.86E+00	− 2.94E−01	− 3.00E−01	− 3.00E−01	− 3.00E−01	− 3.29E+00	− 3.00E−01
F20	− 3.25E+00	− 3.27E+00	− 3.31E+00	− 3.25E+00	− 3.27E+00	− 3.22E+00	− 3.12E+00	− 3.32E+00
F21	− 1.02E+01	− 8.39E+00	− 1.01E+01	− 9.48E+00	− 7.45E+00	− 7.65E+00	− 7.99E+00	− 1.01E+01
F22	− 1.04E+01	− 8.95E+00	− 1.04E+01	− 8.48E+00	− 9.09E+00	− 7.96E+00	− 8.36E+00	− 1.04E+01
F23	− 1.05E+01	− 8.45E+00	− 1.05E+01	− 8.43E+00	− 8.41E+00	− 7.80E+00	− 5.84E+00	− 1.05E+01

(4) The diversity measure of the algorithm div_{Alg} of D-dimensional problem is defined as:

$$div_{\mathrm{Alg}} = \frac{1}{N}\sum_{i=1}^{N}\left(\sqrt{\sum_{j=1}^{D}(x_i^j - \bar{x}^j)^2}\right), \tag{18}$$

where D and x_i^j are the dimension of the problem and the ith solution at the jth dimension, also, $\bar{x}^j$ is the average solution $\bar{x}$ at the jth dimension.

4.1.3 Discussion

The comparison results for all the methods among a set of 23 functions are given in Tables 3, 4, 5 and 6 and Figs. 2, 3, 4 and 5. From Table 3, it can be seen that, in general, the OCBSOD method gives the better results than the other methods over the most functions. In this sense, it achieved the first rank with 11 functions, six unimodal (F1–F5 and F7) and five multimodal (F8–F11 and F21). Meanwhile, the ABC is in the second rank with five functions (F6, F12–F14 and F20), followed by BSO and HS algorithms that obtain better values for only one function F19 and F15, respectively. As well as, for the functions F16-F18 all the algorithms give the optimal value, also, for the two functions F22, F23, the proposed OCBSOD, SSO and ABC algorithms give the better results overall other algorithms.

Figures 2, 3, 4 and 5 depict the convergence curves for the comparative methods along the functions. From these figures, it is possible to observe that the convergence curve for the proposed OCBSOD algorithm is better than other algorithms for functions namely, F1–F5, F7–F11. However, for functions F6, F12, F13 and F14 the ABC is the best algorithm, while for function F15 the HS algorithm is the better, also, for F16, F17 and F18 the SSA is the best with a small difference between it and other algorithms.

In addition, the standard deviation for each algorithm is presented in Table 4. It can notice from it that (1) The proposed algorithm has the smallest variation overall the algorithms along all function, except for the SSO algorithm that gives the best results in 9 functions (F2, F6–F8, F10, F12–F15) and the MFO the give better results in F18. (2) the traditional BSO, OCBSOD, and SSO give the same standard division for the function F17.

Table 5 shows the computational time and the success rate for each algorithm, in which from this table it can be seen that, in terms of time computational, the proposed OCBSOD algorithm takes a small time to achieve the best value for the

Table 4 Standard deviation for all approaches

Function	OCBSOD	BSO	SSO	SSA	MVO	MFO	HS	ABC
F1	1.39E−16	1.13E+02	1.81E−15	1.93E+00	6.88E+00	1.14E+04	5.25E+04	6.87E−04
F2	2.39E−08	1.49E−01	1.45E−10	2.36E+00	3.89E+01	2.24E+01	2.79E+16	1.46E−02
F3	1.56E−15	2.07E+04	2.78E−12	1.75E+04	4.48E+03	5.22E+04	2.88E+05	1.71E+04
F4	3.77E−19	1.15E−09	1.41E−17	3.16E−06	1.73E−02	1.46E+00	1.36E−02	2.19E−01
F5	3.44E−13	2.16E+00	2.78E−12	3.17E+03	3.28E+03	1.88E+08	4.71E+06	2.42E+01
F6	5.77E+00	8.04E+00	1.08E−01	1.40E+00	6.69E+00	1.40E+04	7.96E+04	1.01E−03
F7	1.10E−06	2.24E−03	2.26E−04	2.89E−02	3.43E−02	4.65E−01	6.96E−01	1.27E−01
F8	2.53E+03	1.25E+04	1.11E+01	1.81E+03	1.43E+03	2.25E+03	9.73E+03	3.88E+02
F9	0.00E+00	2.50E+01	1.73E−13	3.81E+01	8.97E+01	6.94E+01	2.52E+02	8.55E+00
F10	1.37E−09	1.32E−01	2.26E−09	1.29E+00	7.75E+00	2.84E−01	1.98E+00	3.94E−01
F11	0.00E+00	5.24E+01	4.52E−02	1.70E−01	7.37E−02	1.32E+02	6.12E+02	5.47E−02
F12	1.03E−04	4.25E+00	5.42E−04	5.18E+00	2.93E+00	1.41E+08	6.35E+08	1.66E−05
F13	1.32E+00	1.25E−01	1.81E+00	2.12E+01	2.44E+01	2.34E+08	8.53E+08	1.64E−03
F14	1.28E−01	1.18E+00	2.52E−01	2.52E−01	2.52E−01	1.77E+00	3.62E−01	1.56E−12
F15	2.30E−06	2.47E−04	1.10E−05	3.56E−03	6.75E−03	1.39E−03	8.54E−06	1.30E−04
F16	2.30E−16	4.79E−16	4.52E−16	8.23E−15	1.12E−07	6.78E−16	1.72E−03	4.49E−13
F17	0.00E+00	0.00E+00	0.00E+00	2.70E−15	3.84E−08	0.00E+00	5.44E−04	7.40E−12
F18	2.18E−15	4.59E−15	2.26E−15	6.81E−14	8.68E−07	1.32E−15	5.42E−02	4.66E−04
F19	1.76E−17	2.27E−15	5.65E−16	2.26E−16	2.26E−16	2.26E−16	9.15E−01	2.26E−16
F20	6.83E−02	5.93E−02	9.28E−02	5.93E−02	6.07E−02	5.84E−02	7.70E−02	8.40E−13
F21	5.27E−05	2.81E+00	1.23E+00	1.75E+00	2.81E+00	3.42E+00	3.38E+00	7.46E−03
F22	1.04E−04	2.98E+00	1.04E+00	3.28E+00	2.45E+00	3.36E+00	3.44E+00	2.56E−03
F23	1.56E−03	3.30E+00	5.42E+00	3.11E+00	2.90E+00	3.69E+00	3.68E+00	4.21E−03

functions except function F1 the MFO is the better. Also, the MVO obtains the shortest time at the functions F19, F21, and F22. In addition, according to the values of success rate SR, in Table 5 it can be seen that the OCBSOD has the higher number of success rate along most functions except the functions F5–F7, F12–F14, F17 and F18. Also, no any approach reached to the value-to-reach (VTR) for those functions (i.e., F5–F7, F12–F14, F17 and F18) except for function F12 the BSO reached seven times.

Moreover, Fig. 6 shows the diversity of the comparative algorithms for four selected functions. From this figure, we can observe that the diversity of the proposed OCBSOD algorithm is better than other methods among the selected functions.

From the previous results, it can be concluded that the OCBSOD provides an effectiveness and efficiency to determine the global solution for the problems with lower diversity, fast convergence rate. This high performance is achieved due to the good initial population that selected by using the chaotic Singer map to build a population; then, the OBL is used to enhance it. As well as, the disruption operator makes the OCBSOD algorithm maintain its diversity that leads to a good performance.

4.1.4 Statistical analysis

The Wilcoxon rank sum test (Wilcoxon 1945) used in this section to add further statistical analysis; here, all the algorithms run until 10^5 function evaluation are reached. This is a nonparametric test which is used to see if there is a significant difference between the median of the control group and other or not. In this study, the null hypothesis is that there is no a significant difference between the control group (OCBSOD) and the other groups (algorithms) at the level of significance equal 5%. The results of Wilcoxon rank sum test are presented in Table 6, in which we can observe that there exists a significant difference between the OCBSOD and the other methods in most of 23 functions. For example, the proposed algorithm is better than HS in all functions, while there is a no significant difference between OCBSOD and both MVO and SSO overall functions except (F19 and F20) and (F15 and F20), respectively. Also, the results of OCBSOD are not a significant difference with the results of BSO, SSA, MFO and ABC in functions (F5, F12, F14 and F16–F18), (F16–F20), (F14 and F16-F20) and (F5, F16, F17 and F19), respectively.

Table 5 Computational time

Function	Measure	OCBSOD	BSO	SSO	SSA	MVO	MFO	HS	ABC
F1	Time	145.23	163.43	580.30	251.21	817.94	143.18	2774.83	227.17
	SR	100	0	0	0	0	0	0	0
F2	Time	110.38	158.53	547.10	313.00	745.82	140.31	3559.96	242.65
	SR	100	0	0	0	0	0	0	0
F3	Time	460.65	501.23	877.63	1007.19	1066.53	810.96	27499.89	1077.61
	SR	100	0	0	0	0	0	0	0
F4	Time	76.61	145.27	241.10	308.33	169.76	121.00	393.12	271.80
	SR	100	100	0	0	0	7	0	0
F5	Time	87.34	154.46	307.31	336.21	310.18	127.18	1276.40	272.91
	SR	100	100	0	0	0	7	0	0
F6	Time	115.07	157.23	3052.36	297.53	868.27	156.49	3842.41	278.64
	SR	0	0	0	0	0	0	0	0
F7	Time	140.53	165.43	590.99	271.85	933.95	206.87	3599.51	321.14
	SR	0	0	0	0	0	0	0	0
F8	Time	129.22	158.68	527.34	266.71	702.60	185.82	17040.46	410.33
	SR	0	0	0	0	0	0	0	0
F9	Time	125.49	176.88	518.79	259.42	36881.04	182.74	3460.97	319.47
	SR	100	100	100	100	100	100	100	100
F10	Time	137.84	187.19	536.40	19250.11	838.63	193.53	3482.39	295.03
	SR	100	0	0	0	0	0	0	0
F11	Time	169.53	215.21	621.81	260.30	841.03	201.10	11980.11	298.32
	SR	100	0	0	0	0	0	0	0
F12	Time	237.37	293.01	649.38	376.84	977.22	239.97	3934.04	17254.32
	SR	100	0	0	0	0	0	0	0
F13	Time	199.64	277.40	592.98	442.27	938.17	215.30	4006.17	8756.12
	SR	0	7	0	0	0	0	0	0
F14	Time	252.91	345.41	416.58	564.70	254.81	294.74	523.47	555.89
	SR	0	0	0	0	0	0	0	0
F15	Time	93.33	182.95	266.39	203.34	26514.85	97.10	241.39	292.17
	SR	100	100	100	97	87	97	100	100
F16	Time	67.06	175.81	274.57	180.28	116.20	78.94	162.95	273.59
	SR	100	100	100	100	100	100	100	100
F17	Time	66.79	198.33	281.25	178.88	102.15	78.50	148.09	268.03
	SR	0	0	0	0	0	0	0	0
F18	Time	76.53	173.57	277.35	176.20	79.48	23779.38	157.25	269.88
	SR	0	0	0	0	0	0	0	0
F19	Time	101.18	162.90	235.00	201.64	94.78	111.74	203.77	300.86
	SR	100	100	0	0	0	0	73	0
F20	Time	105.54	163.87	226.36	204.48	110.20	115.03	323.73	315.34
	SR	100	100	100	100	100	100	100	100
F21	Time	118.59	170.40	280.71	205.52	102.86	135.45	243.02	464.72
	SR	100	70	100	87	50	63	70	100
F22	Time	122.75	172.77	293.42	213.59	108.52	149.30	251.68	441.54
	SR	100	80	100	73	77	63	73	100
F23	Time	142.58	188.31	289.78	227.55	169.35	199.78	309.00	441.26
	SR	100	70	100	67	63	63	37	100

Table 6 Results of Wilcoxon rank sum test for all algorithms along each function

Function	BSO	SSO	SSA	MVO	MFO	HS	ABC
F1	1	1	1	1	1	1	1
F2	1	1	1	1	1	1	1
F3	1	1	1	1	1	1	1
F4	1	1	1	1	1	1	1
F5	0	1	1	1	1	1	0
F6	1	1	1	1	1	1	1
F7	1	1	1	1	1	1	1
F8	1	1	1	1	1	1	1
F9	1	1	1	1	1	1	1
F10	1	1	1	1	1	1	1
F11	1	1	1	1	1	1	1
F12	0	1	1	1	1	1	1
F13	1	1	1	1	1	1	1
F14	0	1	1	1	0	1	1
F15	1	0	1	1	1	1	1
F16	0	1	0	1	0	1	0
F17	0	1	0	1	0	1	0
F18	0	1	0	1	0	1	1
F19	1	1	0	0	0	1	0
F20	1	0	0	0	0	1	1
F21	1	1	1	1	1	1	1
F22	1	1	1	1	1	1	1
F23	1	1	1	1	1	1	1

4.2 Experiment series 2: influence of chaotic map

In this section, the influence of removing the chaotic *Singer* map from the proposed algorithm and generating a random population is tested using the same benchmark functions. The comparison results between the proposed OCBSOD algorithm and its version (that called OBSOD) without chaotic map are illustrated in Table 7, in which we can observe that the OCBSOD has achieved better results, according to the average of the fitness function, in nine functions (F4, F5, F8-F11, F14, F15 and F21). Also, its version OBSOD has the best value in functions (F1-F3, F6, F7, F12, F13, F19, and F20). Meanwhile, the other five function both have nearly the same average value; however, the standard deviation of the OCBSOD is better than the OBSOD version overall the tested functions except F3, F4, F6, F13, F20, F22, F23. Moreover, the time requires by the proposed OCBSOD is smaller than the OBSOD nearly by 11.7457s along all the tested functions, as well as the SR of OCBSOD algorithm is the best.

4.3 Experiment series 3: feature selection

In this Experimental, the quality of the OCBSOD method to determine the optimal subset of features is assessed by comparing it with other feature selection methods.

4.3.1 Datasets description

In this experiment, eight UCI datasets are used in the comparison between the OCBSOD algorithm and the other algorithms. The description of datasets is given in Table 8, in which these datasets have different properties such as the size of features and instances are different from one dataset to another.

For a fair comparison between the algorithms, the tenfold cross-validation (CV) method is used to split the dataset into testing and training sets. This method is performed through split the dataset into ten groups and make one of them as a testing set and the remaining as training set and repeat this assignment ten times in which at each time, different groups are chosen as a testing set. The output is computed as the average of the ten accuracies.

4.3.2 Performance metric

Several performance metric are used to assess the performance of the OCBSOD method through evaluating the fitness function values, the size of the selected features and the accuracy of classifier according to the selected features as in Table 9, where $|x_{\text{best}}^i|$ represents the length of the selected feature at the ith run, while D is the total number of features. The TN, TP, FN and FP are the true negative samples, the true positive samples, the false negative samples and the false positive samples. Here, the K-nearest neighborhood (K-NN) is used as a classifier.

4.3.3 Discussion

The comparison results between the methods for each dataset are presented in Tables 10, 11 and 12 and Figs. 7, 8, 9 and 11 which represent the performance of each algorithm according to different measures. For example, Table 10 illustrates the best, average and worst values of the fitness function for each algorithm along the datasets. It can be seen in this table that, in general, the proposed OCBSOD has the best average and worst values; however, according to the best measure it has a value similar to SSO algorithm, but better than the other algorithms (also, as in Fig. 7). The performance of the OCBSOD is better than BSO algorithm in all cases except the average for the Soybean dataset. The better values in these measures for fitness function are given in this table in bold font.

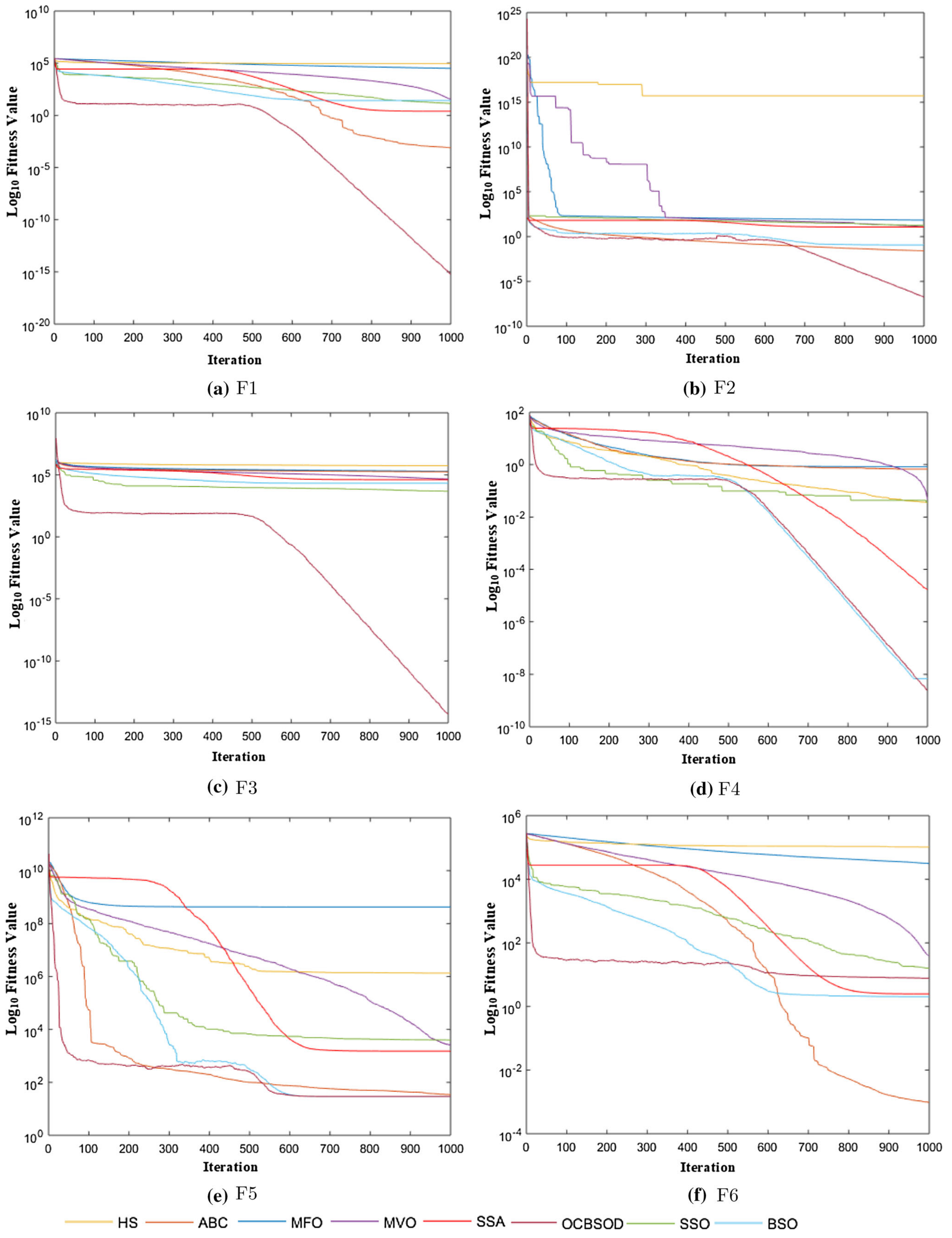


Fig. 2 Convergence curves of F1–F6

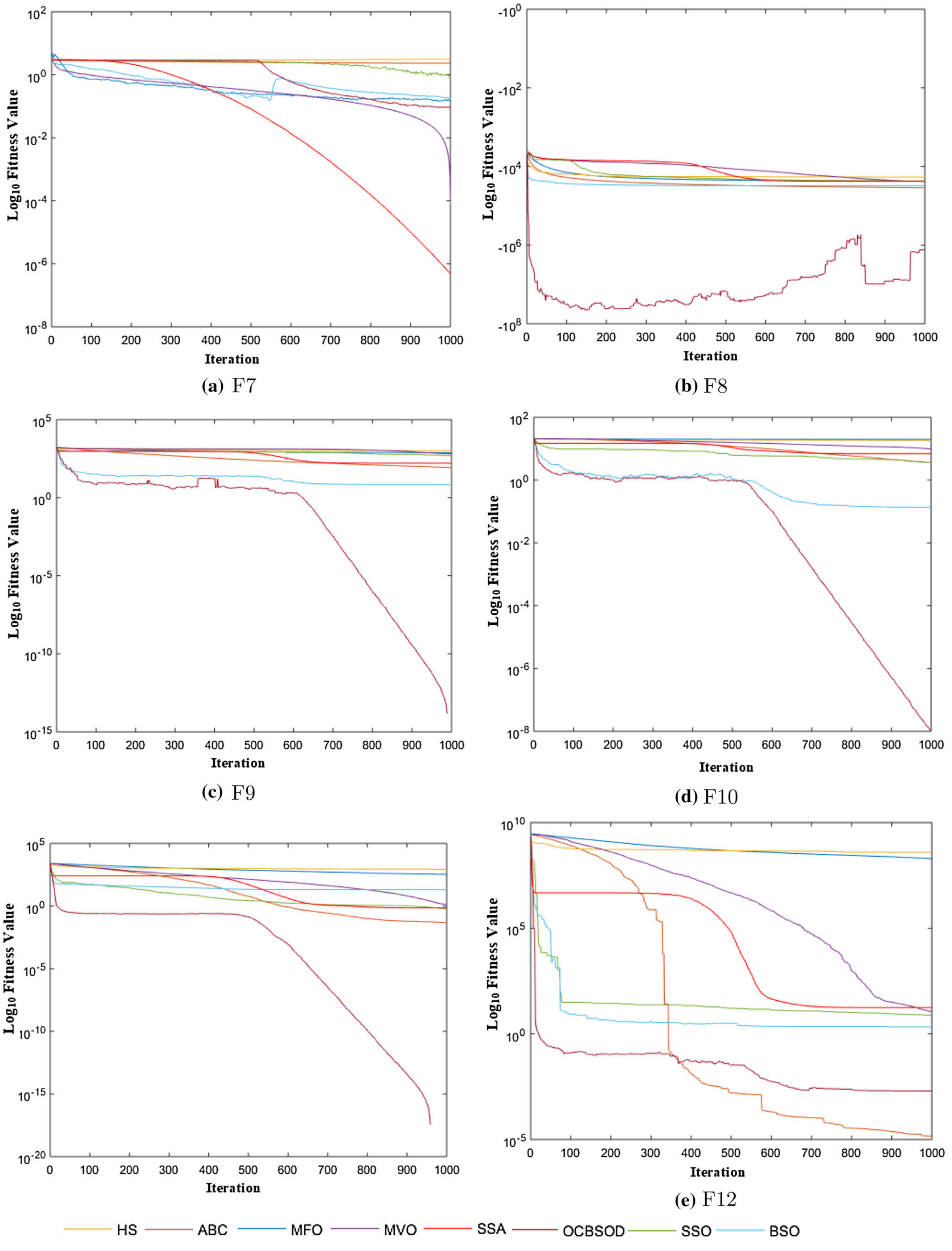


Fig. 3 Convergence curves of F7–F12

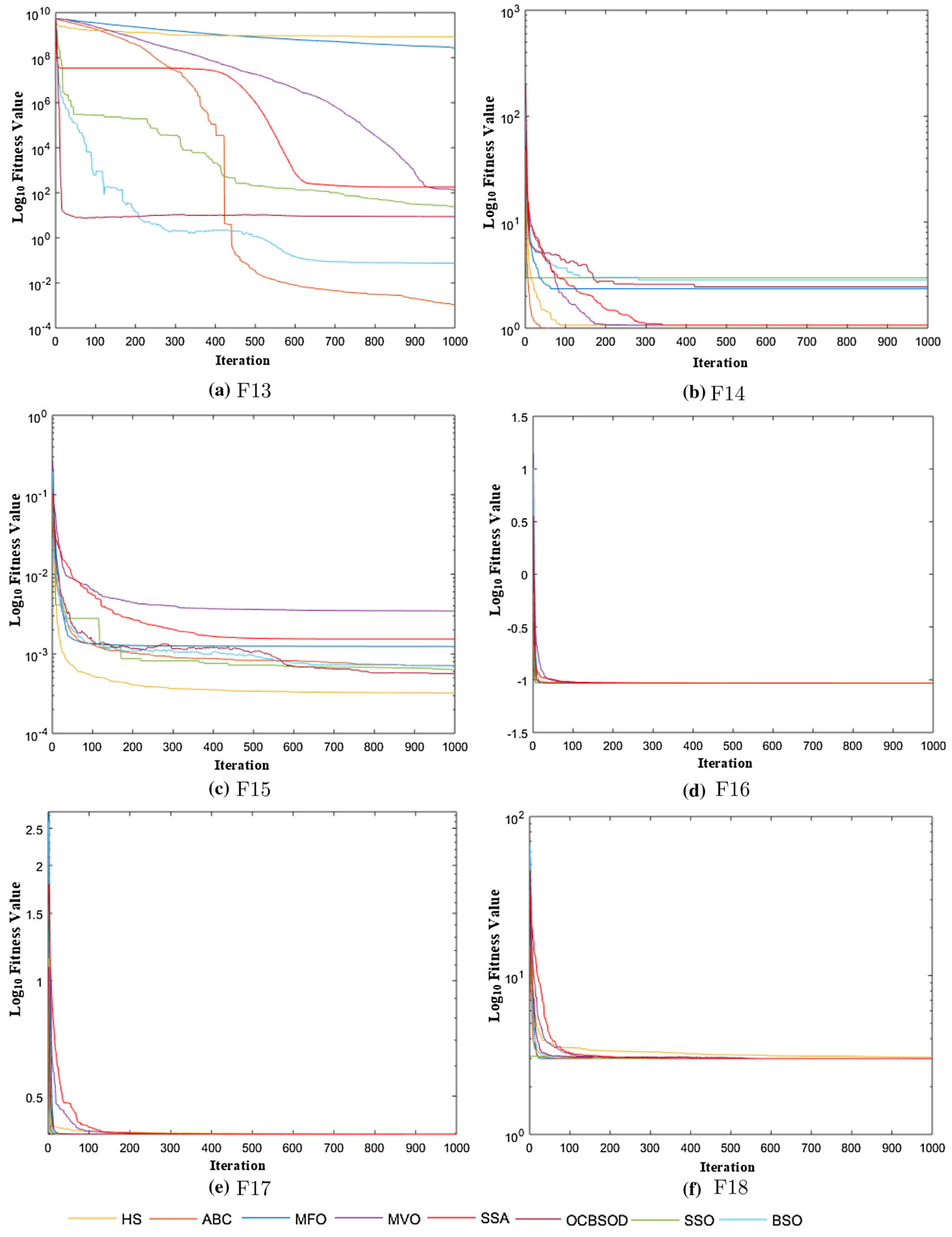
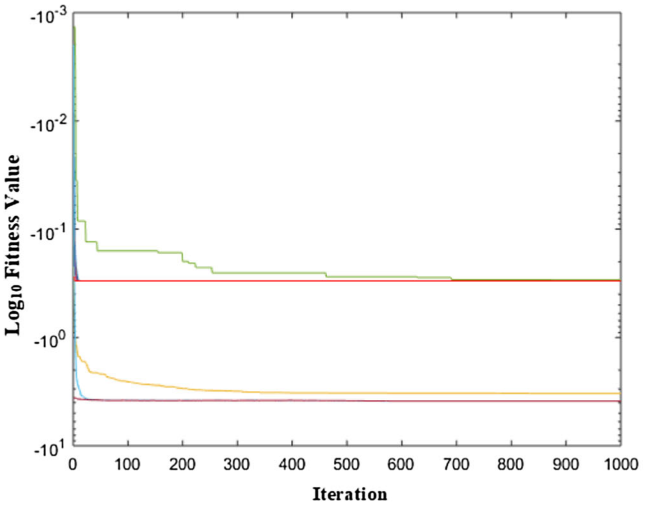
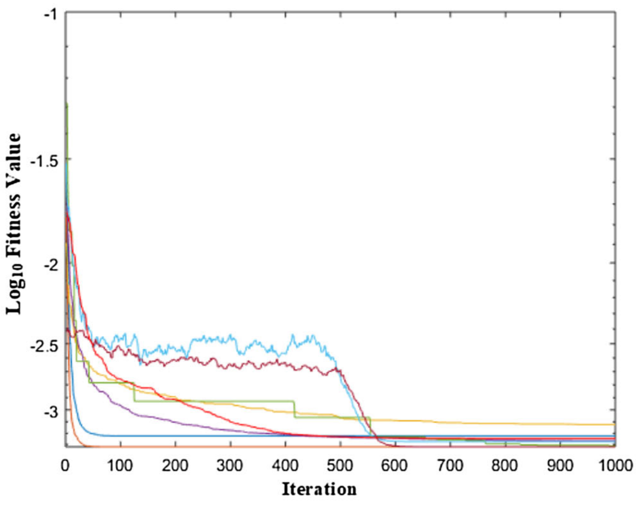


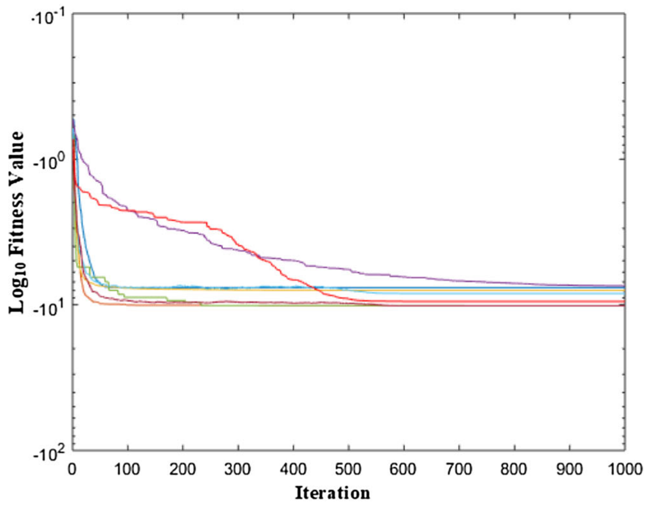
Fig. 4 Convergence curves of the functions F13–F18



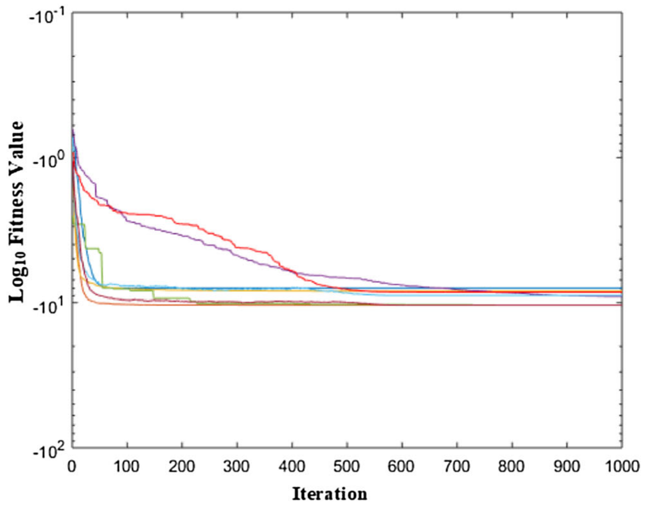
(a) F19



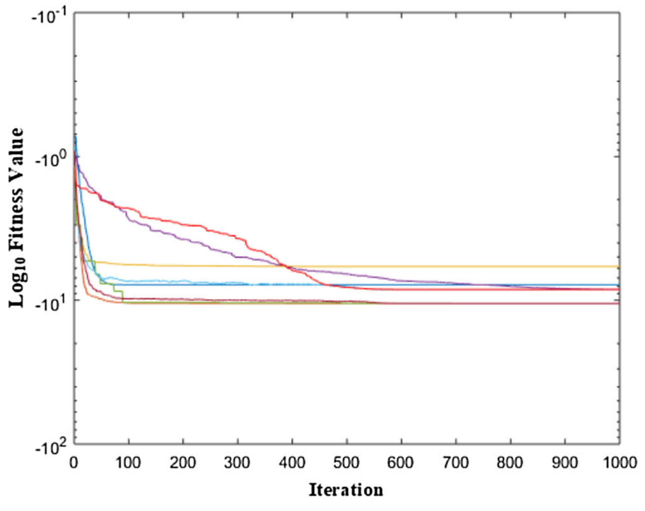
(b) F20



(c) F21



(d) F22



(e) F23

Fig. 5 Convergence curves of F19–F23

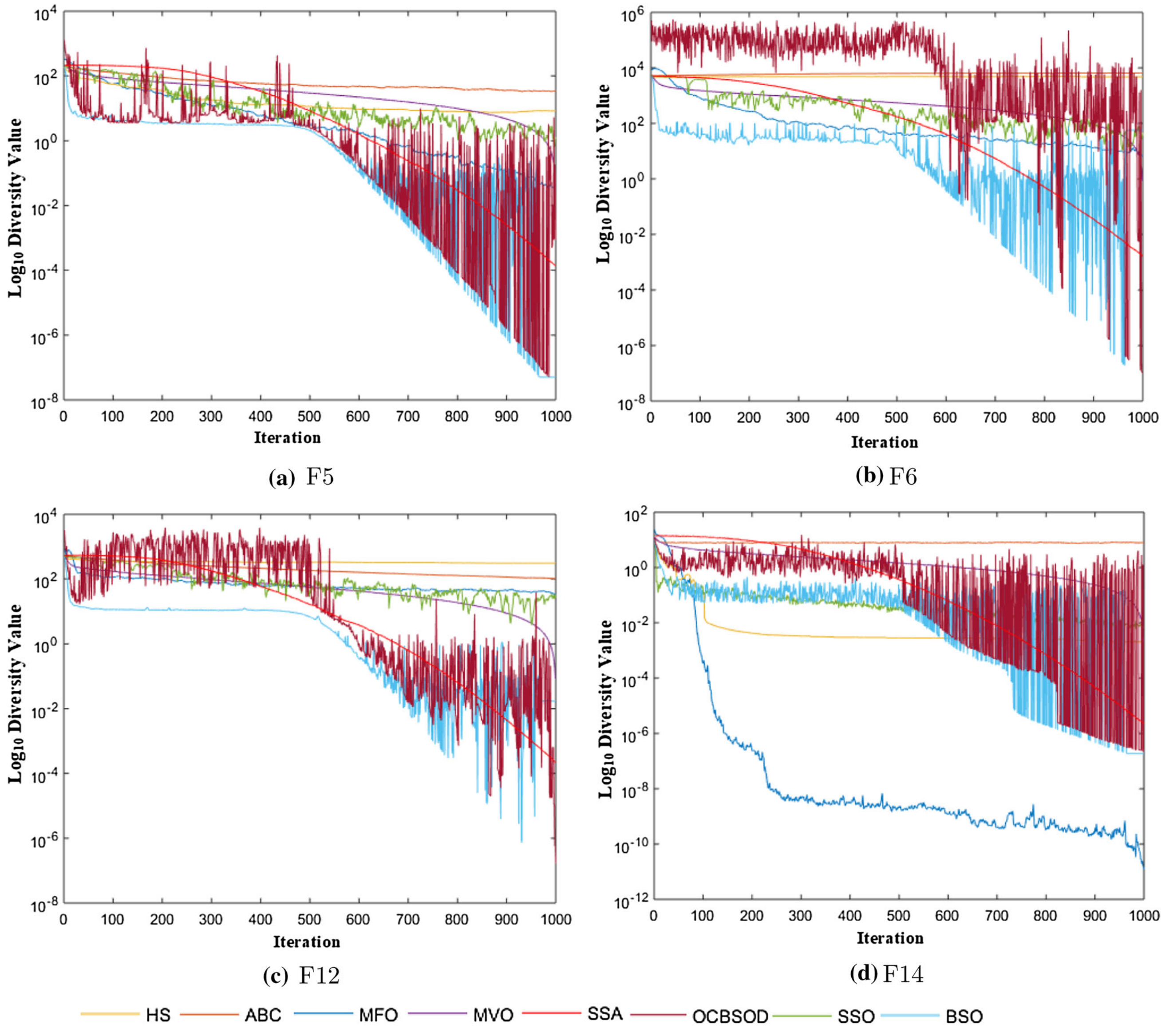


Fig. 6 Diversity for algorithms at F5, F6, F12, and F14

In addition, Table 11 shows the average of CPU time(s), in which it can be seen that the OCBSOD takes the smallest time overall dataset, nearly 33.758s, whereas the SSO algorithm allocates the second rank with 34.436s followed by the HS algorithm, and also the BSO and ABC nearly have needed the same time. Moreover, in terms of the selected features ratio, the OCBSOD has the smallest ratio with value nearly equal to 0.390 overall datasets, followed by the SSO and BSO in the second and third rank, respectively, as in Fig. 8b. Also, for each dataset the proposed OCBSOD gives the better results in BreastCW, Soybean and WaveFN datasets, while BSO gives the smallest ratio of the selected features in Spambase, Congress and Wine. Meanwhile, the HS algorithm is the better in hepatitis and IonoSp datasets as in Fig. 8a.

Figures 9 and 10 show the average of accuracy through running each algorithm 30 runs along each dataset and overall datasets, respectively. From these figures, it can observe that, in general, the BSO has the less accuracy, while the SSO and HS have nearly the same results, as well as, the OCBSOD has the better accuracy. Since there exist two classifiers and eight datasets, so there are 16 cases, the OCBSOD algorithm achieved the first rank with seven cases, also, SSO in the second rank with three cases followed by ABC in the third rank. Also, the HS and BSO have only one case; however, for Soybean dataset, the OCBSOD, SSO, ABC and HS have the same results.

Moreover, Fig. 11 shows the average results of the two classifiers as the average along all datasets, in which from this figure we can conclude that the RF classifier gives the better

Table 7 Comparison results between the proposed OCBSOD and its version OBSOD

Function	Average		Std		Time		SR	
	OCBSOD	OBSOD	OCBSOD	OBSOD	OCBSOD	OBSOD	OCBSOD	OBSOD
F1	5.85E−16	5.27E−16	1.39E−16	1.73E−16	145.23	152.91	100	100
F2	1.70E−07	1.16E−07	2.39E−08	2.54E−08	110.38	112.08	100	100
F3	4.36E−15	2.34E−15	1.56E−15	8.12E−16	460.65	465.84	100	100
F4	2.33E−09	7.42E−09	3.77E−19	1.13E−09	76.61	64.03	100	98
F5	2.91E+01	4.96E+01	3.44E−13	6.71E−01	87.34	66.85	0	0
F6	7.73E+00	3.56E−01	5.77E+00	4.23E−01	115.07	163.68	0	0
F7	1.13E−03	1.04E−03	1.10E−06	8.77E−04	140.53	138.86	0	0
F8	− 3.12E+04	− 2.02E+06	2.53E+03	7.53E+06	129.22	134.59	100	98
F9	0.00E+00	5.68E−15	0.00E+00	1.73E−14	125.49	120.34	100	97
F10	9.47E−09	6.58E−02	1.37E−09	2.91E−02	137.84	167.69	100	100
F11	0.00E+00	5.00E−04	0.00E+00	3.07E−04	169.53	169.73	100	100
F12	1.97E−03	1.32E−04	1.03E−04	2.97E−04	237.37	219.11	0	0
F13	8.81E+00	4.02E−05	1.32E+00	1.76E−04	199.64	190.33	0	0
F14	2.45E+00	2.85E+00	1.28E−01	1.24E+00	252.91	234.69	0	0
F15	5.63E−04	6.76E−04	2.30E−06	2.67E−04	93.33	98.83	100	97
F16	− 1.03E+00	− 1.03E+00	2.30E−16	4.97E−16	67.06	86.92	100	98
F17	3.98E−01	3.98E−01	0.00E+00	0.00E+00	66.79	86.17	0	0
F18	3.00E+00	3.00E+00	2.18E−15	2.42E−15	76.53	75.95	0	0
F19	− 3.47E−01	− 3.86E+00	1.76E−17	1.14E−12	101.18	96.37	100	87
F20	− 3.25E+00	− 3.32E+00	6.83E−02	1.83E−15	105.54	102.45	100	94
F21	− 1.02E+01	− 9.65E+00	5.27E−05	1.54E+00	118.59	110.02	100	90
F22	− 1.04E+01	− 1.04E+01	1.04E−04	7.38E−16	122.75	130.34	100	100
F23	− 1.05E+01	− 1.05E+01	1.56E−03	3.25E−15	142.58	150.51	100	100

Table 8 Description of UCI dataset

No.	Dataset	Classes	Feature	Sample
1	Breast cancer (BreastCW)	2	10	699
2	Hepatitis	2	19	155
3	Spambase	16	57	4601
4	Soybean	19	35	687
5	Ionosphere	2	34	351
6	Waveform (WaveFN)	16	40	500016
7	Congress	435	16	2
8	Wine	3	13	178

results than K-NN classifier. Therefore, the RF classifier is the better classifier, and in this study, that could be combined with the OCBSOD algorithm.

4.3.4 Statistical analysis

In order to statistically analyze the results of the algorithms for FS problem, the Wilcoxon's rank sum test is used as in Table 12. Here, it is important to mention that the stop criteria are set to 10^5 objective function evaluations. From this table, it can be noticed that there exists a significant difference between the OCBSOD and the other methods because the p-value is less than 5%, so we reject the null hypothesis.

From the previous results, one can be concluded that the OCBSOD algorithm provides an effectiveness and efficiency to determine the global solution for the problems with suitable diversity, fast convergence rate. As well as, the accuracy of classification is increased through applying the OCBSOD to find the optimal subset of features. This high performance is achieved due to the good initial population that selected by using the chaotic Singer map to produce a population; then, the OBL strategy is used to enhance it. As well as, the disruption operator makes the OCBSOD algorithm maintain its diversity that leads to a good performance. However, there are some limitations of the proposed method that need to be enhanced such as the disruption operator is applied to update the solutions. However, it is better to use it only for small part of them through using criteria. By this way, the complexity of the OCBSOD can be reduced.

Table 9 Performance measure

Components	Measure name	Formula
Feature selection	The selected features average	$\text{Sel}_{\text{avg}} = \frac{1}{M}\sum_{i=1}^{M}\frac{\lvert x_{\text{best}}^{i}\rvert}{D}$
	Accuracy of classification	$\text{Acc} = \frac{TN+TP}{TN+TP+FN+FP}$
Fitness function	Average of the fitness function	$\text{Average}_{\text{fit}} = \frac{1}{M}\sum_{i=1}^{M}\text{Fit}_i$
	Best fitness function	$\text{Fit}_{\text{Best}} = \max_{i=1}^{M} fit_i^*$
	Worst fitness function	$\text{Fit}_{\text{Worst}} = \min_{i=1}^{M} f_i^*$

Table 10 Results of the fitness function for each algorithm

	OCBSOD	BSO	SSO	ABC	HS
BreastCW					
Average	**0.033**	0.039	0.037	0.038	0.040
Best	**0.015**	0.016	0.015	0.015	0.015
Worst	**0.146**	0.161	0.147	0.158	0.182
Hepatitis					
Average	**0.220**	0.230	0.225	0.229	0.223
Best	0.016	0.015	0.017	0.017	**0.014**
Worst	0.339	0.350	**0.335**	0.349	0.353
Spambase					
Average	**0.055**	0.061	0.062	0.065	0.059
Best	0.003	0.003	0.003	0.003	**0.002**
Worst	0.170	0.191	0.194	0.185	**0.159**
Soybean					
Average	0.022	**0.020**	0.021	0.021	0.021
Best	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Worst	**0.130**	0.149	0.131	0.141	0.131
Ionosphere					
Average	**0.089**	0.108	0.091	0.101	0.097
Best	0.012	0.014	0.012	**0.011**	**0.011**
Worst	**0.210**	0.229	0.221	0.221	0.227
WaveFN					
Average	**0.325**	0.335	0.329	0.331	0.338
Best	0.176	0.180	**0.169**	0.177	0.177
Worst	**0.429**	0.452	0.439	0.451	0.448
Wine					
Average	0.045	0.056	0.054	0.050	**0.039**
Best	**0.025**	0.027	0.027	0.029	0.031
Worst	0.162	0.172	**0.160**	0.163	0.189

Table 11 Average results of CPU time(s)

	OCBSOD	BSO	SSO	ABC	HS
BreastCW	10.309	15.129	11.546	10.459	9.776
hepatitis	3.340	3.756	3.380	4.656	3.938
Spambase	106.908	112.180	109.067	107.454	108.828
Soybean	10.029	10.199	9.581	10.509	9.210
Ionosphere	10.137	11.203	8.633	10.451	9.732
WaveFN	121.415	129.642	126.466	139.640	125.184
Congress	4.017	3.980	3.115	3.765	9.288
Wine	3.906	3.768	3.698	5.837	4.258

Table 12 Wilcoxon's rank sum test results between the feature selection methods

	Statistics	BSO	SSO	ABC	HS
OCBSOD	$p-value$	0.0001	0.002	.0001	0.0001
	h	1	1	1	1

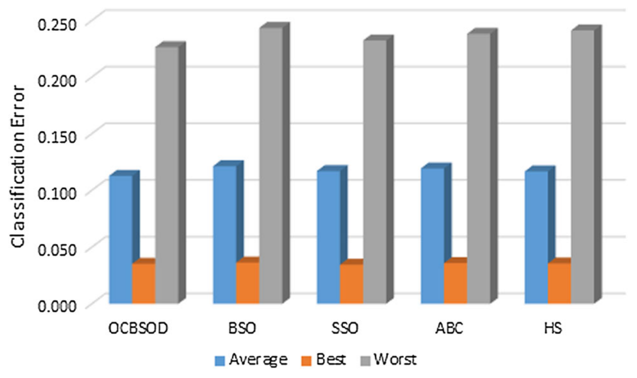


Fig. 7 Average, best and worst fitness function values

5 Conclusions and future works

This paper introduced a modified version of the BSO algorithm called OCBSOD to improve the ability of BSO to exploration and exploitation. The enhancement is performed by generating the initial population using the chaotic Singer map and then benefit from the properties of the OBL to increase the efficiency of the initial population. Thereafter, the solutions of this population are updated using the steps of the BSO algorithm followed by the disruption operator that improves the diversity of the solution and therefore increases the convergence rate. The performance of the OCBSOD algorithm is assessed using a set of three experiments; in the first experiment, the proposed OCBSOD is compared with other seven algorithms, namely BSO, SSO, SSA, MVO, MFO, HS and ABC, to find the optimal solution for functions. According to the results presented in this experiment, the OCBSOD gives the results better than other algorithms regarding all

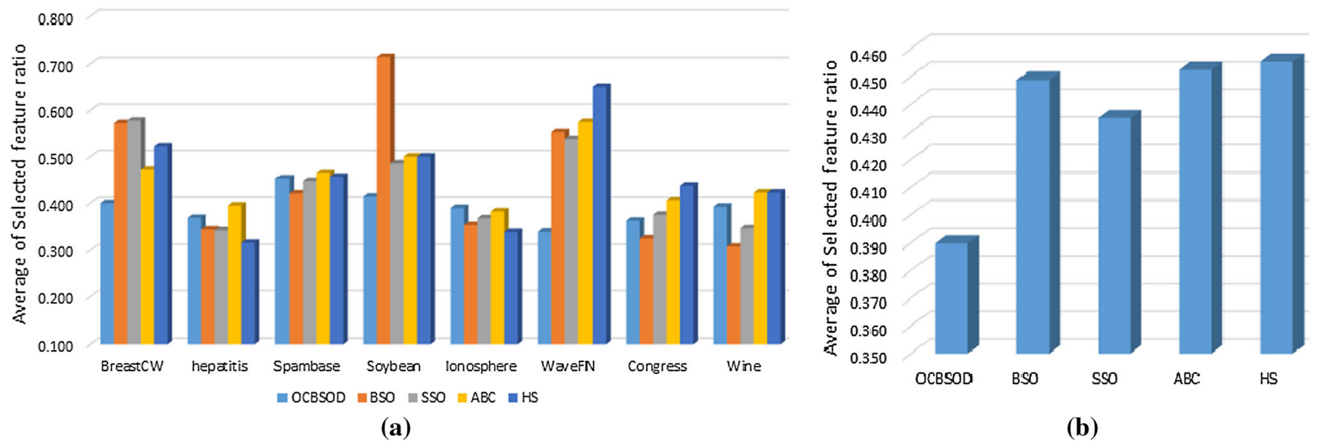


Fig. 8 Average results of selected feature ratio **a** for each dataset, **b** overall dataset

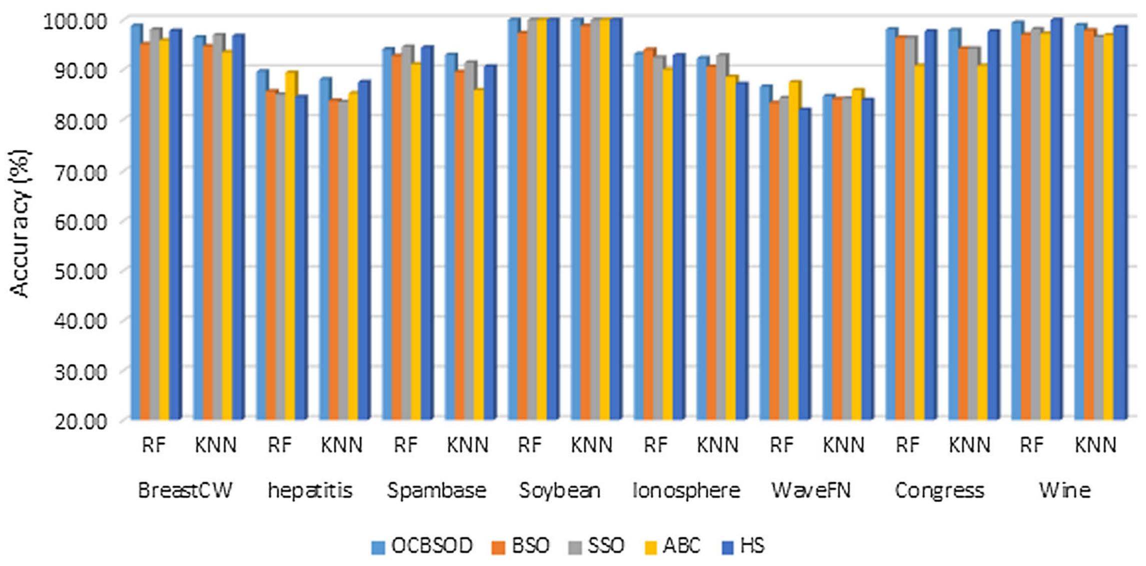


Fig. 9 Classification accuracy value of algorithms along each dataset

performance measures. Meanwhile, in the experimental series 2, the influence of the chaotic singer map is evaluated on the same benchmark functions through generating a random population. The results show that the chaotic singer map affects the performance of the proposed algorithm. To further evaluate the performance of OCBSOD, in the third experiment, the OCBSOD is applied as a feature selection method to find the relevant features from eight datasets to improve the accuracy of classification, where the OCBSOD is compared with other FS methods, namely BSO, SSO, ABC and HS. Based on the performance measures, the classification accuracy of OCBSOD is better than other methods.

Summarizing the main contribution of this article is the combination of the different operators to enhance the searching process in BSO. The inclusion of OBL and DO permits to increase the diversity of the population; meanwhile, the chaotic maps help in the exploitation phase. Another impressive contribution is the application of the proposed approach for feature selection. This implementation permits us to have

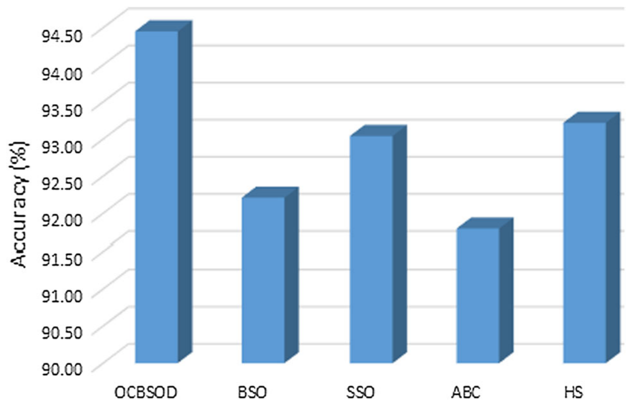


Fig. 10 Classification accuracy value of algorithms overall datasets

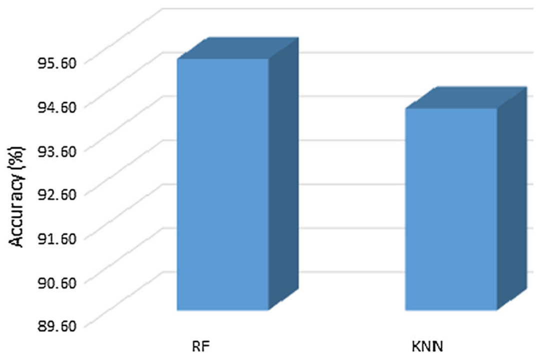


Fig. 11 Average of classification accuracy of two classifiers overall Algorithms

an alternative method to select the best elements from a dataset. The limitation of this work is that the use of the operators could increase the computational effort for specific problems. Here, it is necessary to test it in different domains to verify its performance.

The future scope of this study is to use this algorithm in different applications such as data mining and image processing by considering it as (1) the image segmentation method, (2) multi-objective optimization algorithm, (3) used it to solve the constrained optimization problems, (4) apply it to renewable energy problems and (5) apply the methodology of OCBSOD in other meta-heuristic algorithms to enhance their ability for exploitation and exploration exploitation of the search domain.

Compliance with ethical standards

Conflict of interest It is to specifically state that "no competing interest are at stake and there is no conflict of interest" with other people or organizations that could inappropriately influence or bias the content of the paper.

Financial or personal relationship None of the authors of this paper has a financial or personal relationship with other people or organizations that could inappropriately influence or bias the content of the paper.

Human participants or animals This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abd ElAziz M, Oliva D, Xiong S (2017) An improved opposition-based sine cosine algorithm for global optimization. Exp Syst Appl 90:484–500

Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73:4773–4795

Abualigah LM, Khader AT, Hanandeh ES (2018a) A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. Eng Appl Artif Intell 73:111–125

Abualigah LM, Khader AT, Hanandeh ES (2018b) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48:4047–4071

Abualigah LM, Khader AT, Hanandeh ES (2018c) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25:456–466

Abualigah LMQ (2019) Feature selection and enhanced krill herd algorithm for text document clustering. Springer, Berlin

Abualigah LMQ, Hanandeh ES (2015) Applying genetic algorithms to information retrieval using vector space model. Int J Comput Sci Eng Appl 5:19

Aguirregabiria JM (2009) Robust chaos with variable Lyapunov exponent in smooth one-dimensional maps. Chaos Solitons Fractals 42:2531–2539

Ahmadi S-A (2017) Human behavior-based optimization: a novel metaheuristic approach to solve complex optimization problems. Neural Comput Appl 28:233–244

Cao Z, Hei X, Wang L, Shi Y, Rong X (2015) An improved brain storm optimization with differential evolution strategy for applications of ANNs. Math Prob Eng. https://doi.org/10.1155/2015/923698

Chen J, Cheng S, Chen Y, Xie Y, Shi Y (2015) Enhanced brain storm optimization algorithm for wireless sensor networks deployment. In: Advances in swarm and computational intelligence, Lecture notes in computer science, vol 9140. pp 373–381

Chen J, Xie Y, Ni J (2014) Brain storm optimization model based on uncertainty information. In: 2014 Tenth international conference on computational intelligence and security (CIS). IEEE, pp 99–103

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

Deng W, Xu J, Zhao H (2019) An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. IEEE Access 7:20281–20292

Deng W, Zhao H, Yang X, Xiong J, Sun M, Li B (2017a) Study on an improved adaptive pso algorithm for solving multi-objective gate assignment. Appl Soft Comput 59:288–302

Deng W, Zhao H, Zou L, Li G, Yang X, Wu D (2017b) A novel collaborative optimization algorithm in solving complex optimization problems. Soft Comput 21:4387–4398

El Aziz MA, Hassanien AE (2018) An improved social spider optimization algorithm based on rough sets for solving minimum number attribute reduction problem. Neural Comput Appl 30(8):2441–2452

Ewees AA, El Aziz MA, Hassanien AE (2019) Chaotic multi-verse optimizer-based feature selection. Neural Comput Appl 31(4):991–1006

Frank A, Asuncion A (2010) Uci machine learning repository (http://archive.ics.uci.edu/ml). Irvine, ca: University of california. School of information and computer science 213: 2–2

Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning, 1st edn. Addison-Wesley, Boston

Harwit M (2006) Astrophysical concepts. Springer, Berlin

Jadhav H, Sharma U, Patel J, Roy R (2012) Brain storm optimization algorithm based economic dispatch considering wind power. In: 2012 IEEE International conference on power and energy (PECon). IEEE, pp 588–593

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Comput. Eng. Dep. Eng. Fac. Erciyes Univ

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: IEEE proceedings of international conference on neural networks, vol 4. pp 1942–1948

Krishnanand K, Hasani SMF, Panigrahi BK, Panda SK (2013) Optimal power flow solution using self–evolving brain–storming inclusive teaching–learning–based algorithm. In: International conference in swarm intelligence. Springer, pp 338–345

Labani M, Moradi P, Ahmadizar F, Jalili M (2018) A novel multivariate filter method for feature selection in text classification problems. Eng Appl Artif Intell 70:25–37

Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194:3902–3933

Liu H, Ding G, Wang B (2014) Bare-bones particle swarm optimization with disruption operator. Appl Math Comput 238:106–122

Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249

Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191

Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27:495–513

Sarafrazi S, Nezamabadi-Pour H, Saryazdi S (2011) Disruption: a new operator in gravitational search algorithm. Scientia Iranica 18:539–548

Shi Y (2011) Brain storm optimization algorithm, vol 6728. LNCS, Berlin, pp 303–309

Shi Y (2015) Brain storm optimization algorithm in objective space. In: 2015 IEEE congress on evolutionary computation (CEC). IEEE, pp 1227–1234

Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. KanGAL Rep 2005005:2005

Tian G, Zhang H, Feng Y, Wang D, Peng Y, Jia H (2018a) Green decoration materials selection under interior environment characteristics: a grey-correlation based hybrid MCDM method. Renew Sustain Energy Rev 81:682–692

Tian G, Zhou M, Li P (2018b) Disassembly sequence planning considering fuzzy component quality and varying operational cost. IEEE Trans Autom Sci Eng 15:748–760

Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06), vol 1. pp 695–701

Wilcoxon F (1945) Individual comparisons by ranking methods. Biom Bull 1:80–83

Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82

Yang D, Li G, Cheng G (2007) On the efficiency of chaos optimization algorithms for global optimization. Chaos Solitons Fractals 34:1366–1375

Yang Z, Shi Y (2015) Brain storm optimization with chaotic operation. In: 2015 seventh international conference on advanced computational intelligence (ICACI). IEEE, pp 111–115

Zhan Z-h, Chen W-n, Lin Y, Gong Y-j, Li Y-l, Zhang J (2013) Parameter investigation in brain storm optimization. In: 2013 IEEE symposium on swarm intelligence (SIS). IEEE, pp 103–110

Zhao H, Zheng J, Xu J, Deng W (2019) Fault diagnosis method based on principal component analysis and broad learning system. IEEE Access 7:99263–99272

Zhou D, Shi Y, Cheng S (2012) Brain storm optimization algorithm with modified step-size and individual generation. In: International conference in swarm intelligence. Springer, pp 243–252