A Balanced Butterfly Optimization Algorithm For Numerical Optimization And Feature Selection

Wen Long
Guizhou University of Finance and Economics
Jianjun Jiao
Guizhou University of Finance and Economics
Tiebin Wu
Hunan University of Humanities Science & Technology
Ming Xu
Guizhou University of Finance and Economics
Shaohong Cai (gzcd_csh58@126.com)
Guizhou University of Finance and Economics https://orcid.org/0000-0002-2843-974X

Research Article

Keywords: Butterfly optimization algorithm, Inertia weight, Logistic model, Opposition-based learning, Engineering design problem, Feature selection

Posted Date: February 2nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1310316/v1

A balanced butterfly optimization algorithm for numerical optimization and feature selection

Wen Long[1,2], Jianjun Jiao[3], Tiebin Wu[4], Ming Xu[1,3], Shaohong Cai[2*]

[1] *Key Laboratory of Economics System Simulation*, *Guizhou University of Finance and Economics*, *Guiyang* 550025, *China*
[2] *Key Laboratory of Economics System Simulation*, *Guizhou University of Finance and Economics*, *Guiyang* 550025, *China*
[3] *School of Mathematics and Statistics*, *Guizhou University of Finance and Economics*, *Guiyang* 550025, *China*
[4] *School of Energy and Electrical Engineerig*, *Hunan University of Humanities Science & Technology*, *Loudi* 417000, *China*

ARTICLE INFO

Article history:
Received 8 December 2020

Keywords:
Butterfly optimization algorithm
Inertia weight
Logistic model
Opposition-based learning
Engineering design problem
Feature selection

ABSTRACT

Butterfly optimization algorithm (BOA) is a relatively novel optimization technique for solving function optimization as well as real world applications. However, the paramount challenge in BOA is that it is prone to stagnation in local optima. The purpose of this study is to balance the exploration and exploitation abilities of BOA when two new strategies are introduced. The dynamic inertia weight based on Logistic model as the first strategy is introduced to modify the position updating equation. Another strategy is the opposition-based learning. A new variant of BOA called BBOA based on these two strategies is proposed. Ten widely used benchmark test functions and 30 complex benchmarks from CEC 2014 are selected to verify the effectiveness of BBOA. The used benchmark problems are composed of unimodal, multimodal, rotated, shifted, hybrid and composite functions. The experimental results and analysis show that the proposed BBOA has better exploration ability than the conventional BOA when solving different characteristics functions. Finally, BBOA is applied to solve three real-world engineering applications and sixteen feature selection problems. The results demonstrate that BBOA can outperform other competitors in terms of the accuracy of solution on constrained engineering design and feature selection problems.

1. Introduction

Till date, many nature-inspired meta-heuristic optimization algorithms, such as genetic algorithm (GA), particle swarm optimization (PSO), artificial bee colony (ABC), ant colony optimization (ACO), cuckoo search (CS), bat algorithm (BA), grey wolf optimizer (GWO), firefly algorithm (FA), whale optimization algorithm (WOA), are proposed to handle various search problems and are found in swarm intelligence and evolutionary computation literatures. GA mimics the process of natural selection (Goldberg 1989). PSO is inspired by the social behavior of bird flocking and fish schooling (Kennedy and Eberhart 1995). ABC is based on the principle of honey bee foraging behavior (Karaboga and Basturk 2008). ACO is inspired by the foraging behavior of ant colonies (Blum 2005). CS is based on the obligate brood parasitic behavior of some cuckoo species (Huang et al. 2016). BA mimics the echolocation behavior of microbats (Yang et al. 2010). GWO mimics the social leadership and hunting behavior of grey wolves (Long et al. 2018a). FA is inspired by the idealized behavior of the flashing characteristics of fireflies (Yang 2009). WOA is based on the bubble-net hunting behavior of humpback whales (Long et al. 2020a). The main advantage of these algorithms is their use of the “trial-and-error” principle in searching for solutions. Thus, these algorithms were successfully applied in solving function optimization and real-world engineering optimization problems.

In this paper, we focus on the butterfly optimization algorithm (BOA), which was developed by Arora and Singh (2019). BOA mimics the food searching and mating behavior of biological butterflies in nature. The frame work of BOA is mainly based on the foraging strategy of butterflies, which utilize their sense of smell to determine the location of nectar or mating partner. Similar to other nature-inspired meta-heuristic algorithms, BOA does not require gradient knowledge of the function, easy implementation, and fewer parameters for adjustment. Preliminary studies suggest that the BOA shows excellent performance on function optimization

* Corresponding author.
E-mail address: gzcd_csh58@126.com (S. Cai).

problems, when compared to the other nature-inspired meta-heuristic algorithms. Therefore, BOA is widely applied to solve global optimization and real-world optimization problems (Arora and Anand 2019).

Similar to other nature-inspired meta-heuristic optimization algorithms, the BOA also faces up to some challenging problems. For instance, the basic BOA has tendency to show premature converge to local optima when solving complex multimodal problems. This phenomenon can be attributed to the loss of population diversity in the later stage of iteration. As a matter of fact, convergence and diversity are necessary during the iterative process of BOA. However, the two aspects contradict each other. To avoid falling into local optima by achieving a proper balance between convergence and diversity, many modified version of BOA have been suggested (Arora et al. 2018; Arora and Anand 2018; Arora and Singh 2017; Yuan et al. 2020; Sharma and Saha 2020). While the most BOA variants introduce additional operators or new mechanisms, premature convergence remains a major issue in most existing BOA variants. Thus, the goal of this paper is to balance the exploration and exploitation capabilities of BOA by introducing two strategies. The primary contributions of this study are summarized as follows:

1) A new variant of the BOA, called BBOA, is suggested to solve benchmark test functions and engineering design problems.
2) The dynamic inertia weight based on Logistic model is proposed to effectively balance the exploration and exploitation abilities of the conventional BOA.
3) The opposition-based learning strategy is introduced to further enhance the population diversity.
4) Comprehensive experiments demonstrated that the BBOA obtains excellent performance on function optimization and engineering design problems.

The remainder of this paper is organized as follows. Section 2 provides the preliminary knowledge. The modifications are introduced and the BBOA are explained in Section 3. The BBOA is tested on benchmark problems and real-world engineering applications in Section 4. Section 5 concludes this investigation and presents the future works.

2. Butterfly optimization algorithm

BOA is a new meta-heuristic algorithm developed by Arora and Singh (2019). It simulates the foraging and mating behavior of butterflies. In BOA, the fragrance is formulated as a function of the physical intensity of stimulus as follows:

$$f = cI^a \tag{1}$$

where f represents the perceived magnitude of the fragrance, c denotes the sensory modality, I is the stimulus intensity and a is the power exponent. The search process of BOA is divided two phases, i.e., global search and local search.

There are three phases in BOA: initialization, iteration and final phases. In the initialization phase, the initial parameters and an initial population of butterflies are generated. In the iteration phase, a number of iterations are performed by the algorithm. There are two key steps in the algorithm: global search and local search phases.

In global search phase, the butterfly takes a step toward the optimal butterfly/solution (g^*) :

$$x_i(t+1) = x_i(t) + (r^2 \times g^* - x_i(t)) \times f_i \tag{2}$$

where x_i represents the position vector of ith butterfly, t is the number of iteration, $r \in [0,1]$ is a random number, g^* denotes the global optima, and f_i is the fragrance of ith butterfly.

Local search phase is described as follows:

$$x_i(t+1) = x_i(t) + (r^2 \times x_j(t) - x_k(t)) \times f_i \tag{3}$$

where x_j and x_k are jth and kth butterflies from the population.

* Corresponding author.
E-mail address: gzcd_csh58@126.com (S. Cai).

The above mentioned equations (2) and (3) are used in BOA in a following manner

$$\begin{cases} x_i(t+1) = x_i(t) + (r^2 \times g^* - x_i(t)) \times f_i, & if\ rand < p \\ x_i(t+1) = x_i(t) + (r^2 \times x_j(t) - x_k(t)) \times f_i, & otherwize \end{cases} \tag{4}$$

where $p \in [0,1]$ is a constant number.

In the final phase, till the stopping criteria is not matched, the iteration phase is continued.

The pseudo code of the basic BOA is shown in Algorithm 1.

Algorithm 1 Butterfly optimization algorithm

```
1: Generate initial population of N butterflies
2: Initialize parameters c, a, and p
3: while stopping criteria not met do
4:     for each butterfly in population do
5:         Calculate fragrance using Eq. (1)
6:     end for
7:     Find the best butterfly
8:     for each butterfly in population do
9:         Generate a random number rand in [0,1]
10:        if rand < p then
11:            Update position using Eq. (2)
12:        else
13:            Update position using Eq. (3)
14:        end if
15:    end for
16:    Update the value of a
17: end while
18: Output the best solution found
```

3. Balanced butterfly optimization algorithm

3.1 Modification of position-updating equation

It is well-known that every nature-inspired meta-heuristic algorithm must balance the tradeoff between exploration and exploitation, as it is very important to efficiently find the global best solution. Generally speaking, in the early stages of search process, it is always desired that the search agents are needed to wander throughout the search space rather than gathering around a local optima region (exploration). However, to find the optimum solution of problem, it must converge towards the global best solution in the later stages of search process (exploitation).

BOA as a relatively new nature-inspired meta-heuristic optimization algorithm, the abilities of exploration and exploitation are also needed to balance. According to the Eq. (4), the global best solution (g^*) and the two randomly selected solutions (x_j and x_k) are introduced to balance the exploration and exploitation, respectively. In other words, the individual in the global search phase updates its position by learning from the global best solution of the current population, to accelerate the convergence and improve the exploitation ability. Furthermore, the individual in the local phase renews its position by learning from the current and two randomly selected individuals simultaneously, to improve the diversity and enhance the exploration ability. However, the parameter r in the Eq. (4) is a random number, its ability to balance the exploration and exploitation is limited.

PSO is a nature-inspired meta-heuristic optimization algorithm developed by Kennedy and Eberhart (1995). Empirical studies on PSO with inertia weight (w) have shown that a relatively large w exhibits better global exploration, while a relatively small w results in local exploitation (Long et al. 2020b; Shi and Eberhart 1998; Long et al. 2019b). Inspired by PSO, in order to balance the exploration and exploitation of BOA, we herein introduce an inertia weight (w) and modify the position-updating equation described by Eq. (4) as follows:

$$\begin{cases} x_i(t+1) = w(t) \times x_i(t) + (r^2 \times g^* - x_i(t)) \times f_i, & if\ rand < p \\ x_i(t+1) = w(t) \times x_i(t) + (r^2 \times x_j(t) - x_k(t)) \times f_i, & otherwize \end{cases} \tag{5}$$

where $w \in [0, 1]$ is the inertia weight coefficient.

According to the principle of the SCA, in the early stage of search process, the larger w value means that the search step size is longer and the population search range is wider, which can enhance the global search exploration ability. However, the convergence speed is lower. In order to accelerate the convergence, the value of w should be reduced at a higher speed. In the latter stage of search process, the smaller w value means that the search step size is shorter and the population is concentrated on a small search region, which can improve the local exploitation. However, the population diversity is poor. To avoid the population fall into the local optimum, the value of w should be reduced at a slower speed. Assume that the maximum and minimum values of w are $w_{\max}$ and $w_{\min}$, respectively. In the early stage of the evolution search, assume that the initial decay rate of w is b. The value of b is gradually reduced as the number of iterations increase. When the value of w decreases to the $w_{\min}$, the w will stop decreasing, i.e., the decay rate b is zero. Therefore, the variation of the value of w is in accordance with the Logistic model, and its mathematical formulation is:

$$\begin{cases} \dfrac{d\,w(t)}{dt} = b \cdot \left(1 - \dfrac{w(t)}{w_{\min}}\right) \cdot w(t) \\ w(0) = w_{\max} \end{cases} \tag{6}$$

Using the separation variable method to solve the Eq. (6), and the dynamical adjustment formula of w can be obtained:

$$w(t) = \frac{w_{\min}}{1 + \left(\dfrac{w_{\min}}{w_{\max}} - 1\right) \cdot e^{-bt}} \tag{7}$$

where t is the current iteration, and b is the initial decay rate. From the Eq. (7), when $t = 0$, $w(t) = w_{\max}$, and when $t \to \infty$, it is easy to prove $w(t) = w_{\min}$.

Compared with the position updating Eq. (4) in the basic BOA, the position updating Eq. (5) in our algorithm introduces the dynamic inertia weight w to further balance between convergence speed and population diversity of the BOA.

3.2 Opposition-based learning strategy

In the basic BOA, according to the position updating way of the other butterfly individuals, the new candidate individuals are generated by moving the current individual toward the global best individual (g^*). In the latter stage of evolution search, all of the other butterfly individuals are attracted toward the global best butterfly; they may converge prematurely without enough exploration of search space. Therefore, the basic BOA is prone to premature convergence. Accordingly, the ability to jump to out of local optima has become the most important and attractive goal in BOA improvement.

Opposition-based learning (OBL) is one of the powerful optimization tools developed by Tizhoosh (2005). The OBL strategy has successfully been applied in various meta-heuristic algorithms, such as differential evolution (DE) (Rahnamayan et al. 2008), PSO (Wang et al. 2011), grasshopper optimization algorithm (GOA) (Ewees et al. 2018), sine cosine algorithm (SCA) (Gupta and Deep 2019), shuffled frog leaping (SFL) (Ahandani and Avavi-Rad 2015), antlion optimizer (ALO) algorithm (Dinkar and Deep 2018) used to enhance the exploration ability. Therefore, this paper introduces the OBL strategy to improve the global exploration.

Definition 1. Opposite number. The opposite of real number $x \in [l, u]$ is given by $\hat{x}$:

$$\hat{x} = l + u - x \tag{8}$$

where l and u are the lower and upper bound of search space, respectively.

Definition 2. Opposite point. Suppose $\mathrm{X} = [x_1, x_2, \Lambda\ , x_n]$, where $x_1, x_2, \Lambda\ , x_n \in R$ and $x_i \in [l_i, u_i]$. The

opposite point $\hat{\mathrm{X}}=[\hat{x}_1,\hat{x}_2,\Lambda\ ,\hat{x}_n]$ is defined by:

$$\hat{x}_i = l_i + u_i - x_i,\ i = 1,2,\Lambda\ ,n \tag{9}$$

In OBL strategy, the opposite point $\hat{\mathrm{X}}$ is replaced with it is corresponding solution X based on the fitness function. If $f(\mathrm{X})$ is better than $f(\hat{\mathrm{X}})$, then X not changed, otherwise, $\mathrm{X}=\hat{\mathrm{X}}$.

In this paper, the OBL strategy is applied to the global best individual with a certain probability. The specific implementation is as follows. Suppose $r\in[0,1]$ is a random number, if $r<q$ ($q\in[0,1]$ is a constant number), the OBL strategy is implemented, otherwise, the OBL strategy is not performed. The purpose of this strategy is not performed the OBL strategy every iteration, and thereby reduce the computational complexity of algorithm.

3.3 The flowchart of BBOA

With the descriptions above, the flow chart of the proposed BBOA is shown in Fig. 1.

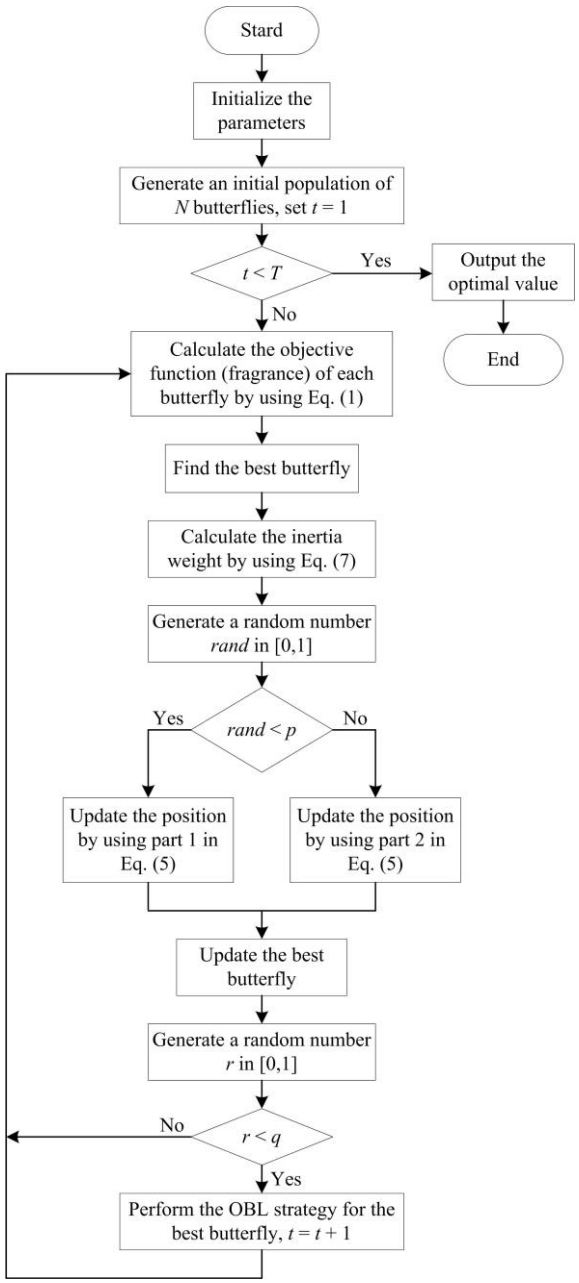


Fig. 1. The flow chart of the proposed BBOA.

4 Simulations and comparisons

To comprehensively investigate the performance of BBOA, a series of experiments are used to handle various mathematical benchmark test cases. All experiments are conducted using MATLAB R2014a software under a Windows 10 operating system and the hardware platform used is configured with Intel (R) Core (TM) i5-5575R CPU (2.80 GHz) and 8GB RAM.

4.1 Experiments on widely used benchmark functions

In this subsection, 10 widely used benchmark functions are chosen from reference (Long et al. 2019a) to investigate the capability of BBOA. The formulation and descriptions of functions are summarized in Table 1.

Table 1

The descriptions of benchmark test functions.

Name	Function formulation	Search range	$f_{\min}$
Sphere	$f_1(\mathbf{X}) = \sum_{i=1}^{D} x_i^2$	[−100, 100]	0
Schwefel 2.21	$f_2(\mathbf{X}) = \max_i\{\lvert x_i \rvert, 1 \le x_i \le D\}$	[−100, 100]	0
Rosenbrock	$f_3(\mathbf{X}) = \sum_{i=1}^{D}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[−30, 30]	0
Quartic	$f_4(\mathbf{X}) = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[−1.28, 1.28]	0
Sumpower	$f_5(\mathbf{X}) = \sum_{i=1}^{D} \lvert x_i \rvert^{(i+1)}$	[−1, 1]	0
Ackley	$f_6(\mathbf{X}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	[−32, 32]	0
Alpine	$f_7(\mathbf{X}) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	[−10, 10]	0
Levy	$f_8(\mathbf{X}) = \sum_{i=1}^{D} (x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) + \lvert x_D - 1 \rvert[1 + \sin^2(3\pi x_D)]$	[−10, 10]	0
Pathological	$f_9(\mathbf{X}) = \sum_{i=2}^{D}\left(0.5 + \frac{\sin^2(\sqrt{100x_{i-1}^2 + x_i^2}) - 0.5}{1 + 0.001(x_{i-1}^2 - 2x_{i-1}x_i + x_i^2)^2}\right)$	[−100, 100]	0
Stretched V-Sine	$f_{10}(\mathbf{X}) = \sum_{i=1}^{D-1} (x_i^2 + 2x_{i+1}^2)^{0.25} \cdot ((\sin 50(x_i^2 + x_{i+1}^2)^{0.1})^2 + 1)$	[−10, 10]	0

In Table 1, these test functions have different characteristics, i.e., unimodal, multi-modal, non-separable, hybrid and composition characteristics. The functions with different characteristics are used to test the different capabilities of algorithm. For instance, the unimodal functions are very beneficial to test the exploitation ability of algorithms since they only have one global optimum. The multimodal functions have many local optima and are very suitable to evaluate the exploration capability of algorithms (Long et al. 2018a).

4.1.1 Results on benchmark functions with 30 dimensions

To verify the effectiveness and efficiency of BBOA, other common algorithms are also implemented on the widely used benchmark functions in Table 1. Thus, the nonlinear inertia weight PSO (NIW-PSO) (Chatterjee and Siarry 2006), covariance matrix adaptation evolution strategy (CMA-ES) (Hansen and Ostermeier 2001), and BOA (Arora and Singh 2019) are selected to optimize these benchmark functions and the results are compared to those obtained by BBOA. For fair comparisons, the parameters of four optimizers are all set in a relatively fair way. Thus, the population size is 30, and the total number of iterations is 500. The dimension of each function is 30. The other best parameters of four algorithms are set using test and trial. In BBOA, $w_{\min} = 0.1$, $w_{\max} = 0.9$, $b = 0.5$. Four algorithms are tested under the same condition and operating system. To reduce random error, each algorithm is independently executed 30 runs. The experimental results of four algorithms in terms of the average value (Mean) and standard deviation (St.dev) of functions are provided in Table 2. It should be noted that the best result of each function is highlighted by bold. Additionally, the Wilcoxon sign rank test (Long et al. 2019a) at 0.05 significant level is utilized to investigate the statistically significant difference between BBOA and other algorithms. The symbols of "+", "≈", and "−" represent that BBOA is superior to, similar to, and worse than the corresponding algorithms, respectively. Meanwhile, the Friedman test is also used to rank the average performance of four algorithms, and the average ranking (AR) and the total ranking (TR) are provided in Table 2.

Table 2
Experimental results of four algorithms on ten benchmark test functions with $D = 30$.

Function	NIW-PSO		CMA-ES		BOA		BBOA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
f_1	2.32E−12	2.11E−12	6.88E−11	3.97E−11	2.50E−11	2.74E−12	**0.00E+00**	**0.00E+00**
f_2	3.87E+00	4.51E−01	2.08E−04	4.23E−05	1.09E−08	1.30E−09	**0.00E+00**	**0.00E+00**
f_3	6.21E+01	2.40E+01	**1.81E+01**	**1.78E−01**	2.90E+01	1.89E−02	2.90E+01	5.49E−03
f_4	2.31E−02	4.43E−03	4.90E−03	2.83E−03	2.47E−03	7.04E−04	**1.51E−04**	**4.54E−05**
f_5	2.12E−31	6.25E−31	2.89E−10	2.63E−10	5.87E−13	3.10E−13	**0.00E+00**	**0.00E+00**
f_6	1.16E+00	1.10E+00	2.80E−06	3.01E−07	1.10E−08	1.10E−09	**8.88E−16**	**0.00E+00**
f_7	5.72E−03	1.25E−03	1.16E−05	5.10E−06	1.29E−08	1.33E−08	**0.00E+00**	**0.00E+00**
f_8	1.51E+01	1.10E+01	3.35E−05	2.73E−05	4.33E−12	5.42E−12	**0.00E+00**	**0.00E+00**
f_9	1.05E+01	5.85E−01	1.25E+01	2.79E−01	1.22E+01	2.18E−01	**0.00E+00**	**0.00E+00**
f_{10}	2.01E+01	3.45E+00	1.79E−01	1.46E−02	4.96E−05	6.55E−05	**0.00E+00**	**0.00E+00**
+/≈/−	10/0/0		9/0/1		9/1/0		−	
AR	3.40		3.10		2.35		1.15	
TR	4		3		2		1	

From Table 2, BBOA can obtain the theoretical optimal values (0) on seven test functions (i.e., f_1, f_2, f_5, and f_7-f_{10}). Compared with NIW-PSO algorithm, BBOA gets better results on all of the test functions. With respect to the CMA-ES algorithm, BBOA finds better results on all of the functions other than f_3. For f_3, the better result is obtained by CMA-ES. Compared to the BOA algorithm, BBOA provides better and similar results on nine and one test functions, respectively. In addition, according to the results of the Wilcoxon's sign rank test, the proposed BBOA is significantly better than NIW-PSO, CMA-ES, and BOA on nine functions. In terms of AR and TR values, BBOA obtains the first average ranking, followed by BOA, CMA-ES, and NIW-PSO, respectively.

To further investigate the advantages of BBOA, the convergence curves of four algorithms for four typical test functions (i.e., f_2, f_5, f_6, and f_{10}) are provided in Fig. 2.

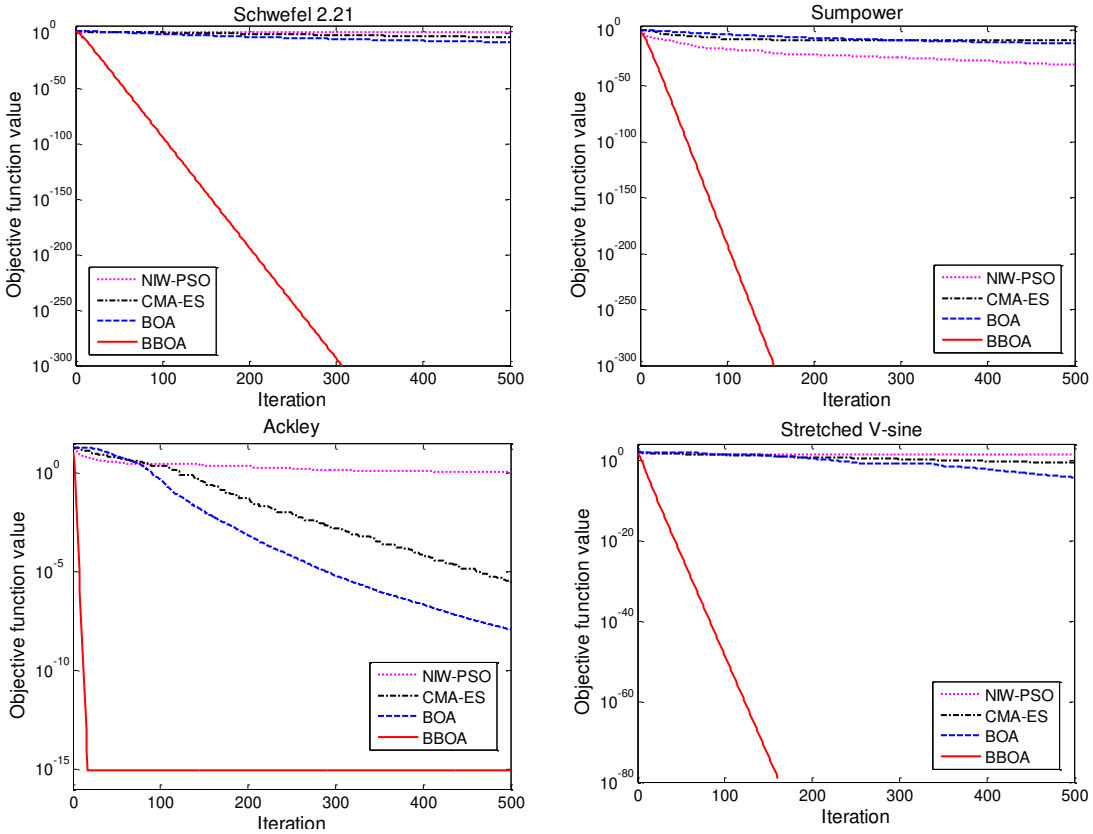


Fig. 2. Convergence curves of four algorithms for four typical test functions with $D = 30$.

As can be seem from Fig. 2, the proposed BBOA can obtain fast convergence and high precision compared to the other three algorithms on four typical test functions with $D = 30$.

4.1.2 Scalability test

The scalability test can evaluate the impact of dimension on both excellence of results and effectiveness of optimizers, simultaneously (Long et al. 2019b). To further test the scalability of the proposed method, the BBOA is applied to solve the higher dimensional functions (i.e., $D = 100$ and $D = 1000$). In this experiment, 10 benchmark test functions from Table 1 are used. We used the same parameter settings as in the experiments above, and maintain the population size or number of fitness function evaluations. The average (Mean) and standard deviation (St.dev) of objective function values obtained by BBOA and other three algorithms, the results of the Wilcoxon's sign rank test, and the results of Friedman's test are reported in Tables 3 and 4, respectively. The best result of each function is marked in bold.

Table 3

Experimental results of four algorithms on ten benchmark test functions with $D = 100$.

Function	NIW-PSO		CMA-ES		BOA		BBOA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
f_1	6.13E+01	2.35E+01	1.20E−02	1.70E−03	2.51E−11	1.94E−12	**0.00E+00**	**0.00E+00**
f_2	1.94E+01	1.42E+00	6.39E−01	1.45E−01	1.28E−08	1.04E−09	**0.00E+00**	**0.00E+00**
f_3	2.71E+03	6.77E+02	1.63E+02	7.01E+01	**9.90E+01**	**1.64E−02**	**9.90E+01**	**2.03E−02**
f_4	5.33E−01	1.36E−01	1.50E−02	4.86E−03	4.50E−03	2.89E−03	**3.71E−04**	**3.74E−04**
f_5	2.55E−24	9.63E−24	1.28E−08	1.08E−08	6.23E−13	2.42E−13	**0.00E+00**	**0.00E+00**
f_6	5.08E+00	6.91E−01	1.92E−02	4.00E−03	1.20E−08	1.89E−09	**8.88E−16**	**0.00E+00**
f_7	2.99E+00	1.87E+00	1.44E−01	6.52E−03	1.45E−08	2.72E−08	**0.00E+00**	**0.00E+00**
f_8	1.94E+02	5.24E+01	9.92E−01	8.26E−04	3.06E−11	7.54E−12	**0.00E+00**	**0.00E+00**
f_9	4.35E+01	1.39E+01	4.65E+01	1.57E−01	4.46E+01	1.04E+00	**0.00E+00**	**0.00E+00**
f_{10}	1.14E+02	6.06E+00	1.36E+01	8.26E−01	5.59E−06	4.76E−06	**0.00E+00**	**0.00E+00**
+/≈/−	10/0/0		10/0/0		9/1/0		−	
AR	3.60		3.20		2.15		1.05	
TR	4		3		2		1	

Table 4

Experimental results of four algorithms on ten benchmark test functions with $D = 1000$.

Function	NIW-PSO		CMA-ES		BOA		BBOA	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
f_1	9.40E+04	6.75E+03	8.70E+03	5.12E+02	2.88E−11	5.28E−12	**0.00E+00**	**0.00E+00**
f_2	4.10E+01	4.13E+00	3.04E+01	5.31E+00	1.46E−08	1.12E−09	**0.00E+00**	**0.00E+00**
f_3	2.53E+07	5.37E+06	6.62E+05	7.50E+04	**9.99E+02**	**1.29E−02**	**9.99E+02**	**2.29E−03**
f_4	3.82E+02	5.59E+01	5.92E+01	4.76E+00	6.60E−03	3.07E−03	**4.54E−04**	**7.28E−05**
f_5	3.58E−18	3.25E−18	4.56E+00	1.35E+00	8.09E−13	5.10E−13	**0.00E+00**	**0.00E+00**
f_6	1.22E+01	2.17E−01	5.57E+00	4.26E−01	1.27E−08	1.33E−09	**8.88E−16**	**0.00E+00**
f_7	4.55E+02	1.98E+01	3.63E+02	1.10E+01	1.87E−07	4.11E−07	**0.00E+00**	**0.00E+00**
f_8	6.94E+03	1.00E+03	2.33E+04	3.50E+03	3.42E−11	2.35E−12	**0.00E+00**	**0.00E+00**
f_9	4.82E+02	4.29E+00	4.92E+02	3.89E+00	4.75E+02	1.35E−01	**0.00E+00**	**0.00E+00**
f_{10}	1.64E+03	5.09E+01	1.20E+03	3.49E+00	3.02E−06	1.86E−06	**0.00E+00**	**0.00E+00**
+/≈/−	10/0/0		10/0/0		9/1/0		−	
AR	3.60		3.30		2.05		1.05	
TR	4		3		2		1	

From Tables 3-4, except for the function f_3 with $D = 100$ and $D = 1000$, BBOA showed very good scalability to the search dimension for the other functions, in other words, the performance of BBOA did not deteriorate seriously as the dimension increased. It should be emphasized that the problem optimization for 1000 dimensions was very challenging for BOA because it does not use any particular operators tailored to solve high-dimensional optimization. In addition, Figs. 3-4 provided the convergence curves of four algorithms on four typical functions

with D = 100 and D = 1000. As can be seen from Figs. 3-4, BBOA can reveal obvious advantages over other three algorithms in terms of convergence and precision.

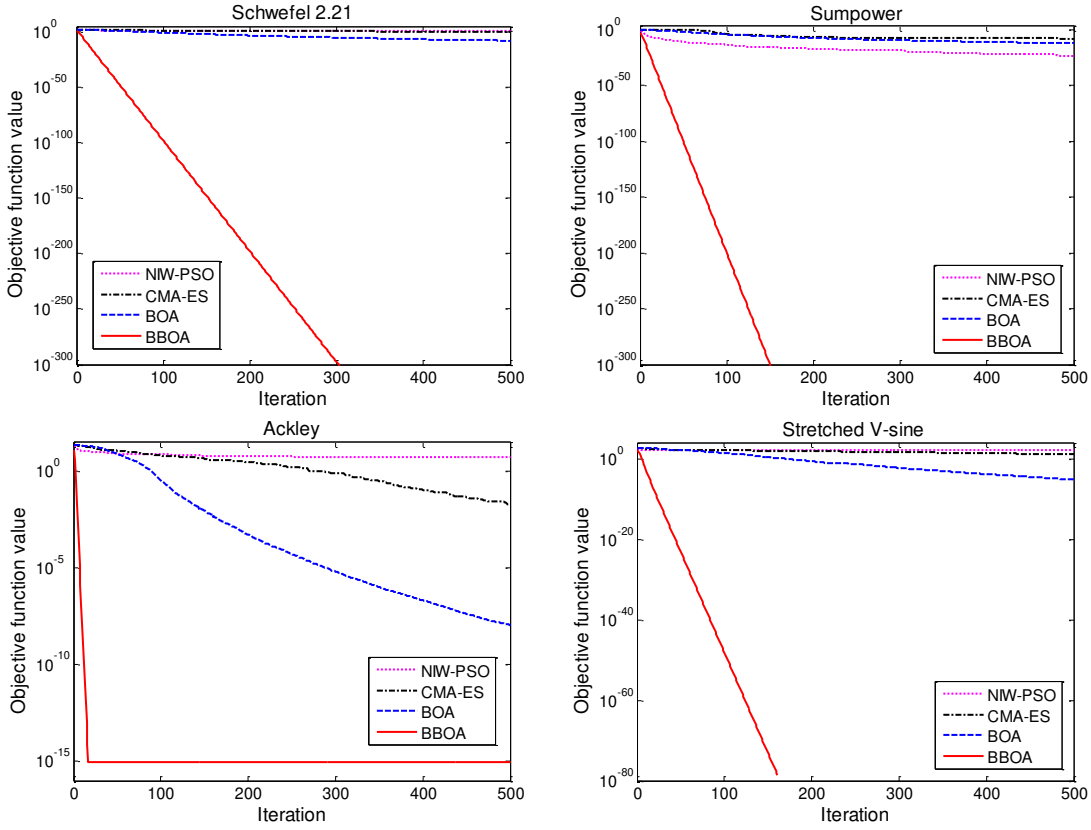


Fig. 3. Convergence curves of four algorithms for four typical test functions with D = 100.

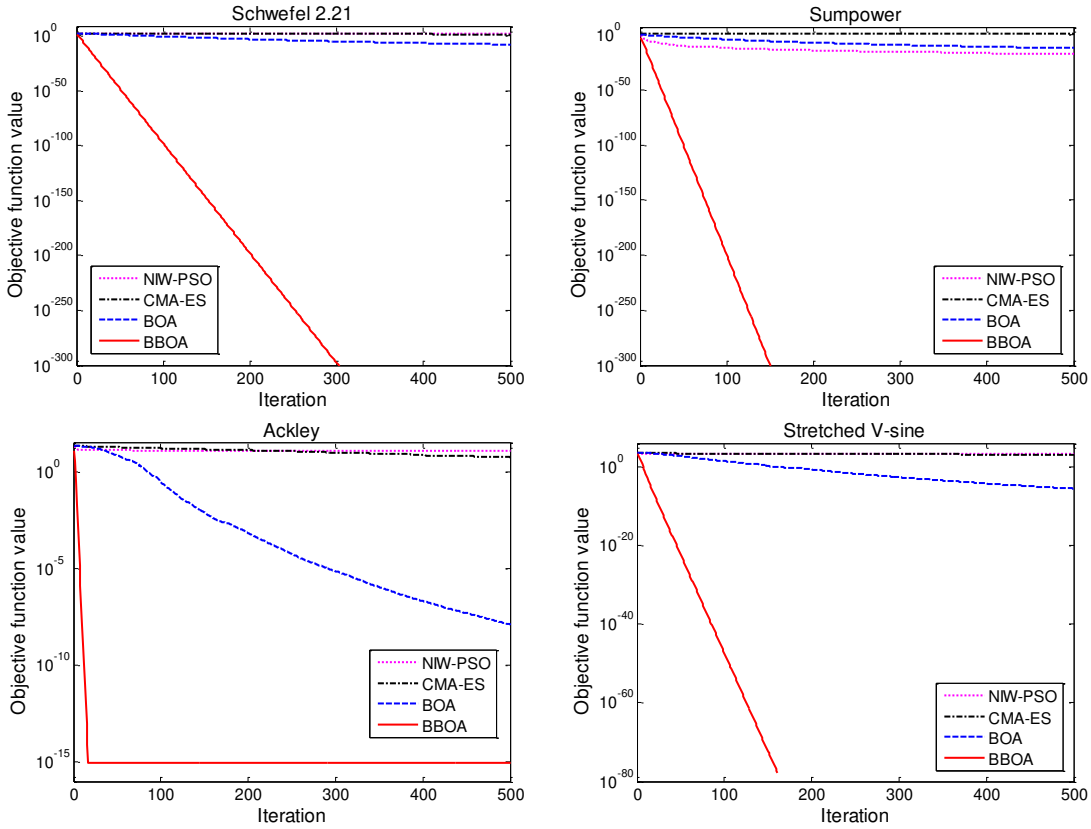


Fig. 4. Convergence curves of four algorithms for four typical test functions with D = 1000.

4.1.3 *The impact of w_{min} and w_{max}*

In BBOA, the inertia weight w is an important parameter to effectively balance between convergence and diversity. However, from the Eq. (7), the value of w is mainly determined by w_{min} and w_{max}. Some experiments are conduced to investigate the impact of w_{min} and w_{max}. According to the repeated test and trials, we concluded that the proposed BBOA is not sensitive to the parameter w_{max}. Therefore, in this subsection, we only analyze the impact of w_{min} by some experiments. We manipulated the value of w_{min} while keeping the other parameters fixed. Vales w_{min} = 0.1, w_{min} = 0.3, w_{min} = 0.5, and w_{min} = 0.7 are examined for all of the 10 functions with D = 30 in Table 1. It should be noted that, in previous experiments, the value of w_{min} is 0.1. For comparison, the results related to w_{min} = 0.1 are also reported. The Mean and St.dev results of BBOA using different w_{min} values are provided in Table 5.

Table 5

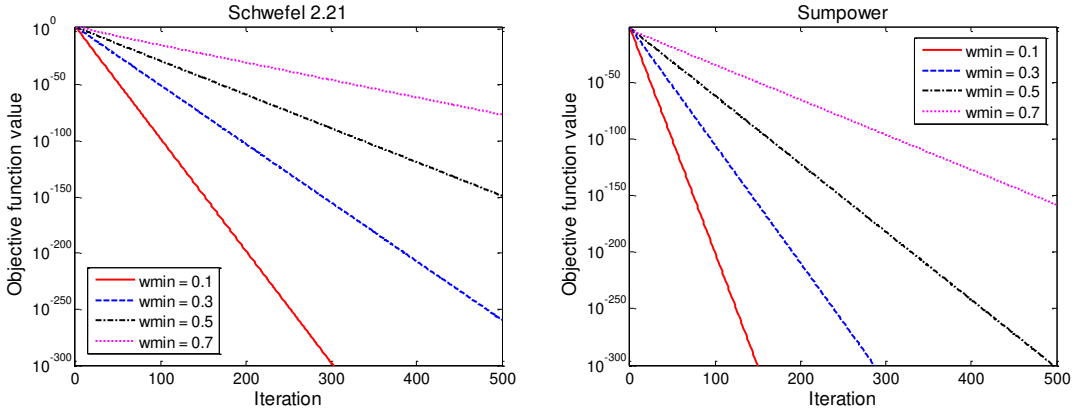
Experimental results of BBOA using different w_{min} values on ten benchmark functions with D = 30.

Function	w_{min} = 0.1		w_{min} = 0.3		w_{min} = 0.5		w_{min} = 0.7	
	Mean	St.dev	Mean	St.dev	Mean	St.dev	Mean	St.dev
f_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	2.49E−297	0.00E+00	3.94E−152	3.30E−153
f_2	0.00E+00	0.00E+00	3.34E−260	0.00E+00	1.23E−149	2.40E−151	3.28E−77	2.26E−78
f_3	2.90E+01	5.49E−03	2.90E+01	1.25E−02	2.90E+01	1.40E−03	2.90E+01	9.53E−03
f_4	1.51E−04	4.54E−05	4.87E−04	6.41E−04	5.09E−04	5.72E−04	8.66E−04	3.80E−04
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.31E−303	0.00E+00	1.25E−156	0.00E+00
f_6	8.88E−16	0.00E+00	8.88E−16	0.00E+00	8.88E−16	0.00E+00	8.88E−16	0.00E+00
f_7	0.00E+00	0.00E+00	5.30E−261	0.00E+00	2.23E−150	2.70E−151	6.84E−78	6.41E−79
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.49E−299	0.00E+00	2.13E−153	1.40E−153
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	6.50E+00	5.97E+00
f_{10}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.30E−75	4.19E−76	5.21E−39	4.97E−40

From Table 5, compared with BBOA using w_{min} = 0.3, BBOA with w_{min} = 0.1 obtains better and similar results on three (f_2, f_4 and f_7) and seven functions, respectively. The performance of BBOA using w_{min} = 0.1 has better than w_{min} = 0.5 on seven test functions. For the other three (f_3, f_6 and f_9) test functions, BBOA with w_{min} = 0.1 and w_{min} = 0.5 find similar results. With respect to the BBOA with w_{min} = 0.7, BBOA using w_{min} = 0.1 gets better results on all of the test functions except for f_3 and f_6. For f_3 and f_6, two algorithms obtain similar results.

In addition, Fig. 5 shows the convergence curves of BBOA using different w_{min} values for four typical test functions (f_2, f_5, f_6 and f_{10}) with D = 30. As can be seen from Fig. 5, BBOA using w_{min} = 0.1 has higher precision and faster convergence than other three cases on f_2 (Schwefel 2.21), f_5 (Sumpower), and f_{10} (Stretched V-sine) functions. For f_6 (Ackley), BBOA using w_{min} values obtain similar results. However, BBOA with w_{min} = 0.1 has better faster convergence than other three cases.

From Table 5 and Fig. 5, according to all of the w_{min} values analyzed, we concluded that the setting of w_{min} = 0.1 for the BBOA is an appropriate choice.



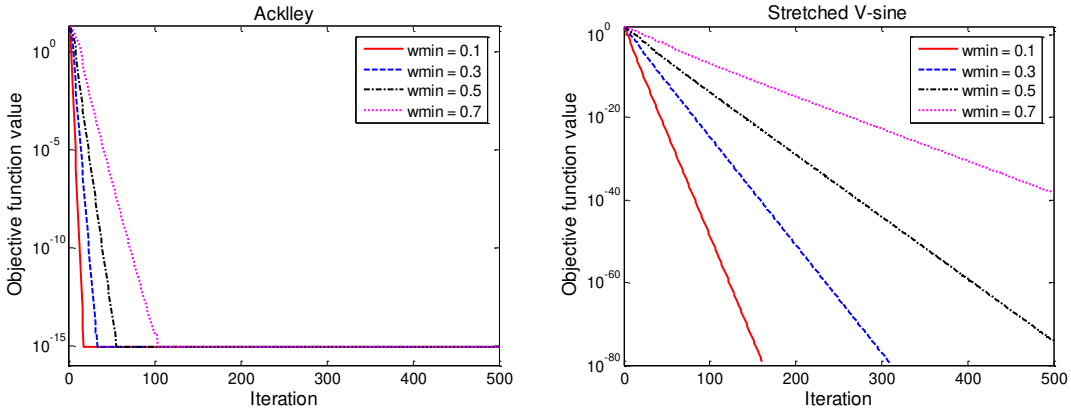


Fig. 5. Convergence curves of BBOA using different $w_{\min}$ values for four typical test functions with $D = 30$.

4.2 Experiments on complex benchmark functions from CEC 2014

In this subsection, the effectiveness and efficiency of BBOA is further verified by using the other 30 functions from IEEE CEC 2014 special session, which is more difficult than the 10 widely used benchmark functions in Table 1. The 30 test functions are categorized into four groups, i.e., Unimodal (F01-F03), Multimodal (F04-F16), hybrid (F17-F22), and composite functions (F23-F30). The detailed information of 30 functions can be found in (Liang et al. 2013). The search range of these 30 test functions is [−100, 100]. These 30 benchmark test functions are rotated, shifted, hybrid and composite functions, respectively. The source codes for these 30 benchmark test functions are available in http://www3.ntu.edu.sg/home/EPNSugan/.

The results of BBOA are compared with the basic BOA, three traditional competitive approaches, and two up-to-date methods. CMA-ES (Hansen and Ostermeier 2001) is a well-known evolution strategy with covariance matrix adaptation and is a traditional competitive method; CLPSO (Liang et al. 2006) is a well-performance particle swarm optimization based on comprehensive learning concept and is a classical competitive approaches; JADE (Zhang and Sanderson 2009) is a well-known adaptive differential evolution with optional external archive and is a traditional competitive meta-heuristic optimization algorithm; RW-GWO (Gupta and Deep 2019a) is an improved version of grey wolf optimizer with random walk concept and is an up-to-date meta-heuristic algorithm developed in 2019; ISCA (Gupta and Deep 2019b) is a novel variant of sine cosine algorithm with crossover operator and is an up-to-date optimization technique developed in 2019. The above mentioned five meta-heuristic optimization techniques represent the state-of-the-art in ES, PSO, DE, GWO, and SCA algorithms respectively and their performance is very competitive. The parameters of five selected approaches are set as their original papers. For a fair comparison, the same maximum number of function evaluations (FEs) is set to $10^4 \times D$ (D is the dimension of the function), which is the termination criteria for seven algorithms. For each test function, the error values ($f(x)-f(x_0)$) are calculated. It is noted that x is the best solution when a method ends and x_0 is the global optimal solution. For each algorithm, the mean error values (Mean) and the standard deviation of function values (St.dev) are calculated over 30 independent trials for each test function. In addition, the Friedman's ranking test results of each algorithm are provided. Comparison results are listed in Table 6. When a method achieved the best performance on the corresponding test function, the mean value is highlighted with bold.

From Table 6, compared with CMA-ES algorithm, the proposed BBA gets better results on 23 benchmark test functions. However, for seven functions (i.e., F01, F05, F10-F12, F17, and F21), the better results is obtained by CMA-ES. With respect to the CLPSO algorithm, BBA finds better results on all of the test functions. Compared to the RW-GWO algorithm, BBA provides better and worse results on 16 and 12 test functions, respectively. In addition, for F24 and F26, RW-GWO and BBA obtain similar results. BBOA outperforms ISCA on 25 benchmark test functions. However, the better results obtained by ISCA for the four functions (i.e., F01, F04, F21, and F22).

Table 6
Experimental results of seven algorithms on CEC 2014 benchmark functions with $D = 30$.

Function	Criterion	CMA-ES	CLPSO	JADE	RW-GWO	ISCA	BOA	BBOA
F01	Mean	9.42E+04	1.48E+08	**4.48E+02**	8.02E+06	1.427E+07	6.10E+08	1.59E+07
	St.dev	7.89E+04	2.57E+07	1.04E+03	3.31E+06	5.937E+06	2.08E+08	2.88E+06
F02	Mean	2.55E+10	6.81E+09	**0.00E+00**	2.23E+05	3.131E+08	4.70E+10	3.11E+08
	St.dev	3.85E+09	1.12E+09	0.00E+00	5.51E+05	2.073E+08	5.81E+09	1.46E+07
F03	Mean	1.45E+04	9.86E+04	**5.63E−04**	3.16E+02	2.629E+03	9.17E+04	5.18E+02
	St.dev	5.66E+03	1.68E+04	2.51E−03	4.34E+02	1.682E+03	7.94E+03	8.92E+01
F04	Mean	2.52E+03	9.77E+02	**0.00E+00**	3.41E+01	1.466E+02	1.38E+04	1.87E+02
	St.dev	5.36E+02	1.39E+02	0.00E+00	1.80E+01	2.550E+01	5.85E+02	2.55E+01
F05	Mean	**2.00E+01**	2.11E+01	2.03E+01	2.05E+01	2.086E+01	2.10E+01	2.03E+01
	St.dev	2.63E−05	4.87E−02	3.79E−02	7.46E−02	9.131E−02	2.71E−02	2.95E−02
F06	Mean	4.09E+01	5.08E+01	9.42E+00	9.84E+00	8.333E+00	3.56E+01	**7.26E+00**
	St.dev	2.13E+00	2.46E+00	2.16E+00	3.49E+00	1.826E+00	1.53E+00	1.68E+00
F07	Mean	2.31E+02	6.32E+01	**0.00E+00**	2.53E−01	3.731E+00	6.18E+02	1.93E+00
	St.dev	2.83E+01	8.63E+00	0.00E+00	1.43E−01	1.859E+00	6.74E+01	5.15E−01
F08	Mean	2.83E+02	2.92E+02	**0.00E+00**	4.38E+01	2.937E+01	2.80E+02	2.75E+01
	St.dev	2.21E+01	1.87E+01	0.00E+00	8.48E+00	6.929E+00	1.83E+01	8.24E−01
F09	Mean	3.28E+02	4.73E+02	**2.62E+01**	6.33E+01	6.115E+01	4.12E+02	2.96E+01
	St.dev	3.47E+01	2.13E+01	4.18E+00	1.30E+01	1.526E+01	8.19E+01	1.15E+01
F10	Mean	2.61E+02	7.62E+03	**5.31E−03**	9.61E+02	6.770E+02	6.94E+03	6.43E+02
	St.dev	1.06E+02	5.19E+02	1.01E−02	2.72E+02	2.522E+02	2.46E+02	1.51E+02
F11	Mean	**1.69E+02**	1.14E+04	1.64E+03	2.68E+03	2.427E+03	7.15E+03	2.29E+03
	St.dev	1.98E+02	5.09E+02	2.64E+02	3.68E+02	6.261E+02	3.63E+02	3.83E+02
F12	Mean	3.03E−01	2.67E+00	**2.71E−01**	5.45E−01	1.605E+00	2.52E+00	9.92E−01
	St.dev	2.18E+00	3.28E−01	3.75E−02	1.66E−01	3.519E−01	2.17E−01	4.49E−02
F13	Mean	5.51E+00	7.61E−01	**2.20E−01**	2.80E−01	3.454E−01	7.46E+00	3.10E−01
	St.dev	3.07E−01	8.47E−02	3.91E−02	6.30E−02	6.063E−02	2.51E−01	5.41E−02
F14	Mean	7.53E+01	1.60E+01	**2.34E−01**	4.23E−01	6.459E−01	2.35E+02	4.12E−01
	St.dev	8.08E+00	3.71E+00	3.00E−02	2.15E−01	2.178E−01	1.62E+01	6.00E−02
F15	Mean	1.02E+04	3.31E+03	**3.10E+00**	8.81E+00	1.327E+01	1.21E+05	9.36E+00
	St.dev	3.24E+04	1.93E+03	4.30E−01	1.51E+00	2.877E+00	5.86E+04	2.42E+00
F16	Mean	1.38E+01	2.24E+01	9.37E+00	1.03E+01	1.068E+01	1.26E+01	**9.13E+00**
	St.dev	5.31E−01	2.33E−01	3.64E−01	6.11E−01	5.420E−01	3.22E−01	2.88E−01
F17	Mean	**5.49E+03**	1.77E+07	9.67E+03	5.71E+05	4.538E+05	2.29E+07	4.32E+05
	St.dev	3.62E+03	4.74E+06	5.99E+04	4.10E+05	4.076E+05	1.06E+07	3.01E+05
F18	Mean	1.52E+09	2.51E+07	**3.58E+02**	6.52E+03	5.833E+03	1.33E+09	5.39E+03
	St.dev	3.93E+08	8.22E+06	1.51E+03	4.62E+03	1.197E+04	6.89E+08	4.73E+03
F19	Mean	2.98E+02	9.23E+01	**4.44E+00**	1.14E+01	1.185E+01	4.39E+02	1.16E+01
	St.dev	4.25E+01	1.19E+01	6.70E−01	2.03E+00	2.084E+00	3.38E+01	2.05E+00
F20	Mean	4.61E+03	5.17E+04	2.89E+03	**6.27E+02**	1.729E+03	3.94E+04	6.77E+02
	St.dev	3.88E+03	1.06E+04	2.33E+03	1.12E+03	1.401E+03	9.43E+03	2.61E+02
F21	Mean	**6.86E+03**	5.71E+06	7.58E+03	2.58E+05	1.579E+05	5.90E+07	3.17E+05
	St.dev	2.76E+03	2.15E+06	3.79E+04	1.76E+05	1.274E+05	2.35E+07	2.26E+05
F22	Mean	1.61E+03	1.36E+03	**1.46E+02**	2.08E+02	2.206E+02	2.58E+03	2.69E+02
	St.dev	9.15E+01	1.71E+02	7.25E+01	1.29E+02	8.724E+01	2.73E+02	7.55E+01
F23	Mean	5.79E+02	3.90E+02	3.15E+02	3.15E+02	3.177E+02	8.99E+02	**2.00E+02**
	St.dev	4.94E+01	8.19E+00	2.30E−13	2.77E−01	1.278E+00	3.62E+02	0.00E+00
F24	Mean	2.12E+02	3.39E+02	2.26E+02	**2.00E+02**	**2.000E+02**	4.53E+02	**2.00E+02**
	St.dev	7.49E+00	6.72E+00	3.59E+00	3.43E−03	1.680E−03	1.12E+02	0.00E+00
F25	Mean	2.12E+02	2.46E+02	2.04E+02	2.04E+02	2.064E+02	2.88E+02	**2.00E+02**
	St.dev	2.97E+02	5.30E+00	1.27E+00	1.18E+00	2.236E+00	3.34E+01	0.00E+00
F26	Mean	1.25E+02	1.10E+02	1.02E+02	**1.00E+02**	1.062E+02	1.28E+02	**1.00E+02**
	St.dev	5.51E+01	2.83E+01	1.41E+01	7.36E−02	2.368E+01	4.02E+01	0.00E+00
F27	Mean	1.07E+03	1.33E+03	3.44E+02	4.09E+02	4.736E+02	4.62E+02	**2.00E+02**
	St.dev	2.30E+02	3.66E+02	5.16E+01	6.09E+00	6.618E+01	1.80E+01	0.00E+00
F28	Mean	2.79E+03	3.02E+03	8.03E+02	4.34E+02	8.718E+02	1.53E+03	**2.00E+02**
	St.dev	5.92E+02	4.20E+02	3.58E+01	8.45E+00	5.716E+01	7.23E+01	0.00E+00
F29	Mean	3.52E+04	4.10E+05	7.59E+02	2.14E+02	2.254E+05	5.28E+02	**2.00E+02**
	St.dev	5.34E+03	1.64E+05	1.59E+02	2.37E+00	1.491E+06	2.34E+01	0.00E+00
F30	Mean	6.48E+05	6.00E+04	1.92E+03	6.69E+02	8.049E+03	5.89E+02	**2.00E+02**
	St.dev	1.31E+05	1.52E+04	6.41E+02	2.14E+02	2.631E+03	1.95E+02	0.00E+00
Average ranking		4.73	5.87	1.95	2.93	3.83	6.07	2.43
Total ranking		5	6	1	3	4	7	2

Additionally, BBOA and ISCA find similar result for the F24 function. The performance of BBOA is significantly better than the basic BOA on all of the test functions. JADE is one of the most competitive meta-heuristic optimization algorithms. Compared with JADE algorithm, BBOA obtains better and worse results on 11 and 18 test functions, respectively. In addition, BBOA and JADE find similar result for the function F05. Regarding the results of the Friedman's ranking test, JADE ranked the first, followed by BBOA, RW-GWO, ISCA, CMA-ES, CLPSO, and BOA. To sum up, BBOA shows a quite competitive performance with respect to the six selected algorithms in terms of the quality and robustness for the CEC 2014 benchmark test functions.

4.3 Experiments on engineering design problems

This subsection further investigates the performance of the proposed BBOA by solving three constrained real engineering design problems (Coello and Montes 2002), i.e., tension/compression spring design, pressure vessel design, and three-bar truss design. These design problems are widely discussed in the literature and have been solved to better clarify the performance of the algorithms. The Deb's feasibility-based rule (Deb 2000) is used to handle the constraints. In BBOA, the population size is 30 and the maximum number of iterations is 1000.

4.3.1 Tension/compression spring design problem

As shown in Fig. 6, the main goal of this design problem is to minimize the weight of the tension/compression spring. There are three variables, i.e., d (x_1), D (x_2), and P (x_3).

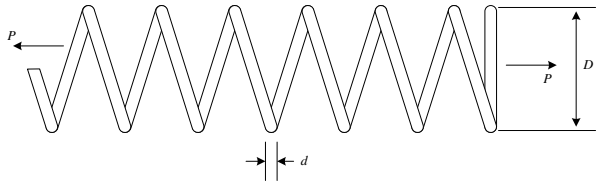


Fig. 6. Tension/compression spring design problem.

The mathematical formulation of tension/compression spring design problem is defined as follows:

$$\textit{Minimize}\quad f(x_1,x_2,x_3)=(x_3+2)x_2x_1^2$$

$$\textit{Subject to}\quad g_1(x_1,x_2,x_3)=1-\frac{x_2^3x_3}{71785x_1^4}\le 0$$

$$g_2(x_1,x_2,x_3)=\frac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\frac{1}{5108x_1^2}\le 0$$

$$g_3(x_1,x_2,x_3)=1-\frac{140.45x_1}{x_2^2x_3}\le 0$$

$$g_4(x_1,x_2,x_3)=\frac{x_1+x_2}{1.5}-1\le 0$$

where $0.05\le x_1\le 2$, $0.25\le x_2\le 1.30$, $2.00\le x_3\le 15$.

The tension/compression spring design problem has already been solved by many optimization approaches. The comparison results of BBOA with other state-of-the-art algorithms are given in Table 7. For the tension/compression spring design problem, as shown in Table 7, the best solution is obtained by the TEO, EEGWO, and CSA. However, the number of function evaluations for the TEO, EEGWO, and CSA are 30,000, 50,000 and 50,000, respectively. Compared with the GA, CPSO, GSA, GWO, MVO, SCA, WOA, MGWO, and IGWO algorithms, BBOA finds better results for the tension/compression spring design problem.

Table 7

Comparison results of different algorithms for tension/compression spring design problem.

Algorithm	x_1 (d)	x_2 (D)	x_3 (P)	f (best)	f (mean)	f (worst)	St.dev	Max_NFEs
GA (Coello and Montes 2002)	10.890522	0.363965	0.051989	0.012681	0.012742	0.012973	5.90E−05	80,000

CPSO (He and Huang 2007)	11.244543	0.357644	0.051728	0.0126747	0.012730	0.012924	5.20E−05	200,000
GSA (Rashedi et al. 2009)	14.22867	0.317312	0.05000	0.0128739	0.0134389	0.0142117	1.34E−02	30,000
GWO (Mirjalili et al. 2014)	12.04249	0.344541	0.051178	0.0126723	0.0126971	0.0127208	2.10E−05	30,000
MVO (Mirjalili et al. 2016)	14.22623	0.315956	0.05000	0.0128169	0.0144644	0.0178397	1.62E−03	30,000
SCA (Mirjalili 2016)	12.72269	0.334779	0.050780	0.0127097	0.0128396	0.0129984	7.80E−05	30,000
WOA (Mirjalili and Lewis 2016)	10.3551	0.374194	0.052406	0.012676	0.012868	0.013072	5.86E−04	30,000
TEO (Kaveh and Dadras 2017)	11.16839	0.358792	0.051775	0.012665	0.012685	0.012715	4.41E−06	300,000
MGWO (Kumar and Kumar 2017)	11.80809	0.348197	0.051334	0.0126696	0.0126799	0.0127057	1.10E−05	30,000
EEGWO (Long et al. 2018a)	11.3113	0.35634	0.051673	0.012665	0.012685	0.012720	2.22E−05	50,000
IGWO (Long et al. 2018b)	11.2756	0.356983	0.051701	0.012667	0.012691	0.012718	1.97E−05	50,000
CSA (Askarzadeh 2016)	11.289012	0.356717	0.051689	0.0126652	0.012666	0.0126702	1.36E−06	50,000
BBOA	11.28603	0.356768	0.051690	0.012666	0.012682	0.012710	3.98E−06	30,000

4.3.2 Pressure vessel design problem

The main objective of the pressure vessel design problem is to minimize the whole cost of the cylindrical pressure vessel (see Fig. 7). There are four variables, i.e., T_s (x_1), T_h (x_2), R (x_3), and L (x_4).

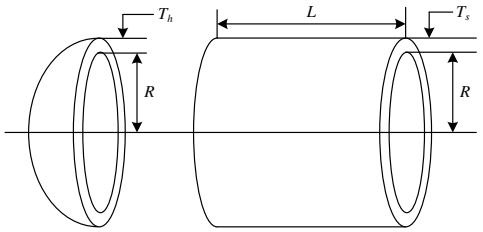


Fig. 7. Pressure vessel design problem.

The mathematical formulation of pressure vessel design problem is defined as follows:

$$\textit{Minimize}\quad f(x_1,x_2,x_3,x_4)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

$$\textit{Subject to}\quad g_1(x_1,x_2,x_3,x_4)=-x_1+0.00193x_3\le 0$$

$$g_2(x_1,x_2,x_3,x_4)=-x_3+0.00954x_3\le 0$$

$$g_3(x_1,x_2,x_3,x_4)=-\pi x_3^2x_4-\frac{4}{3}\pi x_3^3+129600\le 0$$

$$g_4(x_1,x_2,x_3,x_4)=x_4-240\le 0$$

where $0\le x_1,x_2\le 99$, $10\le x_3,x_4\le 200$.

The pressure vessel design problem is solved by several optimization techniques. The comparison results of BBOA with other state-of-the-art algorithms are shown in Table 8. For the compression vessel design problem, the best index in Table 8 shows that the result obtained by BBOA is interior to the CSA, TEO, and HPSO algorithms. Compared with the other algorithms, BBOA provides the better "best" results. For the mean, the worst, and the st.dev indices, the results obtained by BBOA are better than those obtained by all of the selected algorithms. In addition, for the number of function evaluations, TEO has the minimum number of FEs (20,000), while SPGA has a considerable number of FEs (900,000). The number of FEs by BBOA (30,000) is moderate among the algorithms.

Table 8

Comparison results of different algorithms for pressure vessel design problem.

Algorithm	x_1 (T_s)	x_2 (T_h)	x_3 (R)	x_4 (L)	f (best)	f (mean)	f (worst)	St.dev	Max_NFEs
SPGA (Coello 2000)	0.8125	0.4375	40.3239	200.0000	6288.7445	6293.8432	6308.4970	7.4133	900,000
CDE (Becerra and Coello 2006)	0.8125	0.4375	42.0984	176.6377	6059.7340	6085.2303	6371.0455	43.013	204,800
CPSO (He and Huang 2007)	0.8125	0.4375	42.0913	176.7465	6061.0777	6147.1332	6363.8041	86.45	240,000

SCA (Mirjalili 2016)	0.8176	0.4179	41.74939	183.5727	6137.3724	6326.7606	6512.3541	126.609	30,000
CSA (Askarzadeh 2016)	0.8125	0.4375	42.0984	176.6366	6059.7144	6342.4991	7332.8416	384.9454	250,000
TEO (Kaveh and Dadras 2017)	0.8125	0.4375	42.0984	176.6366	6059.71	6138.61	6410.19	129.9033	20,000
EEGWO (Long et al. 2018a)	13.093	6.7922	42.09758	176.6495	6059.8704	6066.7220	6091.0922	10.64121	50,000
IGWO (Long et al. 2018b)	12.853	6.9805	42.09806	176.6416	6059.7659	6059.9066	6060.1246	0.139521	50,000
WOA (Mirjalili and Lewis 2016)	12.970	7.3377	42.03656	177.4064	6067.2991	6205.8012	6463.3448	190.9459	30,000
HPSO (He and Wang 2007)	0.8125	0.4375	42.0984	176.6366	6059.7143	6099.9323	6288.6770	86.20	81.000
BBOA	13.086	6.7918	42.09831	176.6389	6059.7450	6059.8021	6059.8791	5.40E−02	30,000

4.3.3 Three-bar truss design problem

The aim of the three-bar truss design problem (as shown in Fig. 8) is to minimize the volume of a statically loaded three-bar truss and evaluate the optimal cross-sectional areas. There are three design variables.

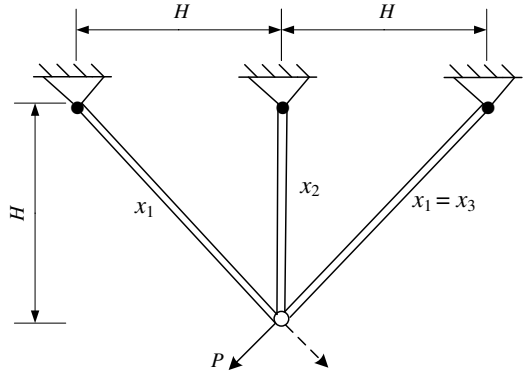


Fig. 8. Three-bar truss design problem.

The mathematical formulation of three-bar truss design problem is given as follows:

$$\textit{Minimize}\quad f(x_1,x_2,x_3)=(2\sqrt{2}x_1+x_2)\times l$$

$$\textit{Subject to}\quad g_1(x_1,x_2)=\frac{\sqrt{2}x_1+x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0$$

$$g_2(x_1,x_2)=\frac{x_2}{\sqrt{2}x_1^2+2x_1x_2}P-\sigma\le 0$$

$$g_3(x_1,x_2)=\frac{1}{x_1+\sqrt{2}x_2}P-\sigma\le 0$$

where $0\le x_1,x_2\le 1$, $l=100\,\text{cm}$, $P=2\,\text{KN/cm}^2$, and $\sigma=2\,\text{KN/cm}^2$.

The optimization techniques in the literature are previously applied to solve the three-bar truss design problem. The comparison results of BBOA with other well-performance optimization algorithms are shown in Table 9. From Table 9, BBOA, GWO, MVO, and MGWO algorithms get similar "best" results for the three-bar truss design problem. In addition, the result obtained by BBOA is better than the GSA, SCA, WOA, and EEGWO.

Table 9

Comparison results of different algorithms for three-bar truss design problem.

Algorithm	x_1	x_2	f(best)	f(mean)	f(worst)	St.dev	Max_NFEs
GSA (Rashedi et al. 2009)	0.777662	0.448853	264.8299	271.0348	279.7925	4.128589	30,000
GWO (Mirjalili et al. 2014)	0.788409	0.409003	263.8959	263.8966	263.8980	4.37E−04	30,000
MVO (Mirjalili et al. 2016)	0.788993	0.407351	263.8959	263.8961	263.8971	2.49E−04	30,000
SCA (Mirjalili 2016)	0.789068	0.407162	263.8984	263.9356	263.9951	2.88E−02	30,000
WOA (Mirjalili and Lewis 2016)	0.792110	0.398620	263.9060	263.9361	264.0214	4.81E−02	30,000
MGWO (Kumar and Kumar 2017)	0.788561	0.407162	263.8959	263.8963	263.8976	4.29E−04	30,000
EEGWO (Long et al. 2018a)	0.788410	0.408990	263.8960	263.8963	263.8966	2.19E−04	50,000
BBOA	0.788566	0.407180	263.8959	263.8960	263.8962	2.12E−04	30,000

4.4 BBOA applied to feature selection problems

To further study the performance of the proposed BBOA, the feature selection (FS) problems are solved. The main purpose of feature selection is to choose the most significant features from the original features to reduce the dimensionality of the datasets. In fact, FS is a typical combinatorial optimization problem which requires lots of computation. In this subsection, BBOA is introduced to solve feature selection problems. However, the solution space of FS is represented by binary values. It must be noted that BBOA is a continuous-space-based optimization method which needs to transform it from continuous version into binary one when solving the FS problems. One of the easiest conversion techniques is to use a transfer function. The advantage of this technique is not to modify the framework of BOA. This paper uses the S-shaped transfer function and is given as follows:

$$T(x) = \frac{1}{1+e^{-\tau x}} \tag{10}$$

where τ is a constant number.

Sixteen well-known benchmark datasets from the UCI data repository (Bache and Lichman 2013) are used to further investigate the effectiveness of BBOA. These datasets are utilized by many researchers or performance comparison in the field of feature selection. The characteristics of these datasets are provided in Table 7.

Table 10

Details of datasets.

Dataset	No. of Attributes	No. of Objects	No. of Classes
BreastEW	30	569	2
Breastcancer	9	699	2
CongressEW	16	435	2
Exactly	13	1000	2
Exactly2	13	1000	2
HeartEW	13	270	2
IonosphereEW	34	351	2
Lymphography	18	148	4
M-of-n	13	1000	2
PenglungEW	325	73	7
SonarEW	60	208	2
SpectEW	22	267	2
Tic-tac-toe	9	958	2
Vote	16	300	2
WineEW	13	178	3
Zoo	16	101	7

We compared the performance of BBOA with other eight population-based algorithms, such as ABC [3], differential evolution (DE) (Storn and Price 1997), gravitational search (GSA) (Rashedi et al. 2009), sine cosine algorithm (SCA) (Mirjalili 2016), salp swarm algorithm (SSA) (Mirjalili et al. 2017), hybrid PSO and grey wolf optimizer (HGWO) (Singh and Singh 2017), conscious crow search algorithm (CCSA) (Zamani et al. 2020), and the basic BOA (Arora and Singh 2019). In all experiments, the population size is 10, and the maximum number of iteration is 100. Each algorithm is conducted 30 independent runs on each dataset to reduce error. The parameter values of the selected eight algorithms are fixed according to their corresponding papers. The average and standard deviation results of nine algorithms are summarized in Table 11. For each dataset, the optimal values are made in boldface. In addition, the average Friedman's test ranking values of nine algorithms on sixteen datasets are also provided in Table 11.

Table 11

Comparison between BBOA and other eight algorithms in term of classification accuracy.

Datasets	Measure	ABC	DE	GSA	SCA	SSA	HGWO	CCSA	BOA	BBOA
BreastEW	Mean	0.9517	0.9522	0.9495	0.9487	0.9518	0.9557	0.9540	0.9502	**0.9637**
	St.dev	0.0033	0.0022	0.0031	0.0031	0.0028	0.0028	0.0029	0.0033	0.0021

Breastcancer	Mean	0.9715	0.9700	0.9687	0.9692	0.9700	0.9708	0.9699	0.9696	**0.9722**
	St.dev	0.0018	0.0017	0.0020	0.0014	0.0021	0.0014	0.0021	0.0016	0.0015
CongressEW	Mean	0.9631	0.9589	0.9529	0.9543	0.9584	**0.9657**	0.9620	0.9566	0.9621
	St.dev	0.0044	0.0029	0.0041	0.0029	0.0032	0.0033	0.0042	0.0036	0.0034
Exactly	Mean	**1.0000**	**1.0000**	0.9970	0.9973	0.9997	**1.0000**	0.8848	0.9959	0.9995
	St.dev	0.0000	0.0000	0.0065	0.0050	0.0010	0.0000	0.1055	0.0066	0.0010
Exactly2	Mean	0.7598	0.7628	0.7568	0.7621	0.7589	0.7648	0.7509	0.7678	**0.7695**
	St.dev	0.0173	0.0138	0.0071	0.0195	0.0127	0.0093	0.0105	0.0178	0.0103
HeartEW	Mean	0.8278	0.8246	0.7965	0.8159	0.8178	0.8237	0.8222	0.8165	**0.8337**
	St.dev	0.0084	0.0076	0.0104	0.0076	0.0090	0.0063	0.0106	0.0058	0.0062
IonosphereEW	Mean	0.9261	0.9194	0.9105	0.9121	0.9151	0.9286	0.9197	0.9137	**0.9311**
	St.dev	0.0083	0.0048	0.0081	0.0053	0.0057	0.0048	0.0066	0.0032	0.0071
Lymphography	Mean	0.8574	0.8568	0.8355	0.8497	0.8486	0.8568	0.8470	0.8544	**0.8586**
	St.dev	0.0105	0.0075	0.0110	0.0090	0.0064	0.0092	0.0110	0.0081	0.0107
M-of-n	Mean	**1.0000**	**1.0000**	0.9975	0.9988	0.9995	**1.0000**	0.9410	0.9997	0.9998
	St.dev	0.0000	0.0000	0.0056	0.0020	0.0013	0.0000	0.0587	0.0008	0.0011
PenglungEW	Mean	0.9062	0.9144	0.9041	0.9171	0.9116	0.9171	0.9130	0.9158	**0.9181**
	St.dev	0.0102	0.0087	0.0089	0.0054	0.0083	0.0070	0.0067	0.0067	0.0048
SonarEW	Mean	0.9041	**0.9147**	0.8940	0.9103	0.9099	0.9135	0.9053	0.9108	0.9121
	St.dev	0.0107	0.0085	0.0099	0.0074	0.0084	0.0080	0.0113	0.0055	0.0083
SpectEW	Mean	0.8333	0.8354	0.8185	0.8219	0.8333	0.8416	0.8373	0.8260	**0.8425**
	St.dev	0.0094	0.0069	0.0072	0.0055	0.0061	0.0070	0.0094	0.0048	0.0063
Tic-tac-toe	Mean	**0.8435**	0.8354	0.8118	0.8121	0.8308	0.8341	0.8122	0.8143	0.8426
	St.dev	0.0044	0.0041	0.0031	0.0033	0.0041	0.0045	0.0166	0.0024	0.0031
Vote	Mean	0.9537	0.9493	0.9492	0.9505	0.9483	**0.9578**	0.9558	0.9522	0.9568
	St.dev	0.0037	0.0038	0.0039	0.0036	0.0047	0.0039	0.0044	0.0033	0.0045
WineEW	Mean	0.9820	0.9798	0.9725	0.9826	0.9775	0.9823	0.9803	0.9809	**0.9842**
	St.dev	0.0039	0.0038	0.0065	0.0040	0.0032	0.0027	0.0050	0.0038	0.0017
Zoo	Mean	0.9837	0.9891	0.9817	**0.9921**	0.9881	0.9851	0.9837	0.9916	0.9896
	St.dev	0.0048	0.0030	0.0048	0.0041	0.0052	0.0051	0.0048	0.0036	0.0008
Average ranking		4.06	4.25	8.38	6.09	6.06	2.69	6.03	5.50	1.94
Total ranking		3	4	9	8	7	2	6	5	1

From Table 11, BBOA outperforms ABC on 13 datasets. However, ABC obtains the better results on 4, 9, and 13 datasets. Compared with DE algorithm, the BBOA achieves the better and worst results on 13 and three datasets (4, 9, and 11), respectively. In particular, the BBOA performs better than SCA and SSA on all of the datasets except for "Zoo" and "Exactly" datasets, respectively. With respect to the GSA, CCSA, and BOA, the BBOA provides the better results on all of the datasets. Additionally, the BBOA performs better than HGWO on 12 datasets. However, the HGWO achieves the better results on four datasets. According to the Friedman ranking test results, the BBOA ranked the first, followed by HGWO, ABC, DE, BOA, CCSA, SSA, SCA, and GSA.

5 Conclusions

In this paper, a new variant of BOA, called BBOA, has been proposed. The proposed BBOA introduces two different operators, i.e., the modified position-updating equation strategy based on the dynamic inertia weight, and the opposition-based learning strategy. These two operators can balance the exploration and exploitation of BOA. The forty benchmark test functions, three engineering design, and sixteen feature selection problems are selected to investigate the performance of the proposed BBOA and compare with the state-of-the-art algorithms. The experimental results show that the proposed BBOA algorithm has better precision and faster convergence. In the future, it is interesting to generalize BBOA for solving constrained single objective, multi-objective, and practice optimization problems.

Acknowledgement

This work was supported by the Science and Technology Foundation of Guizhou Province ([2020]1Y012), Natural Science Research Projects of Education Department of Guizhou Province (KY[2021]015), Guizhou Key Laboratory of Big Data Statistics Analysis (BDSA20200101, BDSA20190106), Natural Science Foundation in the

Hunan Province of China (2020JJ4382), and Key Projects of Education Department of Hunan Province (19A254).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Human and animals participants All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Declaration of Helsinki and its later amendments or comparable ethical standards. The article does not contain any studies with animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Ahandani MA, Alavi-Rad H (2015) Opposition-based learning in shuffled frog leaping: An application for parameter identification. Inform Sci 291:19−42

Arora S, Anand P (2019) Binary butterfly optimization approaches for feature selection. Expert Syst Appl 116:147−160

Arora S, Anand P (2018) Learning automata based butterfly optimization algorithm for engineering design problems. Int J Comput Mater Sci Eng 7(4):1−15

Arora S, Singh S (2017) An improved butterfly optimization algorithm with chaos. J Intell Fuzzy Syst 32:1079−1088

Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23(3): 715−734

Arora S, Singh S, Yetilmezsoy K (2018) A modified butterfly optimization algorithm for mechanical design optimization problems. J Braz Soc Mech Sci Eng 40:1−21

Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Comput Struct 169:1−12

Bache K, Lichman M (2013) UCI machine learning repository.

Becerra R, Coello CAC (2006) Cultured differential evolution for constrained optimization. Comput Methods Appl Mech Eng 195:4303−4322

Blum C (2005) Ant colony optimization: introduction and recent trends. Phys Life Rev 2:353−373

Chatterjee A, Siarry P (2006) Nonlinear inertia weight variation for dynamic adaption in particle swarm optimization. Comput Oper Res 33:859−871

Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41: 113−127

Coello CAC, Mezura-Montes E (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inf 16:193−203

Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186: 311−338

Dinkar SK, Deep K (2018) An efficient opposition based Levy flight antlion optimizer for optimization problems. J Comput Sci 29:119−141

Ewees AA, Elaziz MA, Houssein EH (2018) Improved grasshopper optimization algorithm using opposition-based learning. Expert Syst Appl 112:156−172

Goldberg D (1989) Genetic algorithm in search optimization and machine learning. Addison-Wesley, New York

Gupta S, Deep K (2019a) A hybrid self-adaptive sine cosine algorithm with opposition based learning. Expert Syst Appl 119: 210−230

Gupta S, Deep K (2019b) A novel random walk grey wolf optimizer. Swarm Evol Comput 44:101−112

Gupta S, Deep K (2019c) Improved sine cosine algorithm with crossover scheme for global optimization. Knowl-Based Syst 165:374−406

Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9:159−195

He Q, Hang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20:89−99

He Q, Wang L (2007) A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 186:1407−1422

Huang L, Ding S, Yu S, Wang J, Lu K (2016) Chaos-enhanced cuckoo search optimization algorithms for global optimization. Appl Math Model 40:3860−3875

Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687−697

Kaveh A, Dadras A (2017) A novel meta-heuristic optimization algorithm: Thermal exchange optimization. Adv Eng Softw 110:69−84

Kennedy V, Eberhart R (1995) Particle swarm optimization. in: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942−1948

Kumar V, Kumar D (2017) An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems. Adv Eng Softw 112:231−254

Liang JJ, Qin A, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10:281−295

Liang JJ, Qu B, Suganthan PN (2013) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore

Long W, Cai S, Jiao J, Xu M, Wu T (2020a) A new hybrid algorithm based on grey wolf optimizer and cuckoo search for parameter extraction of solar photovoltaic models. Energy Convers Manag 203:1−14

Long W, Jiao J, Liang X, Tang M (2018a) An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. Eng Appl Artif Intell 68:63−80

Long W, Jiao J, Liang X, Tang M (2018b) Inspired grey wolf optimizer for solving large-scale function optimization problems. Appl Math Model 60:112−126

Long W, Wu T, Jiao J, Tang M, Xu M (2020b) Refraction-learning-based whale optimization algorithm for high-dimensional problems and parameter estimation of PV model. Eng Appl Artif Intell 89:1−14

Long W, Wu T, Cai S, Liang X, Jiao J, Xu M (2019a) A novel grey wolf optimizer algorithm with refraction learning. IEEE Access 7:57805−57819

Long W, Wu T, Liang X, Xu S (2019b) Solving high-dimensional global optimization problems using an improved sine cosine algorithm. Expert Syst Appl 123:108−126

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

Mirjalili S, Lewis A (2016) The whale optimization algorithm, Adv. Eng. Softw. 95:51−67

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

Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46−61

Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120−133

Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12: 64−79

Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inform Sci 179:2232−2248

Sharma S, Saha AK (2020) m-MBOA: a novel butterfly optimization algorithm enhanced with mutualism scheme. Soft Comput 24:4809−4827

Shi Y, Eberhart R (1998) A modified particle swarm optimizer. in: IEEE International Conference on Evolutionary Computation, pp. 69−73

Singh N, and Singh SB (2017) Hybrid algorithm of particle swarm optimization and grey wolf optimizer for improving convergence performance. J Appl Math 2017:1−15

Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341−359

Tizhoosh HR (2005) Opposition-based learning: A new scheme for machine intelligence. in: Proceedings of International Conference on Computation Intelligence Modeling Control Automation, pp. 695−701

Wang H, Wu Z, Rahnamayan S, Liu Y, Ventresca M (2011) Enhancing particle swarm optimization using generalized opposition-based learning. Inform Sci 181:4699−4714

Yang XS (2009) Firefly algorithm for multimodal optimization. in: Proceedings of the International Conference on Stochastic Algorithms, pp. 169−178

Yang XS (2010) A new metaheuristic bat-inspired algorithm. in: Proceedings of the Nature Inspired Cooperative Strategies for

Optimization, pp. 65−74

Yuan Z, Wang W, Wang H, Hosse K (2020) Improved butterfly optimization algorithm for CCHP driven by PEMFC. Appl Thermal Eng 173:1−10

Zamani H, Nadimi-Shahraki MH, Gandomi AH (2020) CCSA: Conscious neighborhood-based crow search algorithm for solving global optimization problems. Appl Soft Comput 85:105583

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13:945−958