Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 3498363, 25 pages
https://doi.org/10.1155/2017/3498363

Research Article

Elite Opposition-Based Water Wave Optimization Algorithm for Global Optimization

Xiuli Wu,[1] Yongquan Zhou,[2,3] and Yuting Lu[1]

[1]*School of Computer and Electronics Information, Guangxi University, Nanning 530004, China*
[2]*College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China*
[3]*Key Laboratory of Guangxi High Schools Complex System and Computational Intelligence, Nanning 530006, China*

Correspondence should be addressed to Yongquan Zhou; yongquanzhou@126.com

Received 13 August 2016; Revised 4 November 2016; Accepted 15 November 2016; Published 15 January 2017

Academic Editor: Manuel Doblaré

Water wave optimization (WWO) is a novel metaheuristic method that is based on shallow water wave theory, which has simple structure, easy realization, and good performance even with a small population. To improve the convergence speed and calculation precision even further, this paper on elite opposition-based strategy water wave optimization (EOBWWO) is proposed, and it has been applied for function optimization and structure engineering design problems. There are three major optimization strategies in the improvement: elite opposition-based (EOB) learning strategy enhances the diversity of population, local neighborhood search strategy is introduced to enhance local search in breaking operation, and improved propagation operator provides the improved algorithm with a better balance between exploration and exploitation. EOBWWO algorithm is verified by using 20 benchmark functions and two structure engineering design problems and the performance of EOBWWO is compared against those of the state-of-the-art algorithms. Experimental results show that the proposed algorithm has faster convergence speed, higher calculation precision, with the exact solution being even obtained on some benchmark functions, and a higher degree of stability than other comparative algorithms.

1. Introduction

Optimization problem is wide and varied; in many scientific and engineering computation areas, a majority of problems of people encounter can be attributed to objective optimization problem; thus, the research of optimization problem has been a very active field. In fact, the optimization method can be divided into two types of deterministic optimization and stochastic optimization; although the deterministic optimization method is relatively mature, its application condition is harsh and difficult to deal with large-scale optimization problems, which prompted the stochastic optimization method, especially development of heuristic optimization method.

In recent years, the heuristic optimization algorithm, especially the metaheuristic optimization algorithm, has been concerned by many researchers. Metaheuristic optimization algorithm originates from the simulation of various types of physical, biological, social, and other phenomena in nature to solve optimization problems. As a stochastic optimization method, the metaheuristic optimization algorithm has the advantage of simple and universal, strong robustness, suitable for parallel processing and wide application range. Due to the advantages of metaheuristic, several algorithms have been proposed recently, such as particle swarm optimization (PSO) [1], genetic algorithm (GA) [2], ant colony optimization (ACO) [3], artificial bee colony (ABC) [4], cuckoo search (CS) [5], bat algorithm (BA) [6], firefly algorithm (FA) [7], flower pollination algorithm (FPA) [8], and water wave optimization (WWO) [9].

Water wave optimization (WWO) is a relatively new metaheuristic initially proposed by Zheng in 2015 [9], inspired by the shallow water wave theory [10] for global optimization. WWO has the advantages of simple framework and thus easiness of implementation; even with a small population size it performs well [9]. At present, as a new metaheuristic optimization method, WWO has been successfully applied to the optimization problems such as high speed [9] and TSP [11].

In order to further improve the performance of WWO, some modified approaches are introduced to strengthen its performance. Zhang et al. [12] improved on WWO; an improved version with variable population size (VC-WWO) is proposed by them, and, meanwhile, a comprehensive learning mechanism is developed in refraction operator to increase the solution diversity. Zheng and Zhang [13] developed a simplified version of WWO (Sim-WWO); in Sim-WWO, leaving out the refraction operator and in order to better balance exploration and exploitation as well as partially compensate the effect of weeding out refraction operator, a strategy of population size reducing is introduced. In order to apply WWO to combinatorial optimization problem, the traveling salesman problem (TSP), Wu et al. [11] redefined the propagation, breaking, and refraction operator based on the original WWO. In this paper, an improved water wave optimization algorithm based on elite-opposition (EOBWWO) learning strategy has been applied to function optimization and structure engineering design problems. The improvements include three parts: elite opposition-based learning (EOBL) strategy enhances the diversity of population, local neighborhood search strategy is introduced to enhance local search in breaking operation, and improved propagation operator provides the improved algorithm with a better balance between exploration and exploitation. We tested the performance of EOBWWO on 20 benchmark functions and two structure engineering design problems. The experimental results show that proposed algorithm has signification performance advantage including a fast convergence speed and a high calculation precision; in addition, the improved algorithm is able to obtain the exact solution on some test functions.

This paper is organized into the following sections. Section 2 introduces the original WWO algorithm briefly, the detailed description of EOBWWO algorithm is presented in Section 3, simulation experiments and results discussed are described in Section 4, and finally the conclusion is given in Section 5.

2. Water Wave Optimization (WWO) Algorithm

Water wave optimization (WWO) algorithm is inspired by shallow water wave theory and developed by Zheng [9], where each individual in the population is analogous to the "water wave" object with a wave height h and a wavelength λ. Without losing generality, suppose there is a maximization problem F and its objection function is f, where practical problem F can be compared with the shallow water wave model; the corresponding relation is shown as follows:

$$\begin{aligned} &\text{Practical problem } F && \text{Shallow water wave model} \\ &\text{The search space of } F \xrightarrow{\text{analogous to}} && \text{Seabed area} \\ &\text{Each solution of } F \xrightarrow{\text{analogous to}} && \text{A water wave objection} \\ &\text{The fitness of each solution} \longrightarrow && \text{It is inversely proportional to vertical distance to the seabed.} \end{aligned} \tag{1}$$

When the population is initialized, for each wave, the wave height h is set to a constant $h_{\max}$ and wavelength λ is generally set to 0.5. The fitness value of each water wave is inversely proportional to the vertical distance to the seabed; from this we can know that from the seabed nearer the water wave fitness value is bigger, the wave height is bigger, and the wavelength is smaller, as illustrated in Figure 1. During the process of optimization problem-solving, search globally in the solution space by simulating the propagation, breaking, and refraction operation of water waves.

2.1. Propagation. In WWO, all water waves have to be propagated once at each generation. It is assumed that the original water wave is x, x' is a new wave created by propagation operator, the dimension of the maximum value function F is D, the propagation operation is shifted, and each dimension of the original water wave x is given as

$$x'(d) = x(d) + \text{rand}(-1, 1) \cdot \lambda L(d), \tag{2}$$

where $d \in D$, rand$(-1, 1)$ is used to control the propagation step which is a uniformly distributed random number fixed in $[-1, 1]$, and $L(d)$ is the length of the dth dimension of the search space. If the length of $L(d)$ is longer than the length of the dth dimension of the search space, a new position will be reset randomly as

$$L(d) = lb(d) + \text{rand}() * (ub(d) - lb(d)), \tag{3}$$

where $lb(d)$ and $ub(d)$ are the lower bound and upper bound of dth dimension of the search space and rand( ) is a random number within the range $[0, 1]$.

After propagating, we evaluate fitness of x'; if $f(x') > f(x)$, x' instead of x in the population, meanwhile the wave height of x' is reset to $h_{\max}$; otherwise, x remained, and in order to simulate energy dissipation of wave in the process of propagation, its height is decreased by one.

It is a natural phenomenon that when a wave travels from deep water to shallow water, its wave height increases and its wavelength decreases, as illustrated in Figure 1. In a bid to simulate this phenomenon, WWO uses the way in which the

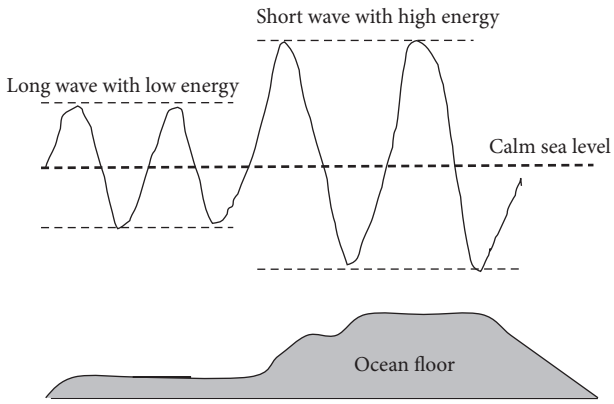

Figure 1: Different wave shapes in deep and shallow water.

wavelength of each wave is updated after each generation as follows:

$$\lambda = \lambda * \alpha^{-(f(x)-f_{\min}+\varepsilon)/(f_{\max}-f_{\min}+\varepsilon)}, \quad (4)$$

where α is a control parameter named wavelength reduction coefficient, $f_{\max}$ and $f_{\min}$ are the maximum and minimum fitness values among the current population, respectively, and ε is a very small positive constant to avoid division-by-zero.

2.2. Breaking. In the water wave theory, with the energy of water wave increasing constantly, crest becomes more and more steep, and the wave breaks into a series of solitary waves when its velocity of crest exceeds the wave celerity. After propagating, WWO only performs breaking on the wave x which is a new best solution x^*, which is used to improve the diversity of the population. The detailed process is as follows: first of all, we select randomly k dimensions (where k is a random number between 1 and a predefined number $k_{\max}$) and perform operations on each selected dimension of original wave x to generate each dimension of solitary wave x' as follows:

$$x'(d) = x(d) + N(0,1) \cdot \beta L(d), \quad (5)$$

where $N(0,1)$ is a Gaussian random number with mean 0 and standard deviation 1 and β is breaking coefficient. If the solitary wave with the best fitness is better than x^*, x' is selected instead of x^*; otherwise, x^* remained.

2.3. Refraction. In WWO, the refraction operation only performs on a wave x whose height decreases to zero to avoid search stagnation, which simulates the phenomenon that wave ray is not perpendicular to the isobath. By refraction, in the way that random number centered halfway between the original positions and x^* to calculate each dimension of new wave x', the details are as follows:

$$x'(d) = N\left(\frac{(x^*(d) + x(d))}{2}, \frac{|x^*(d) - x(d)|}{2}\right). \quad (6)$$

Followed by refraction, the wave height of x' is also reset to $h_{\max}$; meanwhile its wavelength is updated as follows:

$$\lambda = \lambda \frac{f(x)}{f(x')}. \quad (7)$$

To sum up, the role of propagation operator is to make the high fitness wave exploit small area and the low fitness wave explore large area, the breaking operator enhances the local search among the promising best waves, and the refraction operation helps avoid search stagnation and thus reduces premature convergence. The basic framework of WWO is as Algorithm 1 [9].

3. The EOBWWO Algorithm

In order to improve the performance including global searching and local searching abilities of WWO and obtain a better balance between exploration and exploitation even further, there are three optimization strategies applied to the original WWO; they are elite opposition-based learning (EOBL) strategy [42], local neighborhood searching (LNS) strategy [43], and improved propagation operator.

3.1. Elite Opposition-Based Learning (EOBL) Strategy. The optimization process of WWO algorithm can be regarded as the transformation continually of its search space. When the algorithm falls into local optimum, the search space is difficult to contain the global optimal solution. Thus it is very significant to guide the current solution space approximation to the space of global optimal solution. In a bid to enhance the global search ability (i.e., exploration ability) of WWO, the elite opposition-based learning (EOBL) strategy is introduced.

Before introducing the EOBL, we should firstly explain opposition-based learning (OBL) [44]. The main idea of OBL is that it generates the opposition solution of current solution, evaluates current solution and opposition solution at the same time, and chooses the better one to enter the next iteration. We assume $x = (x_1, x_2, \ldots, x_D)$ is a point in current population (D is the dimension of search space; $x_j \in [a_j, b_j]$, $j = 1, 2, \ldots, D$), and its opposition point $\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_D)$ is defined as follows:

$$\tilde{x}_j = a_j + b_j - x_j. \quad (8)$$

Since the opposition solution of OBL generated may not be more favorable than the current search space to search the global optimal solution, thus in this paper, we can use elite opposition-based learning (EOBL) strategy. EOBL is a new technique in the field of intelligence computation and its model can be described as follows: Suppose the elite individual (optimal individual in population) in the current population is $X_e = (x_{e,1}, x_{e,2}, \ldots, x_{e,D})$, for an individual $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$, the elite opposition solution $\widetilde{X}_i = (\tilde{x}_{i,1}, \tilde{x}_{i,2}, \ldots, \tilde{x}_{i,D})$ of X_i is defined as follows:

$$\tilde{x}_{i,j} = \eta * \left(da_j + db_j\right) - x_{e,j}, \quad (9)$$

Input: Define objective function $f(x)$, $x = (x_1, x_2, \ldots, x_D)$
Output: The best solution x^*;
(1) *Initialization*: Initialize parameters including α, β, λ, randomly initialize a population P of NP waves;
(2) while stop criterion is not satified do
(3) for each $x \in P$ do
(4) Propagate x to a new x' based on equation (2);
(5) if $f(x') < f(x)$ then
(6) if $f(x') < f(x^*)$ then
(7) Break x based on equation (5);
(8) Update x^* with x';
(9) endif
(10) Replace x with x';
(11) else
(12) Decrease $x.h$ by one;
(13) if $x.h = 0$ then
(14) Refract x to a new x' based on equation (6) and equation (7);
(15) endif
(16) endif
(17) Update the wavelength based on equation (4);
(18) endfor
(19) endwhile

Algorithm 1: The WWO algorithm.

where $i = 1, 2, \ldots, NP$, NP is the population size, $j = 1, 2, \ldots, D$, $\eta \in U(0, 1)$ and η is a generalized coefficient, and $[da_j, db_j]$ is the dynamic boundary of the jth dimensional search space and can be obtained by the following formula:

$$da_j = \min\left(x_{i,j}\right), \quad db_j = \max\left(x_{i,j}\right). \tag{10}$$

The fixed boundary is not conducive to preserve the search experience; thus we use dynamic boundary of the search space to replace the fixed boundary to preserve the search experience in order to make the opposition solution located in the search space which is narrowing. Moreover, if the operator of dynamic boundary makes $\widetilde{x}_{i,j}$ jump out of $[da_j, db_j]$, the following method can be used to reset $\widetilde{x}_{i,j}$:

$$\widetilde{x}_{i,j} = \text{rand}\left(da_j, db_j\right). \tag{11}$$

The EOBL generates the opposition population according to the elite individual and evaluates the current population and the elite population at the same time; in addition, it makes full use of the characteristics of the elite individuals to contain more useful search information than the ordinary individuals which improve the diversity of the population to certain extent. EOBL can enhance the ability of global exploration of WWO.

3.2. Local Neighborhood Search (LNS). WWO only performs the breaking operator on the new best solution to enhance local search around the best solution. In a bid to further enhance the local search ability to improve the convergence speed, the local neighborhood search [43] (LNS) model is added before the breaking operation.

The main idea of LNS is using the best solution found so far in a small neighborhood of the current solution rather than the entire population to update the current solution. The experience of an individual's neighborhood is considered when updating the individual's location, so that the graph of interconnections of them is called neighborhood structure. Suppose there is a WWO population $X = (X_1, X_2, \ldots, X_{NP})$, X_i ($i \in [1, NP]$) is a vector in the current population, and its dimension is D. The indices of each vector are random in a bid to maintain the diversity of each neighborhood. Next, we can define the neighborhood of radius r (r is a nonzero integer and $2r + 1 < NP$), where $r = 10$, for each vector X_i; that is to say, neighborhood of X_i consists of $X_{i-k}, \ldots, X_i, \ldots, X_{i+k}$. For analysis, we suppose that the vectors can be arranged into a ring topology according to their indices. Figure 2 illustrates the concept of local neighborhood model. In addition, the neighborhood topology is static and about the definition of the set of vectors all the time. LNS model is described in

$$L_i = X_i + m * \left(X_{n\_\text{opt}} - X_i\right) + n * \left(X_p - X_q\right), \tag{12}$$

where X_{n_opt} is the best vector in the neighborhood of X_i and $p, q \in [i - r, i + r]$ ($p \neq q \neq i$) and m and n are the scaling factors, where $m, n \in \text{rand}()$. In the improved version of WWO, the new best solution is updated according to (12), and the breaking operation is performed by the updated solution as

$$x'(d) = L_i(d) + N(0, 1) \cdot \beta L(d), \tag{13}$$

where L_i is the best solution updated by LNS.

3.3. Improvement of WWO. In original WWO, all water waves have to be propagated once at each generation, and

Input: Define objective function $f(x)$, $x = (x_1, x_2, \ldots, x_D)$
Output: The best solution x^*;
(1) *Initialization*: Initialize related parameters including α, β, λ, η, $w_{\max}$, $w_{\min}$, a, b, m and n, initialize dynamic boundary of the search space, randomly initialize a population P of NP waves
(2) while stop criterion is not satisfied do
(3) *Update the current population with EOBL accordding to equation (9), equation (10) and equation (11);*
(4) for each $x \in P$ do
(5) *Propagate x to a new x' based on equation (14);*
(6) if $f(x') < f(x)$ then
(7) if $f(x') < f(x^*)$ then
(8) *Break x based on equation (12) and equation (13);*
(9) Update x^* with x';
(10) Endif
(11) Replace x with x';
(12) else
(13) Decrease $x.h$ by one;
(14) if $x.h = 0$ then
(15) Refract x to a new x' based on equation (6) and equation (7);
(16) endif
(17) endif
(18) Update the wavelength based on equation (4);
(19) endfor
(20) endwhile

Algorithm 2: The framework of EOBWWO algorithm.

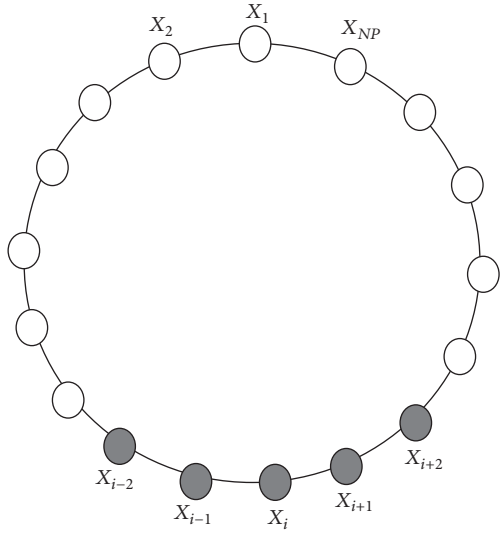

Figure 2: Neighborhood ring topology of radius 2.

the search behavior of each water wave is affected by the other waves in the group. Similar to PSO [1, 45, 46], an inertial weight is embedded into (2) in order to learn the past experience. Moreover, as shown in (2) and (4), propagation operator can make high fitness wave search small region and low fitness wave explore large region in global search process. In (2), the search step size is a random number fixed in the range $[-1, 1]$, which is not very reasonable because search step size prefers fairly large step at the beginning in order to strengthen the probability of reaching the optimal regions, and, with the iteration going on, search step size should be decreased gradually to enhance the local exploitation ability. The random step size in the propagation process is improved by referencing to the method of [47]; the improved propagation operator is as follows:

$$w = w_{\max} - (w_{\max} - w_{\min}) \frac{\text{iter}}{\text{iter}_{\max}},$$
$$\rho = \frac{1}{1 + e^{a*\text{iter}-b}} \text{rand}(-1, 1), \tag{14}$$
$$x'(d) = w * x(d) + \rho * \lambda L(d),$$

where $w_{\max}$ and $w_{\min}$ are, respectively, the maximum and minimum inertial weight, where $w \in [0.4, 1.5]$, iter is the current iteration number and $\text{iter}_{\max}$ is the maximum iteration number, and a and b are two selected constants, where $a = 0.02$, $b = 25$. Improved propagation operator not only makes use of the past experience, but also makes the search step size decrease gradually with the iteration going on. The whole pseudocode of EOBWWO can be summarized as Algorithm 2.

4. Simulation Experiments and Results Analysis

In order to identify the effectiveness and efficiency of EOBWWO, 20 standard test functions are applied in this section. The detailed parameters about 20 benchmark functions [48, 49] including functional form, scope, optimal solution, and the iterations are illustrated in Table 1. The 20 benchmark

functions can be divided into three groups, unimodal functions (f_{01}~f_{06}) as group 1, multimodal functions (f_{07}~f_{14}) as group 2, and low-dimension functions (f_{15}~f_{20}) as group 3. In the unimodal functions, the global optimum of f_{06} is located in a smooth, long, and narrow parabolic valley; when the traditional gradient optimization method is searched to the valley edge, it is difficult to carry out global optimization. However, it is very slow to change the value in the long and narrow area, which can be used to evaluate the performance of the algorithm. In the multimodal functions, f_{09} has many local minima, it is a typical nonlinear multimodal function, which has a wide range of search space, and it is generally considered to be a complex multimodal problem which is difficult to deal with. In general, unimodal functions are suitable for evaluating the exploitation; however, multimodal functions tend to be a good choice for evaluating exploration [50].

The rest of this section is organized as follows: experimental setting is given in Section 4.1, experiment results of 30 dimensions and discussion are represented in Section 4.2, high-dimension test results including 100 dimensions, 1000 dimensions, and 10000 dimensions for some unimodal functions and multimodal functions are described in Section 4.3, and two design problems are shown in Section 4.4.

4.1. Experimental Setting and Comparative Methods. The empirical analysis was conducted on a computer of Intel(R) with 3.5 GHz Xeon CPU and 8 GB of memory, the operating system is Windows 7, and the programs are written in Matlab 2012a.

The scope and dimension of variables have significant influence on the complexity of optimization. The scope of the benchmark function and the dimension of the low-dimension functions are illustrated in Table 1. The dimensions of unimodal functions and multimodal functions are, respectively, 30, 100, 1000, and 10000.

The performance of proposed EOBWWO algorithm is evaluated by comparing it to five state-of-the-art metaheuristic algorithms: ABC [4], CS [5], FPA [6], BA [8], and WWO [9]; the parameters settings of aforementioned algorithms are given in Table 2.

4.2. Experiment Results and Discussion. In Tables 3-4, the dimension is 30, whereas the standard benchmark functions are listed in Table 1. In this paper, all function optimization experimental results of the algorithms are repeated 30 times to ensure the credibility in statistics. There are four evaluation indicators: max, min, median, and Std represent the worst fitness value, optimal fitness value, median of the test results, and standard deviation, respectively. The last column of Tables 3–5 gives the rank of the algorithms in terms of median values among the six algorithms. The minimum value, best median value, and the minimum standard deviation values among the six algorithms of each benchmark function are shown in bold.

Moreover, nonparametric Wilcoxon rank tests were conducted on the results of EOBWWO and other comparative algorithms on the 20 benchmark functions, and the test results are shown in Table 6, where the value of h is 1 indicating that the performance and comparative method are statistically different with 95% confidence, and 0 implies that there is no statistical difference [9].

In Table 3, on the unimodal group, EOBWWO obtained the exact solution except function f_{06} and obtained the minimum standard deviation. Although ranking fifth on function f_{06}, standard deviation of EOBWWO is less than the other algorithms. All these mean that EOBWWO has a higher calculation precision and better stability in the optimization of the unimodal functions.

On group 2 of 8 multimodal functions, seeing from Table 4, EOBWWO can find the exact solution for f_{07}, f_{09}, f_{10}, f_{11}, f_{12}, and f_{13}, and the standard deviations of the five functions of EOBWWO are zeros. In addition, EOBWWO obtains the best median value on all functions. For functions f_{08} and f_{14}, the worst fitness value, best fitness value, median value, and standard deviations of EOBWWO are less than the other five algorithms. The multimodal functions are more complex than unimodal functions due to the local minima; thus the above analysis indicates that EOBWWO has a strong global search ability and higher calculation precision.

On group 3 of 6 low-dimension functions, results are illustrated in Table 5. For f_{15}, EOBWWO obtains the minimum values including the worst fitness, best value fitness, median value, and standard deviations among the comparative algorithms. For f_{16}, ABC, CS, WWO, and EOBWWO can find the exact solution, and the standard deviations of ABC and EOBWWO are zeros. For f_{17}, ABC, CS, WWO, and EOBWWO have the same median value and optimal fitness value, while the standard deviation of CS is minimal. For f_{18}, FPA can obtain the better fitness value and median, but the standard deviation of WWO is the best. For f_{19}, optimal fitness value, median value, and the standard deviation of FPA are better than those of EOBWWO. And for f_{20}, it is obvious that ABC, CS, WWO, and EOBWWO obtain the exact solution and the standard deviation of CS is better. Through the above analysis of Table 5, we can draw a conclusion that EOBWWO has certain advantages in dealing with low-dimension functions according to the experimental results.

In summary, as a result of introducing the three major optimization strategies in the improvement, the calculation precision of EOBWWO is better than the comparative algorithms for most benchmark functions. In addition to functions f_{06}, f_{18}, and f_{19}, the calculation precision of EOBWWO is inferior to ABC, FPA, and FPA, respectively. Moreover, EOBWWO can find the exact solution on f_{01}–f_{05}, f_{07}, f_{09}, f_{11}, f_{12}, f_{13}, and f_{16}, and the standard deviations of these functions are zeros, which show the higher calculation precision and stronger stability of EOBWWO. EOBWWO obtains exact solution on multimodal functions f_{07}, f_{09}, f_{11}, f_{12}, and f_{13}, which shows that EOBWWO has better global search performance.

In order to show the performance of the EOBNWWO clearly, Figures 3–22 represent the convergence curves and Figures 23–42 describe the ANOVA test of global minimum of benchmark functions in Table 1. From Figures 3–22, obliviously, the convergence rate of EOBWWO is faster than other comparison algorithms including WWO on f_{01}–f_{07},

Table 1: Benchmark test functions.

Name	Functions	Scope	Best	Iterations
Sphere	$f_{01}(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	0	1500
Schwefel's problem 1.2	$f_{02}(x) = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$	$[-100, 100]$	0	1200
Schwefel's problem 2.21	$f_{03}(x) = \max\{\lvert x_i \rvert, \ 1 \le i \le n\}$	$[-100, 100]$	0	6000
Schwefel's problem 2.22	$f_{04}(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-100, 100]$	0	5000
Step	$f_{05}(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]$	0	1000
Rosenbrock	$f_{06}(x) = \sum_{i=1}^{n-1} \left[(x_i - 1)^2 + 100 \left(x_i^2 - x_{i+1} \right)^2 \right]$	$[-100, 100]$	0	1000
Rastrigin	$f_{07}(x) = \sum_{i=1}^{n} \left[x_i^2 - 10 \cos(2\pi x_i) + 10 \right]$	$[-5.12, 5.12]$	0	1000
Ackley	$f_{08}(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos 2\pi x_i \right) \right) + 20 + e$	$[-100, 100]$	0	1000
Griewank	$f_{09}(x) = \frac{1}{4000} \sum_{i=1}^{n} \left(x_i^2 \right) - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	$[-100, 100]$	0	1500
Zakharov	$f_{10}(x) = \sum_{i=1}^{n} x_i^2 + \left(\frac{1}{2} \sum_{i=1}^{n} i x_i \right)^2 + \left(\frac{1}{2} \sum_{i=1}^{n} i x_i \right)^4$	$[-100, 100]$	0	1500
Modified Drop Wave	$f_{11}(x) = -\frac{1 + \cos\left(\sqrt{\sum_{i=1}^{n} x_i^2} \right)}{(1/2)\left(\sum_{i=1}^{n} x_i^2 \right) + 2}$	$[-5.12, 5.12]$	-1	1500
Weierstrass	$f_{12}(x) = \sum_{i=1}^{n} \left\{ \sum_{k=0}^{K} \left[a^k \cos\left(2\pi b^k (x_i + 0.5) \right) \right] \right\} - n \sum_{k=0}^{K} \left[a^k \cos\left(2\pi b^k \cdot 0.5 \right) \right]$, $a = 0.5, \ b = 3, \ K = 20$	$[-0.5, 0.5]$	0	1000
Alpine	$f_{13}(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	$[-10, 10]$	0	5000
Salomon	$f_{14}(x) = -\cos\left(2\pi \sqrt{\sum_{i=1}^{D} x_i^2} \right) + 0.1 \sqrt{\sum_{i=1}^{D} x_i^2} + 1$	$[-100, 100]$	0	1000

Table 1: Continued.

Name	Functions	Scope	Best	Iterations
Powell ($D = 24$)	$f_{15}(x) = \sum_{i=1}^{n/4} \left[\left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \left(x_{4i-2} + 10x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4 \right]$	$[-4, 5]$	0	1000
Eason ($D = 2$)	$f_{16}(x_1, x_2) = -\cos(x_1)\cos(x_2) * \exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$	$[-100, 100]$	-1	200
Shekel function 2 $m = 5$ ($D = 4$)	$f_{17} = -\sum_{i=1}^{m} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}, \quad m = 5$	$[0, 10]$	-10.1532	100
Shekel $m = 7$ ($D = 4$)	$f_{18} = -\sum_{i=1}^{m} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}, \quad m = 7$	$[0, 10]$	-10.4029	100
Shekel $m = 10$ ($D = 4$)	$f_{19} = -\sum_{i=1}^{m} \left[(x - a_i)(x - a_i)^T + c_i \right]^{-1}, \quad m = 10$	$[0, 10]$	-10.5364	100
Goldstein and Price ($D = 2$)	$f_{20} = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$[2, 2]$	3	1000

Table 2: The parameters setting for six algorithms.

Algorithm	Parameter values
ABC [4]	Limit = 50, the population size is 50
CS [5]	$\beta = 1.5$, $Pa = 0.25$, the population size is 50
FPA [6]	$\rho = 0.8$, the population size is 50
BA [8]	$A = 0.25$, $r = 0.5$, $f \in [0, 2]$, the population size is 50
WWO [9]	$\lambda = 0.5$, $h_{\max} = 12$, $\alpha = 1.0026$, $\beta \in [0.01, 0.25]$, $k_{\max} = \min(12, D/2)$, the population size is 50
EOBWWO	$\lambda = 0.5$, $h_{\max} = 12$, $\alpha = 1.0026$, $\beta \in [0.01, 0.25]$, $k_{\max} = \min(12, D/2)$, $\eta \in U(0, 1)$, $w \in [0.4, 1.5]$, $a = 0.02$, $b = 20$, $r = 10$, $m, n \in$ rand( ), the population size is 30

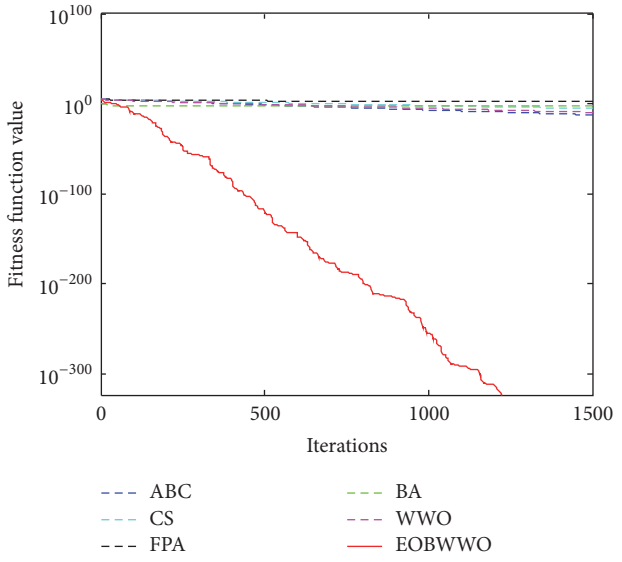

Figure 3: $D = 30$, convergence curves for f_{01}.

f_{09}–f_{13}, and f_{17}, and the exact solution of EOBWWO is obtained in some functions f_{01}–f_{05}, f_{07}, f_{09}–f_{13}, and f_{16}-f_{17}. All of these indicate that EOBWWO has a faster convergence speed and a higher calculation precision than the other comparative algorithms. Figures 23–42 show the ANOVA test of global minimum for f_{01}–f_{20}; it can be easily found that the standard deviation of EOBWWO is much smaller for most functions and the standard deviation is even zero on some functions (e.g., f_{01}–f_{05}, f_{07}–f_{13}, and f_{15}-f_{16}). Figures 23–42 imply that EOBWWO has strong stability.

4.3. High-Dimension Function Test Results. In order to validate performance of EOBWWO comprehensively, in this subsection, we choose four functions including two unimodal functions (f_{01}, f_{06}) and two multimodal functions (f_{07}, f_{09}) to test the 100 dimensions, 1000 dimensions, and 10000 dimensions, respectively, on the six algorithms. The results of all the algorithms about the four functions are summarized in Table 7. The maximum numbers of iteration of each algorithm on each function are consistent with Table 1.

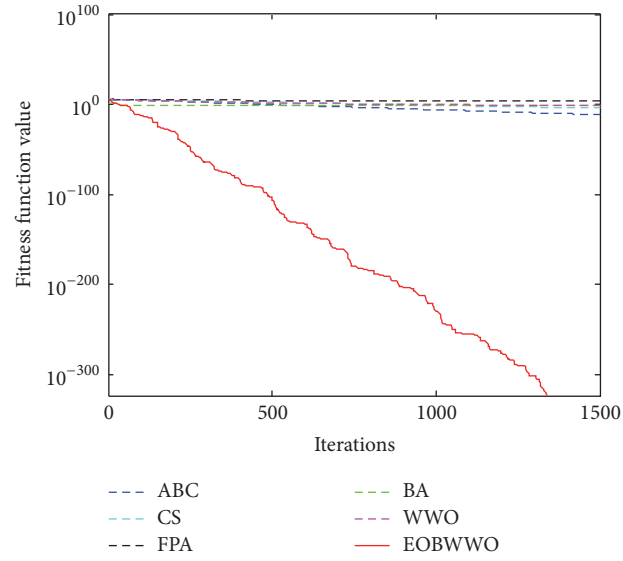

Figure 4: $D = 30$, convergence curves for f_{02}.

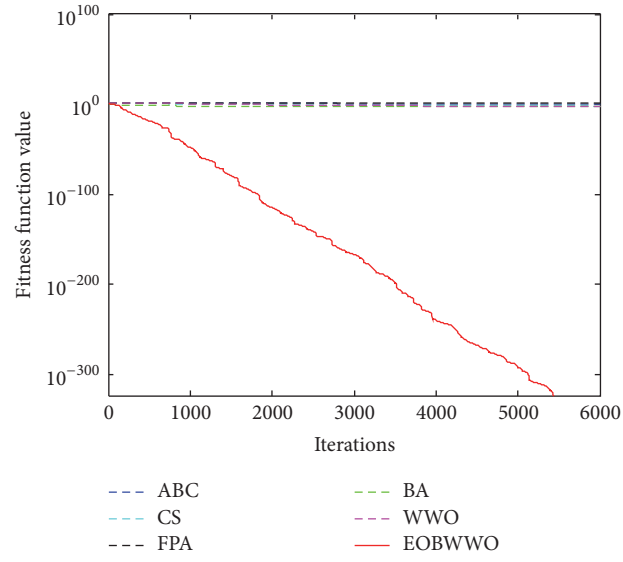

Figure 5: $D = 30$, convergence curves for f_{03}.

For unimodal function f_{01}, it can be seen obviously that the performance of EOBWWO outperforms the other comparative algorithms for dimensions 100, 1000, and 10000 from Table 7. With the increase of dimension, EOBWWO is still obtaining the exact solution and the standard deviation is zero on f_{01}. In the five comparison algorithms, as far as the median value changes in each dimension are concerned, the stability of FPA is better, but BA obtains the minimum value in each dimension. From the results of function f_{06} in Table 7, it is very easy to find that although the performance of EOBWWO is not very good when the dimension is 30, with the increase of dimension, not only does EOBWWO obtain the minimum value in each dimension, but also the standard deviation is the smallest and the range is not very

Table 3: Experiment results of unimodal functions for different algorithms ($D = 30$).

Number	Algorithm	Results				
		Max	Min	Median	Std	Rank
f_{01} ($D = 30$)	ABC	$4.44633E-12$	$3.45813E-14$	$1.92E-13$	$9.22745E-13$	2
	CS	$1.36762E-5$	$1.02709E-6$	$4.95555E-6$	$2.28819E-6$	4
	FPA	783.5925333	76.04241806	303.9240824	188.3527023	6
	BA	$1.568634E-3$	$1.029165E-3$	$1.282484E-3$	$1.31272E-4$	5
	WWO	$5.55407E-10$	$1.07596E-10$	$2.69635-10$	$1.15369E-10$	3
	EOBWWO	0	**0**	**0**	**0**	1
f_{02} ($D = 30$)	ABC	$3.44699E-11$	$1.17159E-13$	$1.51639E-12$	$6.29287E-12$	2
	CS	$1.28405E-4$	$0.53823E-5$	$6.08593E-5$	$2.96568E-5$	3
	FPA	$1.09372E+4$	$1.677E+3$	$3.61889E+3$	$1.89233E+3$	6
	BA	0.037048461	0.01382428	0.020204478	0.004767846	5
	WWO	0.511649493	0.000169285	0.013059176	0.200939335	4
	EOBWWO	0	**0**	**0**	**0**	1
f_{03} ($D = 30$)	ABC	0.829003374	0.3204228717	0.555760641	0.120472774	5
	CS	0.088057162	0.000337518	0.007576085	0.022275036	3
	FPA	15.1544391	4.697724887	8.12103893	2.186818227	6
	BA	0.015412963	0.010993868	0.013059351	0.001157761	4
	WWO	0.331726179	$6.95792E-9$	$1.53742E-6$	0.062504822	2
	EOBWWO	0	**0**	**0**	**0**	1
f_{04} ($D = 30$)	ABC	$6.47604E-24$	$6.96705E-28$	$2.21019E-26$	$1.20736E-24$	2
	CS	74.1784179	0.006545946	3.896226938	19.85723139	4
	FPA	$2.64849E+13$	11.2387456	3401.123936	$4.8539E+12$	6
	BA	1.089641426	0.116013073	0.14358676	0.180342929	3
	WWO	723.5000563	349.8831021	530.7411865	92.94725208	5
	EOBWWO	0	**0**	**0**	**0**	1
f_{05} ($D = 30$)	ABC	0	**0**	**0**	**0**	1
	CS	0	**0**	**0**	**0**	1
	FPA	1464	194	692.5	300.9258052	6
	BA	1	0	0	0.449776445	4
	WWO	5	0	0	1	5
	EOBWWO	0	**0**	**0**	**0**	1
f_{06} ($D = 30$)	ABC	27.62119073	**7.911323481**	**20.45437209**	4.364699718	1
	CS	26.10849123	23.69813132	25.20307209	0.695990009	2
	FPA	83.14751217	43.48073835	61.12668297	11.14304633	6
	BA	25.51159629	24.8746208	27.74338401	0.985458463	4
	WWO	29.32841665	24.00396307	27.69619184	1.153955218	3
	EOBWWO	28.9707934	28.2967605	28.8377944	**0.162741102**	5

large. In addition, for EOBWWO, the change of the order of magnitudes of median is the smallest in different dimensions. Those provide strong evidence that EOBWWO has higher performance in dealing with complex functions. Among the other comparative algorithms, the performance of BA is better, the second is CS, the third is WWO, and the fourth and fifth are BC and FPA, respectively.

For multimodal functions, EOBWWO obtained the exact solution and the standard deviation is zero on f_{07} and f_{09} for the dimensions 100, 1000, and 10000. Taking into account median value, the performance of BA is better among comparative algorithms. However, considering the order of magnitudes of median, the difference between the five algorithms is not obvious.

Furthermore, some high dimensional tests of EOBWWO are also tested in functions f_{02}, f_{04}, f_{10}, f_{12}, and f_{13}; details of the experimental results are shown in Table 8. As shown in Tables 7 and 8 and the above analysis, EOBWWO has

Table 4: Experiment results of multimodal functions for different algorithms ($D = 30$).

Number	Algorithm	Results				
		Max	Min	Median	Std	Rank
f_{07} ($D = 30$)	ABC	4.543430376	0.872091069	2.32396378	0.77610662	2
	CS	100.3117085	69.593399457	83.78234227	9.01595302	5
	FPA	155.7273116	103.5726964	121.9253488	12.6731423	6
	BA	44.0464352	9.204919504	26.66189504	8.91738265	3
	WWO	119.6471701	47.99518568	81.08466559	20.4603731	4
	EOBWWO	0	**0**	**0**	**0**	1
f_{08} ($D = 30$)	ABC	0.200343498	0.077326914	0.142626325	0.03146059	2
	CS	2.644000719	1.141797447	2.096514981	0.38634392	4
	FPA	7.743838028	3.174296934	4.343237221	0.91961437	6
	BA	2.121661491	0.026334395	1.250303713	0.79120840	3
	WWO	8.74527255	2.713986881	3.436036696	1.37954679	5
	EOBWWO	8.88178$E-16$	**8.88178E − 16**	**8.88178E − 16**	**0**	1
f_{09} ($D = 30$)	ABC	0.009995413	1.40219$E-6$	1.03254$E-5$	0.00251343	2
	CS	0.21732293	0.046844017	0.097110776	0.04071839	5
	FPA	7.743838028	3.174296934	4.343237221	0.91961437	6
	BA	8.8209$E-5$	4.84725$E-5$	6.96553$E-5$	1.0629$E-5$	3
	WWO	0.027745165	0.000101686	0.004216065	0.008990433	4
	EOBWWO	0	**0**	**0**	**0**	1
f_{10} ($D = 30$)	ABC	1.82152$E-5$	3.55257$E-7$	4.91547$E-6$	3.92981$E-6$	2
	CS	23.75901871	20169151391	8.018029438	6.27034500	4
	FPA	2.78$E+8$	1.07$E+7$	7.32$E+7$	7.20$E+7$	6
	BA	0.517543758	0.102778936	0.249802147	0.10685336	3
	WWO	3.57$E+5$	1148.931415	28435.22582	70797.7997	5
	EOBWWO	3.2059$E-249$	**0**	**4.3182E − 304**	**0**	1
f_{11} ($D = 30$)	ABC	−0.025319222	−0.025319222	−0.025319992	8.93607$E-15$	6
	CS	−0.100574784	−0.100574784	−0.100574784	4.53639$E-14$	3
	FPA	−0.100574784	−0.100574784	−0.100574784	8.35309$E-12$	5
	BA	−0.100574784	−0.100574784	−0.100574784	1.18291$E-14$	3
	WWO	−0.100574784	−0.100574784	−0.100574784	1.93291$E-15$	2
	EOBWWO	−1	**−1**	**−1**	**0**	1
f_{12} ($D = 30$)	ABC	0.02864526	0.007552463	0.012569017	0.00450223	2
	CS	32.10594109	16.77196848	26.78636751	3.59576559	5
	FPA	19.30313173	12.51112225	15.67535455	1.74928731	4
	BA	42.35819689	25.57284909	31.84105217	3.87397750	6
	WWO	19.88306619	9.975451796	14.2300995	2.59319376	3
	EOBWWO	0	**0**	**0**	**0**	1
f_{13} ($D = 30$)	ABC	8.85875$E-8$	6.75126$E-15$	6.88565$E-9$	2.1219$E-8$	2
	CS	3.531716667	1.26783277	2.278563536	0.625010328	5
	FPA	7.466256332	0.073108474	2.782338916	1.589472384	6
	BA	0.018080917	0.012650753	0.014787578	0.001091235	3
	WWO	1.942396367	0.026078564	0.13612587	0.64846015	4
	EOBWWO	0	**0**	**0**	**0**	1
f_{14} ($D = 30$)	ABC	2.600171726	1.999873348	2.399879727	0.184266606	5
	CS	1.633485698	0.922951921	1.400436259	0.17500411	4
	FPA	5.30000938	2.701931127	3.871080144	0.61020963	6
	BA	0.499873346	0.199873346	0.299873346	0.06214554	2
	WWO	1.211882567	0.599906059	0.808361684	0.174395376	3
	EOBWWO	0.099873348	**0**	**0.099873346**	**0.018234295**	1

Table 5: Experiment results of low-dimension functions for different algorithms.

Number	Algorithm	Results				
		Max	Min	Median	Std	Rank
f_{15} ($D = 24$)	ABC	0.455803159	0.088628487	0.244130317	0.092018378	5
	CS	0.006558415	0.002334688	0.003589486	0.001062387	2
	FPA	3.516583313	0.439394393	1.119151909	0.756461677	4
	BA	0.085283799	0.017280708	0.035769424	0.019917509	3
	WWO	1.582142729	0.0749512	0.437894406	0.30106576	6
	EOBWWO	$3.44639E-63$	**3.5938E − 262**	**6.1033E − 207**	**6.29221E − 64**	1
f_{16} ($D = 2$)	ABC	−1	**−1**	**−1**	**0**	1
	CS	−1	−1	−1	$1.47866E-13$	1
	FPA	−0.999991624	−1	−0.999999921	$1.56202E-6$	6
	BA	−0.999999927	−1	−0.999999978	$2.08821E-8$	5
	WWO	−1	−1	−1	$2.06163E-17$	1
	EOBWWO	−1	**−1**	**−1**	**0**	1
f_{17} ($D = 4$)	ABC	−10.1531456	−10.15319968	−10.15319968	$1.0346E-5$	1
	CS	−10.15319968	**−10.15319968**	**−10.15319968**	**5.03511E − 15**	1
	FPA	−10.15315656	−10.15319968	−10.15319952	$7.82319E-6$	5
	BA	−5.055111551	−5.055182873	−50.551596	$1.85604E-5$	6
	WWO	−5.10077214	−10.15319968	−10.15319968	1.281841952	1
	EOBWWO	−10.14720526	−10.15319968	−10.15319968	0.001093567	1
f_{18} ($D = 4$)	ABC	−10.40251326	−10.40294057	−10.40294057	$7.79447E-5$	4
	CS	−10.40294057	−10.40294057	−10.40294057	$2.21007E-14$	3
	FPA	−10.40202483	**−10.40294054**	**−10.40293239**	$1.60136E-5$	1
	BA	−5.087559948	−5.08766627	−5.087628993	$2.54514E-5$	6
	WWO	−10.40294057	−10.40294057	−10.40294057	**2.00647E − 15**	2
	EOBWWO	−10.40130898	−10.40294057	−10.40294057	0.000360071	5
f_{19} ($D = 4$)	ABC	−5.174214497	−10.53640982	−10.53640982	0.978996012	2
	CS	−10.53640982	−10.53640982	−10.53640982	$1.33033E-12$	2
	FPA	−10.53388561	**−10.53640939**	**−10.5363024**	**0.000210477**	1
	BA	−5.128396631	−10.53632794	−5.128439601	1.37202234	6
	WWO	−5.175646742	−10.53640982	−10.53640982	0.978736954	2
	EOBWWO	−10.52292613	−10.53640982	−10.53640982	0.002461764	2
f_{20} ($D = 2$)	ABC	3.00425387	3	**3**	0.00077653	1
	CS	3	3	**3**	**1.91987E − 15**	1
	FPA	3	3	**3**	$2.62468E-15$	1
	BA	3.000011451	3.00000068	3.000001582	$2.81115E-6$	6
	WWO	3	3	**3**	$4.77801E-15$	1
	EOBWWO	3	3	**3**	$4.14957E-15$	1

the ability efficiently and stably to handle high dimensional functions.

4.4. Structural Engineering Design Examples. Many structural design problems in the real world are constrained optimization problems which are nonlinear with complex constraints and the optimal solution even does not exist in some cases. In order to evaluate the performance of EOBWWO even further, in this subsection, EOBWWO was used to solve two structural design problems: design of a compressing spring and design of a welded beam.

4.4.1. Test Problem 1: Design of a Tension/Compression Spring. Design of a tension or compressing spring problem is introduced by Belegundu [14] firstly and it deals with the optimal design of tension/compression spring for a minimum weight. As shown in Figure 43, a tension/compression spring problem has three design variables: the wire diameter $d(x_1)$,

Table 6: Statistical comparison between EOBWWO and the other five algorithms.

	ABC		CS		FPA		BA		WWO	
	p value	h	p value	h	p value	h	p value	h	p value	h
f_{01}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{02}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{03}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{04}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{05}	NaN	0	NaN	0	$1.2108E-12$	1	0.0027	1	$1.6428E-9$	1
f_{06}	$3.0199E-11$	1	$3.0199E-11$	1	$3.0199E-11$	1	$6.2828E-6$	1	$2.0338E-9$	1
f_{07}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{08}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{09}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{10}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{11}	$1.2059E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.197E-12$	1	$5.1032E-13$	1
f_{12}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{13}	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1
f_{14}	$2.5190E-11$	1	$2.5190E-11$	1	$2.5190E-11$	1	$2.5190E-11$	1	$2.5190E-11$	1
f_{15}	$3.0199E-11$	1	$3.0199E-11$	1	$3.0199E-11$	1	$3.0199E-11$	1	$3.0199E-11$	1
f_{16}	NaN	0	$1.20088E-12$	1	$1.2118E-12$	1	$1.2118E-12$	1	0.3337	0
f_{17}	$3.1245E-4$	1	$1.0446E-7$	1	$1.7216E-6$	1	$2.9953E-11$	1	$1.6737E-4$	1
f_{18}	0.0851	0	$1.3993E-5$	1	$1.587E-7$	1	$2.9916E-11$	1	$1.187E-7$	1
f_{19}	0.159	0	0.0371	1	$5.4608E-10$	1	$3.6058E-11$	1	$2.1012E-6$	1
f_{20}	0.5130	0	$2.8286E-9$	1	0.1657	0	$2.9766E-11$	1	0.5097	0

the mean coil diameter $D(x_2)$, and the number of active coils $N(x_3)$. The minimum weight is subject to constraints on minimum deflection, shear stress, surge frequency, and limits on outside diameter [15]. A detailed description of the problem is as follows:

$$\text{Minimize:} \quad f(X) = (x_3 + 2) \times x_2 x_1^2 \tag{15}$$

$$\text{Subject to:} \quad g_1(X) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \tag{16}$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \tag{17}$$

$$g_3(X) = 1 - \frac{140.45 x_1}{x_2^3 x_3} \leq 0 \tag{18}$$

$$g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0, \tag{19}$$

where the experimental parameters are set as follows: $(0.05, 0.25, 2) \leq X(x_1, x_2, x_3) \leq (1, 1.3, 15)$ [29]. Table 9 lists the optimal solution for compression spring design obtained by EOBWWO. The results are 20 runs independently and the number of iterations of EOBWWO is 5000. In the process of dealing with tension/compression spring constrained optimization problem, first of all we need to determine whether the four constraints are satisfied. If these constraint conditions are all satisfied, then calculate $f(X)$ according to formula (15) and compare $f(X)$ with the original fitness values, the better result as fitness value of constrained optimization problem. Otherwise, the original fitness value remains and continues to iterate.

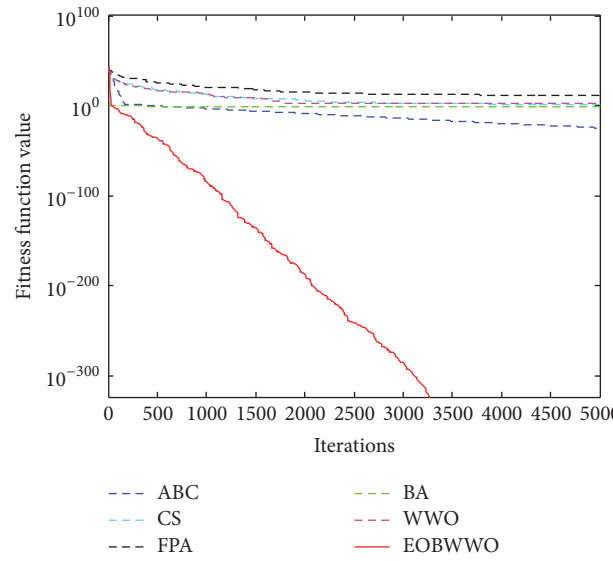

Figure 6: $D = 30$, convergence curves for f_{04}.

Table 7: Experiment results of high-dimension functions for different algorithms (f_{01}, f_{06}, f_{07}, f_{09}).

Number	Algorithm	Dimension	Worst	Best	Median	Std
f_{01}	ABC	100	7.637713	1.257573	3.295669	1.651127
		1000	$1.434828E+6$	$1.304834E+6$	$1.371354E+6$	$3.090254E+4$
		10000	$3.062472E+7$	$2.988161E+7$	$3.023603E+7$	$1.446642E+4$
	CS	100	26.040384	9.3680842	15.818794	4.115038
		1000	$6.516648E+4$	$4.647782E+4$	$5.504538E+4$	$4.829252E+3$
		10000	$1.280442E+6$	$1.058719E+6$	$1.167367E+6$	$5.345232E+4$
	FPA	100	$8.187843E+3$	$4.416928E+3$	$5.803334E+3$	$1.062651E+3$
		1000	$1.400121E+5$	$1.400121E+5$	$1.400121E+5$	$1.400121E+5$
		10000	$1.550111E+6$	$9.442398E+5$	$1.182740E+6$	$1.704160E+6$
	BA	100	0.021140987	0.015486339	0.018443888	0.001474661
		1000	3.550121599	3.091995126	3.394685417	0.109635184
		10000	1130.145132	750.0726451	865.3136568	94.15923749
	WWO	100	6.483196131	3.07003824	0.917763681	0.91773681
		1000	$2.763344E+5$	$2.088491E+5$	$2.375400E+5$	$1.655635E+4$
		10000	$7.929174E+6$	$5941746E+6$	$6.548260E+5$	$4.464375E+5$
	EOBWWO	100	0	0	0	0
		1000	0	0	0	0
		10000	0	0	0	0
f_{06}	ABC	100	470.7882	276.5788	407.6043	48.01618428
		1000	202219.7218	177258.0229	193783.4147	5721.958598
		10000	4156258.387	4029855.806	4108026.596	2985721724
	CS	100	121.3484018	108.7940185	114.162504	3.194704574
		1000	5108.984648	3833.317047	4359.012469	293.4115925
		10000	73588.75287	57413.36292	66913.83583	4034.78353
	FPA	100	559.2407	313.9292	421.0102	53.82862792
		1000	8635.941471	5242.413467	6518.179353	870.7185594
		10000	87358.09362	59543.23061	71905.7802	7351.701592
	BA	100	100.7466	97.73262	99.84007	0.79162325
		1000	1614.346207	1424.000673	1479.067336	47.45627579
		10000	201943.4784	116612.8731	136871.9433	20275.79495
	WWO	100	264.9252737	103.030758	118.948408	36.85200769
		1000	20094.16217	13833.01993	16535.50983	1361.529306
		10000	566850.0669	398083.154	470903.7464	37718.72406
	EOBWWO	100	98.93733	98.35499	98.83611	0.1326465
		1000	998.9477482	998.4598459	998.7574767	0.133930167
		10000	9998.963982	9997.990523	9998.740734	0.255601018
f_{07}	ABC	100	136.8533	108.4902	122.9485	8.215112956
		1000	11758.89361	11266.87845	11600.7465	122.7986144
		10000	175645.833	173987.3858	174819.5595	470.6736986
	CS	100	532.4556943	393.0034167	456.391023	79.34852616
		1000	11758.89361	11266.87845	11600.7465	159.2707509
		10000	100168.3367	97832.02684	98994.11566	503.1069646
	FPA	100	719.314	632.8825	674.7777	23.77504554
		1000	9679.894951	9201.466552	9432.549086	125.8751735
		10000	102588.4972	99293.30339	101542.8715	817.4508097
	BA	100	146.1837	48.6256	102.4563	21.99337966
		1000	1868.357958	1058.160315	1289.192742	214.602082
		10000	62214.8107	58922.9904	60387.27428	739.8935726
	WWO	100	752.071607	532.8408358	645.477105	57.03009115
		1000	1632.88218	9939.23755	10334.79578	192.1639397
		10000	116931.0586	112418.7607	114726.7636	1248.129627
	EOBWWO	100	0	0	0	0
		1000	0	0	0	0
		10000	0	0	0	0

Table 7: Continued.

Number	Algorithm	Dimension	Worst	Best	Median	Std
f_{09}	ABC	100	5.135663	1.823636	2.836978	1.811452675
		1000	16666.76931	15706.49796	16346.9208	252.8474194
		10000	28259.6583	274901.5152	279624.617	1490.552232
	CS	100	3.870724583	2.423491994	3.008555665	0.41301955
		1000	797.6679633	582.0471041	690.0181057	55.72921205
		10000	12543.25214	9844.343528	10872.16696	672.723997
	FPA	100	106.3429	50.14951	70.05947	11.61798957
		1000	1342.792454	861.2975273	1101.229098	134.1435148
		10000	13324.24358	9169.071683	11107.79476	1225.099338
	BA	100	$6.2E-4$	$3.56E-4$	$4.84E-4$	$5.58029E-5$
		1000	0.025577018	0.021496965	0.023453281	0.001074614
		10000	0.744676593	0.500121455	0.599339533	0.058517493
	WWO	100	1.648358348	1.294987676	1.41369559	0.087489969
		1000	4194.059126	2989.289164	3304.633104	257.7285706
		10000	67780.01333	53734.62138	60711.98145	2937.168912
	EOBWWO	100	0	0	0	0
		1000	0	0	0	0
		10000	0	0	0	0

Table 8: Experiment results of high-dimension functions of EOBWWO.

Functions	Dimensions	Worst	Best	Median	Std
f_{02}	10000	0	0	0	0
f_{04}	10000	0	0	0	0
f_{10}	10000	0	0	0	0
f_{12}	10000	0	0	0	0
f_{13}	10000	0	0	0	0

Table 9: Statistical results of best tension/compression spring model obtained by EOBWWO.

Worst	Best	Mean	Std
0.012703814	0.012665234	0.012680022	$1.20767E-5$

In the process of using EOBWWO algorithm to deal with tension/compression spring constrained optimization problem, when the position of individual in current population is changed (create a new solution) and required to estimate the new solution, the steps of dealing with the tension/compression spring constrained optimization problems are as follows.

Step 1. Calculate the values of the four constraint conditions (see (16)–(19)) and estimate whether they are all satisfied with the constraint conditions. If these constraint conditions are all satisfied, go to Step 2; otherwise, go to Step 3.

Step 2. Calculate fitness value of new solution by formula (15), and compare new fitness value to original fitness value. According to the comparison results determine whether to update the current individual. Go to Step 4.

Step 3. Keeping the original individual that violates any constraint, go to Step 4.

Step 4. Continue performing the following operations.

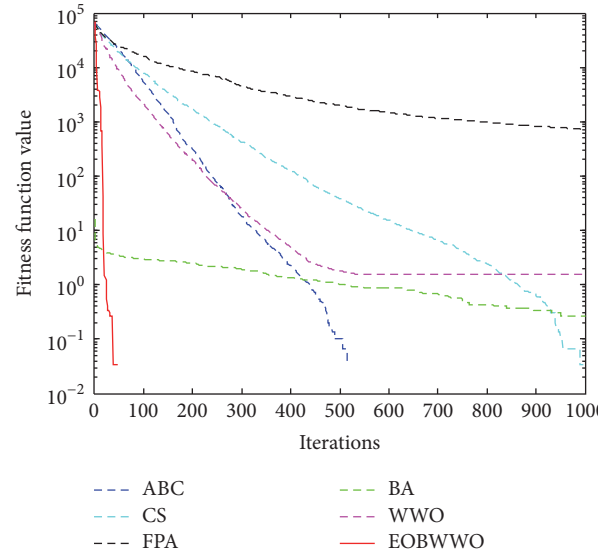

Figure 7: $D = 30$, convergence curves for f_{05}.

The reason behind keeping the individual that violates any constraint of the constraint conditions is that each iteration of the population has to implement the elite opposition-based learning (EOBL) strategy and propagation operation; then the individual that violates any constraint may satisfy the constraints in the next iteration.

Table 10: Best results of compression spring by different model.

Researcher(s)	Method	Design variables			$f(X)$
		$x_1(d)$	$x_2(D)$	$x_3(N)$	
Belegundu [14]	a	0.05	0.315900	14.25000	0.0128334
Arora [15]	b	0.053396	0.399180	9.185400	0.0127303
Coello [16]	GA	0.051480	0.351661	11.632201	0.01270478
Coello [17]	GA	0.05148	0.35166	11.6322	0.0127
Coello and Montes [18]	GA	0.051989	0.363965	10.890522	0.0126810
He et al. [19]	PSO	0.05169040	0.35674999	11.28712599	0.0126652812
Coello and Becerra [20]	EP[c]	0.05	0.3174	14.0318	0.01272
Raj et al. [21]	ECT[d]	0.05386200	0.41128365	8.68437980	0.01274840
Hedar and Fukushima [22]	SA[e]	0.051742503409	0.358004783455	11.2139073627	0.012665285
He and Wang [23]	PSO	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [24]	ES[f]	0.051643	0.355360	11.397926	0.012698
Omran and Salman [25]	CODEQ[g]	0.0516837458	0.3565898352	11.2964717107	0.0126652375
Aragón et al. [26]	T-cell[h]	0.05162	0.35511	11.3845	0.01267
Akay and Karaboga [27]	ABC	0.051749	0.358179	11.203763	0.012665
Gandomi et al. [28]	BA	0.05169	0.35673	11.2885	0.01267
Gandomi [29]	ISA[i]	NA	NA	NA	0.012665
Baykasoğlu and Ozsoydan [30]	FA	0.0516674837	0.3561976945	11.3195613646	0.0126653049
Present study	EOBWWO	**0.05169826**	**0.356939073**	**11.2760014**	**0.012665234**

a: mathematical optimization technique; b: numerical optimization technique; c: evolutionary programming; d: evolutionary computational technique; e: simulated annealing; f: evolution strategies; g: chaotic search, opposition-based learning, differential evolution, and quantum mechanics; h: TCA is the T-cell algorithm; i: interior search algorithm; NA: there is no relevant data.

Table 11: The optimal solution of the welded beam design example obtained by EOBWWO.

Worst	Best	Mean	Std
1.71846399	1.69634711	1.70146693	0.005170473

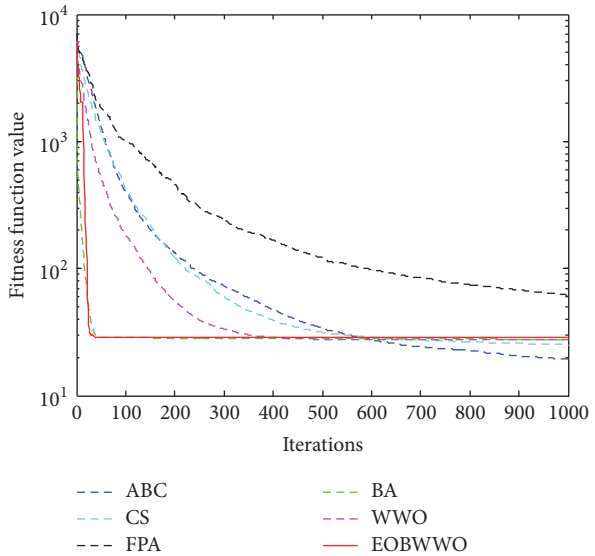

Figure 8: $D = 30$, convergence curves for f_{06}.

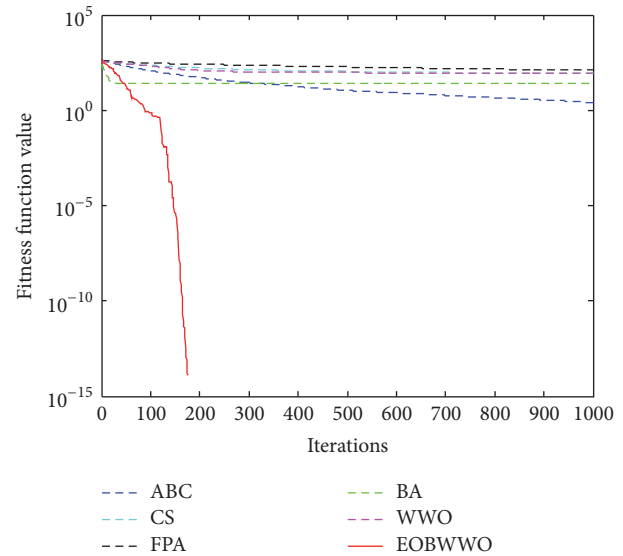

Figure 9: $D = 30$, convergence curves for f_{07}.

As one of the most well-known design benchmark problems, many researchers have studied this problem. Belegundu [14] introduced this problem and used eight different mathematical optimization techniques for this problem. Arora [15] solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Table 10 summarized the optimal results of design of a tension/compression spring obtained by EOBWWO and other researchers.

As seen from Table 10, the proposed method obtained the best design overall of 0.012665234 corresponding to $X = (0.05169826, 0.356939073, 11.2760014)$ and the results

Table 12: The optimal solution of the welded beam design example using different methods.

Researcher(s)	Method	Design variables				$f(X)$
		$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	
Deb [31]	GA	0.2489	6.1730	8.1789	0.2533	2.4331
Leite and Topping [32]	GA	0.2489	6.1097	8.2484	0.2485	2.4000
Coello [33]	GA	0.2088	3.4205	8.9975	0.2100	1.7483
Deb [34]	GA	NA	NA	NA	NA	2.38
Coello [16]	GA	0.208800	3.420500	8.997500	0.210000	1.748309
Hu et al. [35]	PSO	0.20573	3.47049	9.03662	0.20573	1.72485084
He et al. [19]	PSO	0.2444	6.2175	8.2915	0.2444	2.3810
Liu [36]	SA	0.2444	6.2175	8.2915	0.2444	2.3810
Hedar and Fukushima [22]	SA	0.205644	3.4725787	9.03662391	0.2057296	1.7250022
He and Wang [23]	PSO	0.202369	3.544214	9.048210	0.205723	1.728024
Mahdavi et al. [37]	HSA	0.20573	3.47049	9.03662	0.20573	1.7248
Montes and Coello [24]	ES	0.199742	3.612060	9.037500	0.206082	1.73730
Fesanghary et al. [38]	HHSA	0.20572	3.47060	9.03682	0.20572	1.7248
Kaveh and Talatahari [39]	PSO + ABC	0.205729	3.469875	9.036805	0.205765	1.724849
Kaveh and Talatahari [40]	ABC	0.205700	3.471131	9.036683	0.205731	1.724918
Gandomi et al. [41]	FA	0.2015	3.562	9.0414	0.2057	1.73121
Akay and Karaboga [27]	FA	0.205730	3.470489	9.036624	0.205730	1.724852
Gandomi et al. [28]	BA	0.2015	3.562	9.0414	0.2057	1.7312
Gandomi [29]	ISA	NA	NA	NA	NA	2.3812
Baykasoğlu and Ozsoydan [30]	FA	0.205730	3.470489	9.036624	0.205730	1.724852
Present study	EOBWWO	**0.205832588**	**3.253654976**	**9.0315042**	**0.205962951**	**1.69634711**

HAS: harmony search algorithm; HHSA: hybridizing harmony search algorithm.

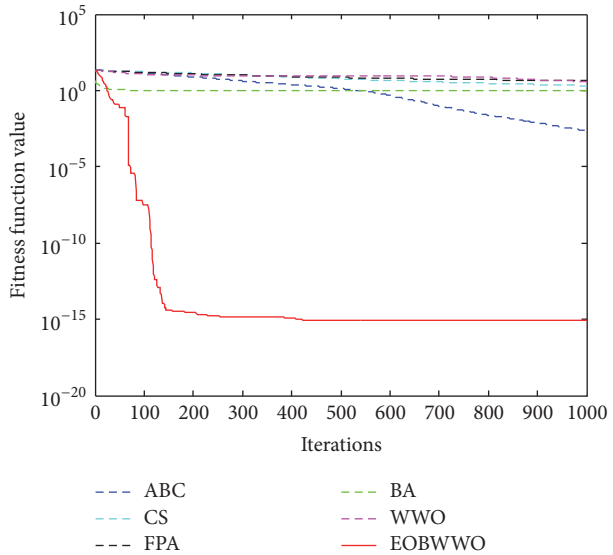

Figure 10: $D = 30$, convergence curves for f_{08}.

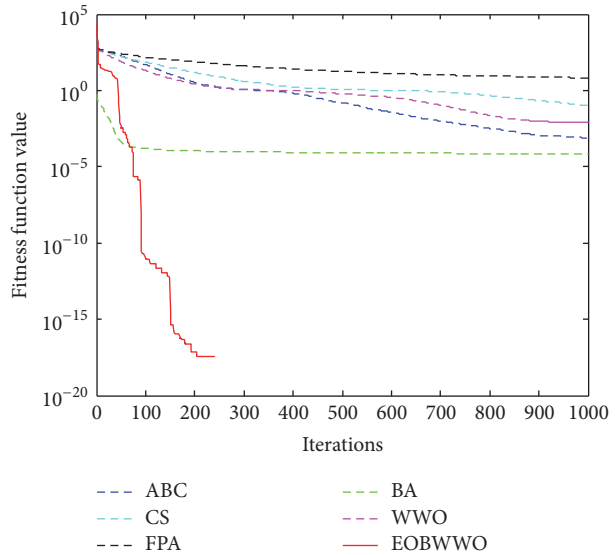

Figure 11: $D = 30$, convergence curves for f_{09}.

obtained by EOBWWO are better than the comparative methods.

4.4.2. Test Problem 2: Design of a Welded Beam. As another well-known design benchmark problem, the objective of welded beam problem is to minimize overall cost of fabrication subject to constraints on shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, end deflection of the beam δ, and side constraints. As depicted in Figure 44 [28], this problem consists of four design variables: thickness of the

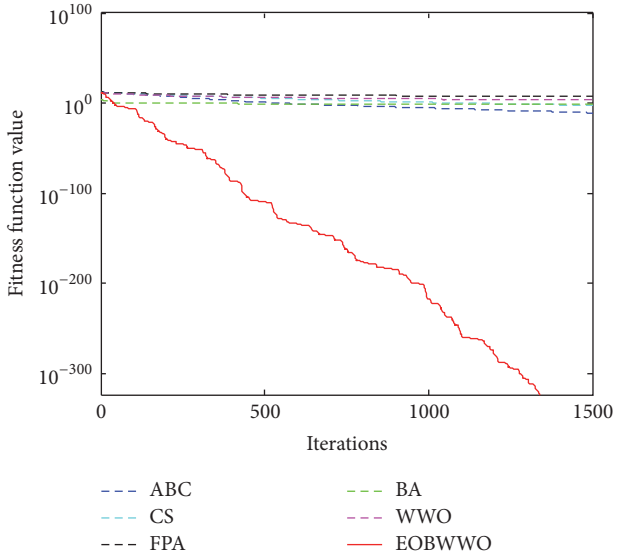

Figure 12: $D = 30$, convergence curves for f_{10}.

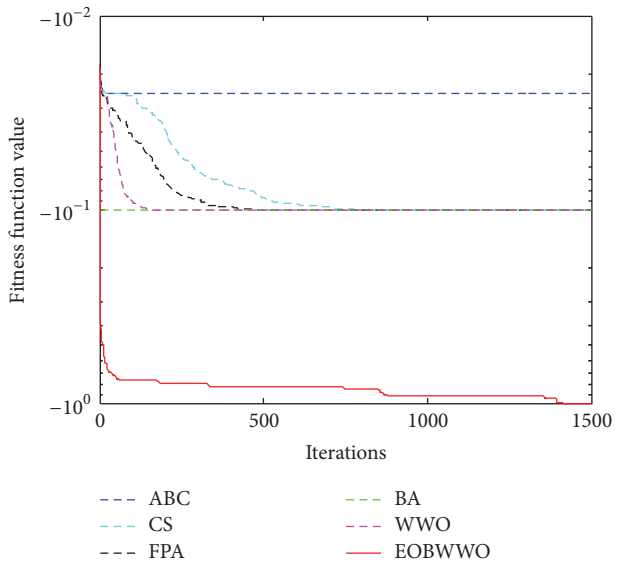

Figure 13: $D = 30$, convergence curves for f_{11}.

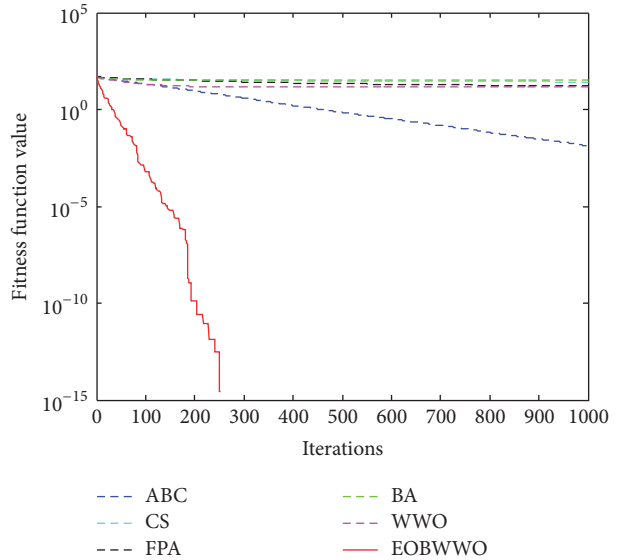

Figure 14: $D = 30$, convergence curves for f_{12}.

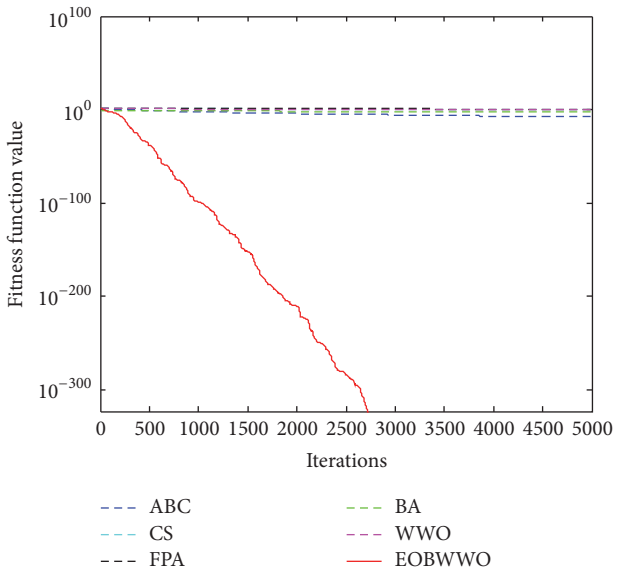

Figure 15: $D = 30$, convergence curves for f_{13}.

weld $h(x_1)$, the length of the welded joint $l(x_2)$, the width of the beam $t(x_3)$, and the thickness of the beam $b(x_4)$.

The problem can be formulated as follows:

$$\begin{aligned} \text{Minimize:}\quad & f(X) \\ & = 1.10471 x_1^2 x_2 \\ & \quad + 0.04811 x_3 x_4 (x_2 + 14) \end{aligned} \tag{20}$$

$$\text{Subject to:}\quad g_1(X) = \tau(X) - \tau_{\max} \le 0 \tag{21}$$

$$g_2(X) = \sigma(X) - \sigma_{\max} \le 0 \tag{22}$$

$$g_3(X) = x_1 - x_4 \le 0 \tag{23}$$

$$g_4(X) = 0.15 - x_1 \le 0 \tag{24}$$

$$g_5(X) = \delta(X) - 0.25 \le 0 \tag{25}$$

$$g_6(X) = P - P_c(X) \le 0 \tag{26}$$

$$\begin{aligned} & g_7(X) \\ & = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) \\ & \quad - 5 \le 0, \end{aligned} \tag{27}$$

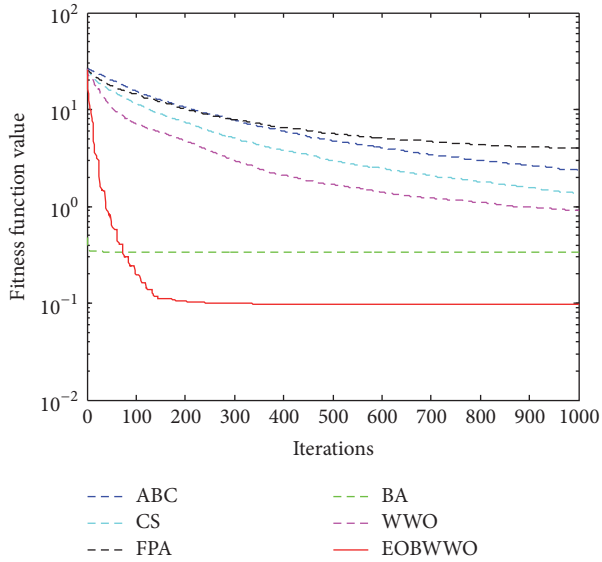

Figure 16: $D = 30$, convergence curves for f_{14}.

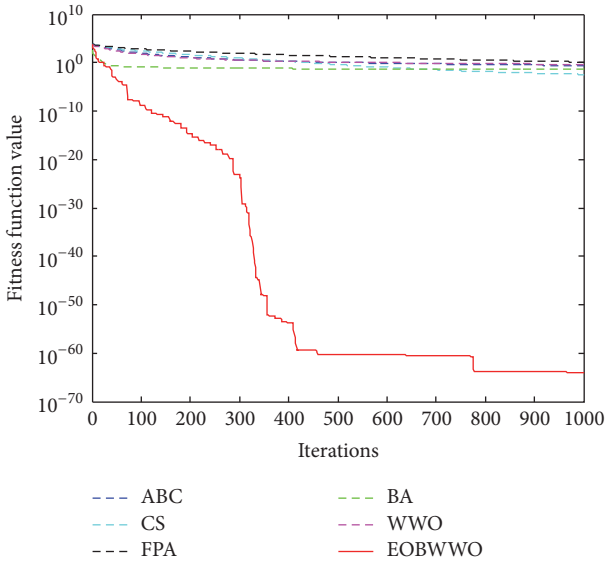

Figure 17: $D = 24$, convergence curves for f_{15}.

where

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2},$$

$$\tau'' = \frac{MR}{J},$$

$$M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$

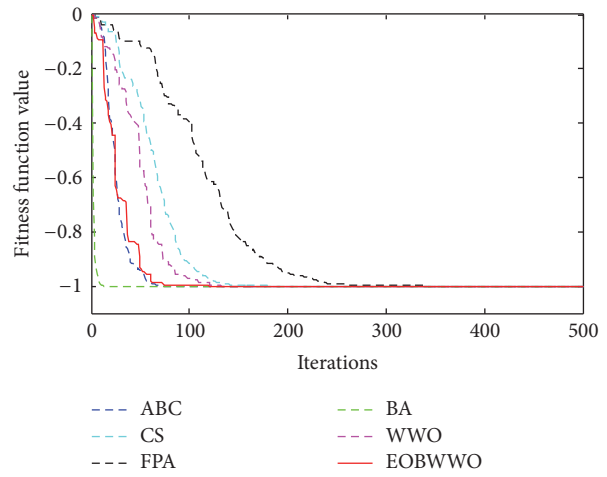

Figure 18: $D = 2$, convergence curves for f_{16}.

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(X) = \frac{6PL}{x_3^2x_4},$$

$$\delta(X) = \frac{4PL^3}{Ex_3^3x_4},$$

$$P_c(X) = \frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P = 6,000\,\text{lb},$$

$$L = 14\,\text{in},$$

$$E = 30 \times 10^6\,\text{psi},$$

$$G = 12 \times 10^6\,\text{psi},$$

$$X = (x_1, x_2, x_3, x_4),$$

$$(0.1, 0.1, 0.1, 0.1) \le X \le (2, 10, 10, 2). \tag{28}$$

EOBWWO algorithm was run to find the minimum cost of fabrication of this design problem, where the range of values of the four experimental parameters x_1, x_2, x_3, x_4 is set as follows: $(0.1, 0.1, 0.1, 0.1) \le X(x_1, x_2, x_3, x_4) \le (2, 10, 10, 2)$ [30]. Similarly, in the process of searching the minimum overall cost of fabrication, it is also the first to determine whether the seven constraint conditions are satisfied when calculating the fitness value. If these constraint conditions are all satisfied, then make use of formula (20) to calculate fitness value. In each run, EOBWWO is at the cost of 5000-function evaluation to locate the optimal solution. The experimental results of design of a welded beam obtained by EOBWWO are listed in Table 11. Similarly, in the process of searching the minimum overall cost of fabrication, when the position of any individual in the current population is changed and required to estimate the new solution, the steps

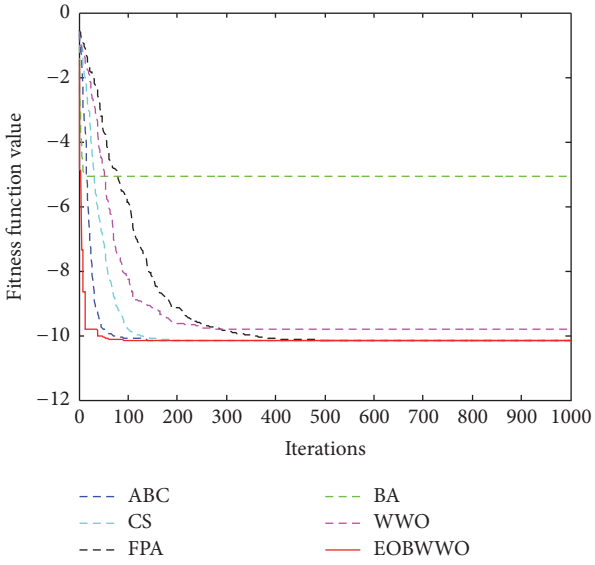

Figure 19: $D = 4$, convergence curves for f_{17}.

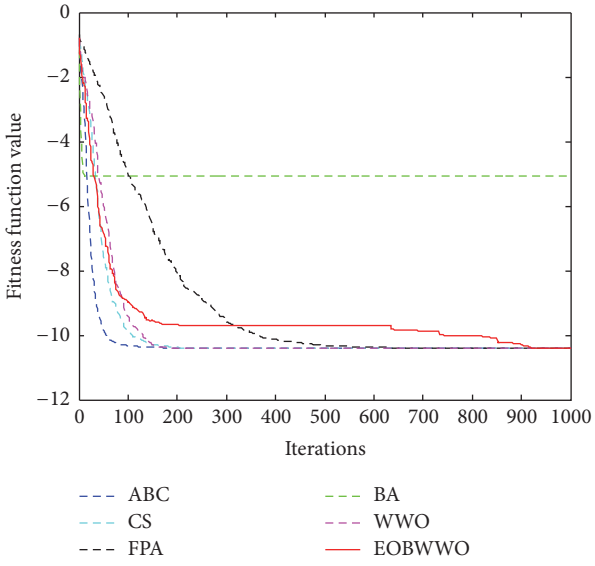

Figure 20: $D = 4$, convergence curves for f_{18}.

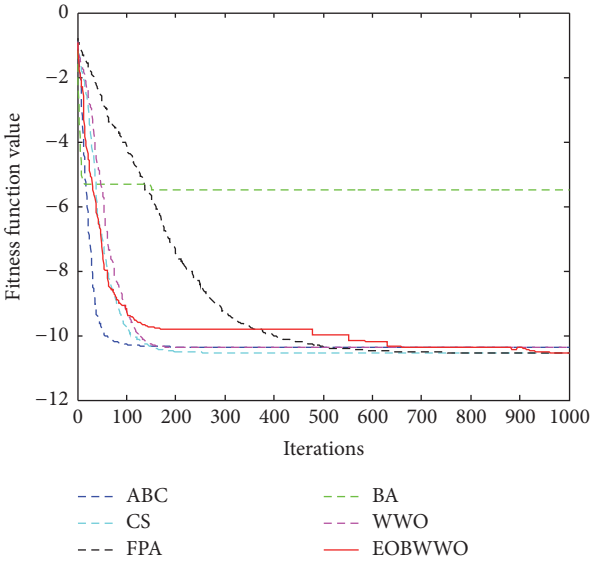

Figure 21: $D = 4$, convergence curves for f_{19}.

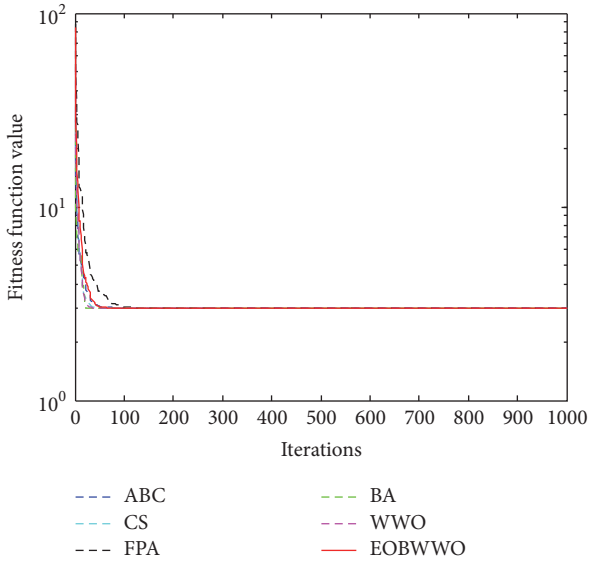

Figure 22: $D = 2$, convergence curves for f_{20}.

of using EOBWWO algorithm to deal with welded beam problem are as using EOBWWO algorithm to deal with compression spring.

The results of comparing with those of other optimization algorithms reported in the literature are shown in Table 12. It can be seen from Table 12 remarkably that the proposed EOBWWO algorithm is much better than other algorithms in the design of welded beam, and the optimal solution obtained by EOBWWO is 1.69634711 corresponding to $X = (0.205832588, 3.253654976, 9.0315042, 0.25962951)$.

5. Conclusions

In this paper, three strategies are added to the original WWO algorithm to improve the convergence speed and

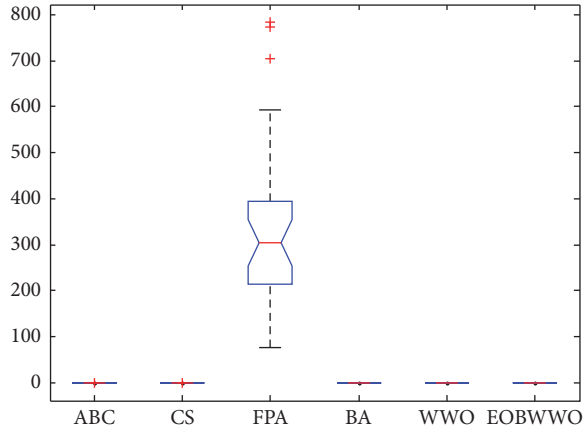

Figure 23: $D = 30$, ANOVA test of global minimum for f_{01}.

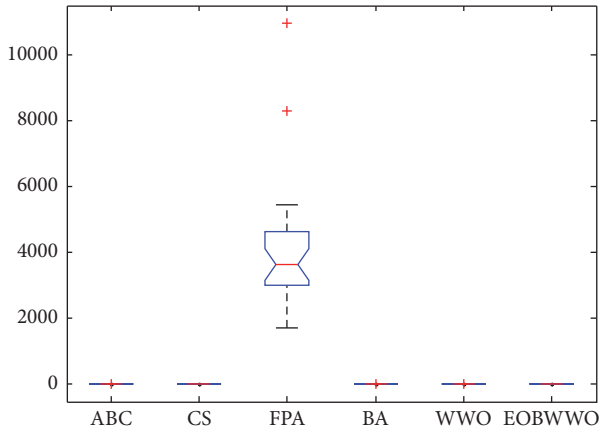

Figure 24: $D = 30$, ANOVA test of global minimum for f_{02}.

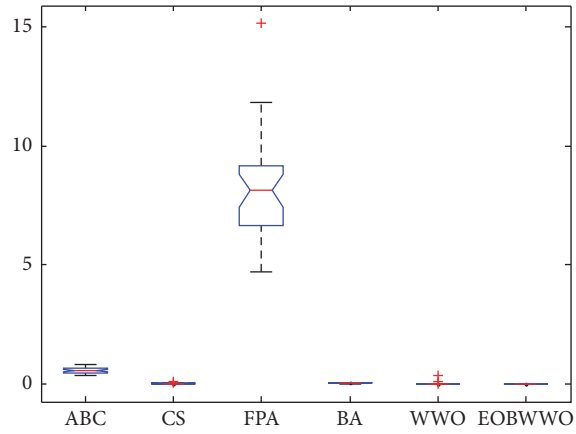

Figure 25: $D = 30$, ANOVA test of global minimum for f_{03}.

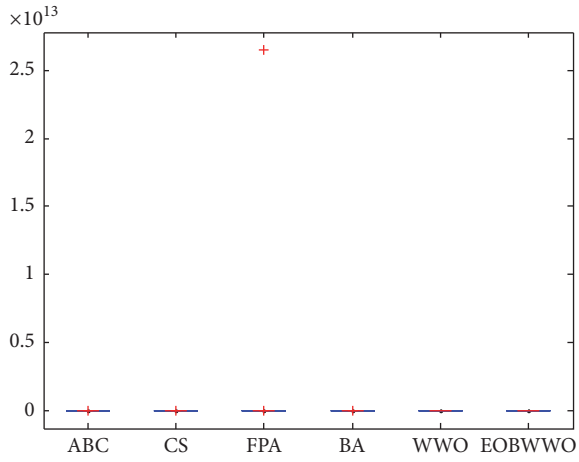

Figure 26: $D = 30$, ANOVA test of global minimum for f_{04}.

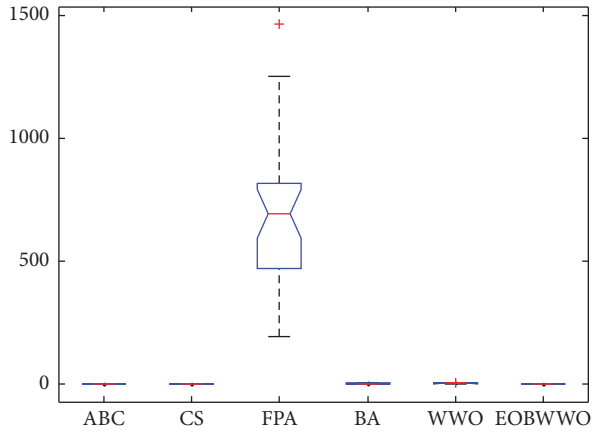

Figure 27: $D = 30$, ANOVA test of global minimum for f_{05}.

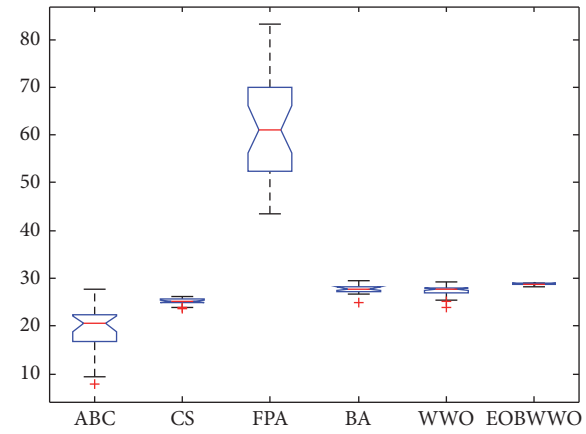

Figure 28: $D = 30$, ANOVA test of global minimum for f_{06}.

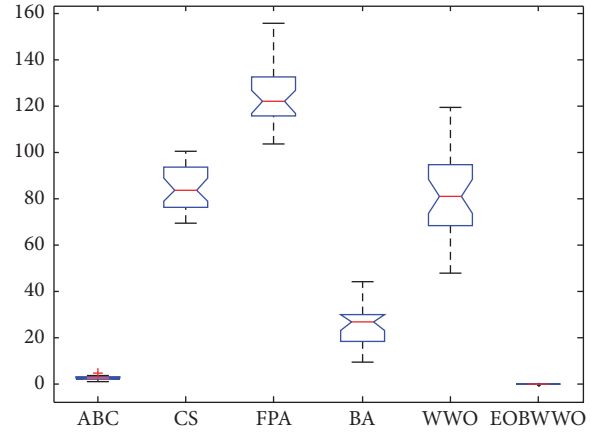

Figure 29: $D = 30$, ANOVA test of global minimum for f_{07}.

calculation precision of WWO algorithm even further for function optimization and structure engineering design problems. The elite opposition-based (EOB) learning strategy enhances global exploration capability by means of increasing population diversity. The local neighborhood search strategy is introduced to enhance local exploitation capability via enhancing the local search around the promising optimal solution. In addition, the improved propagation operator provides the improved algorithm with a better balance between exploration and exploitation. By using the above-mentioned three strategies, EOBWWO can deal with function optimization including multimodal functions and

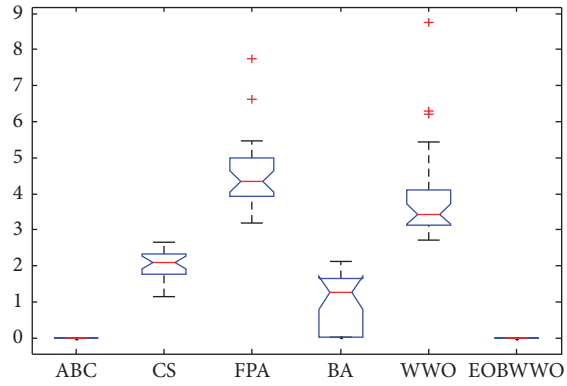

Figure 30: $D = 30$, ANOVA test of global minimum for f_{08}.

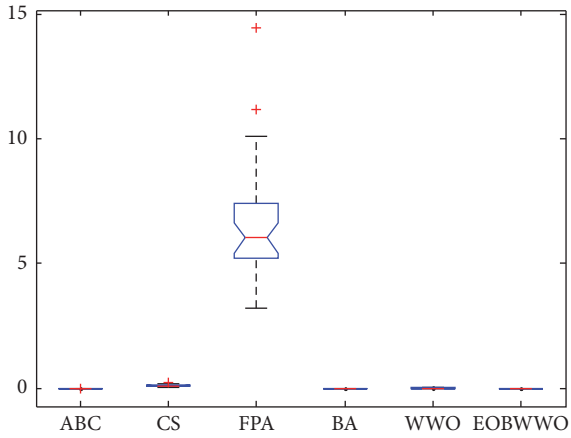

Figure 31: $D = 30$, ANOVA test of global minimum for f_{09}.

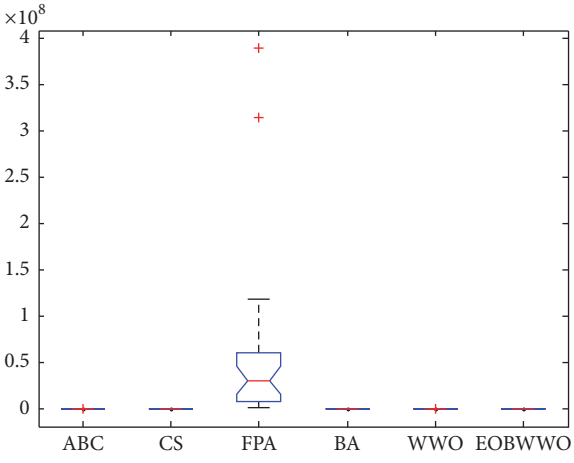

Figure 32: $D = 30$, ANOVA test of global minimum for f_{10}.

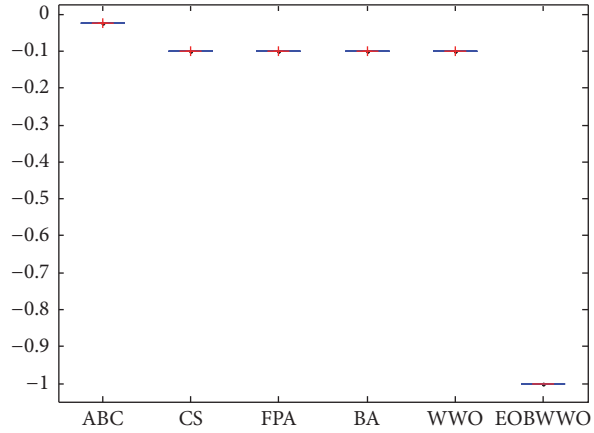

Figure 33: $D = 30$, ANOVA test of global minimum for f_{11}.

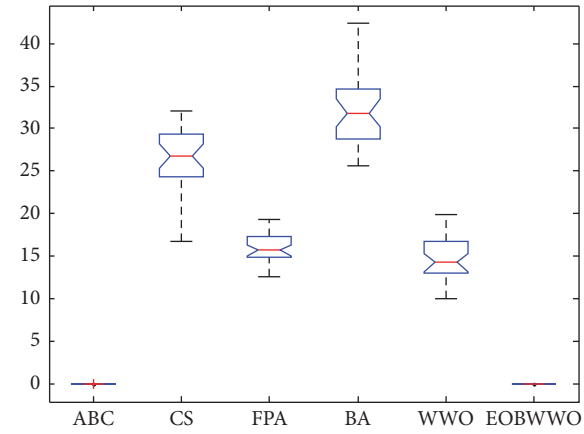

Figure 34: $D = 30$, ANOVA test of global minimum for f_{12}.

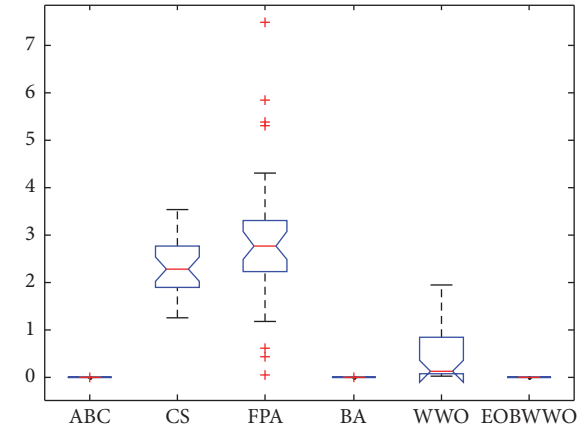

Figure 35: $D = 30$, ANOVA test of global minimum for f_{13}.

structural design problems. The results of 20 benchmark functions and two structural design problems in Section 4 demonstrated that the performance of EOBWWO is better than the comparative algorithms at most benchmark functions and other solving methods for the two structural design problems. The improved EOBWWO algorithm can significantly improve the convergence speed and calculation precision of the original WWO algorithm. There are various important issues for the further research topics of EOBWWO. On the one hand, structural design problems not only exist in the real world widely, but also are generally nonlinear and constrained optimization problems. Therefore,

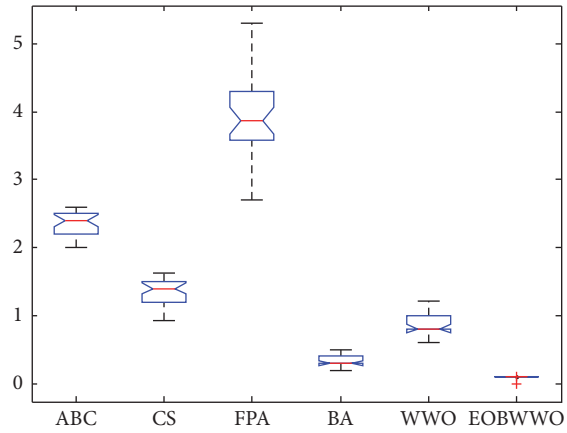

Figure 36: $D = 30$, ANOVA test of global minimum for f_{14}.

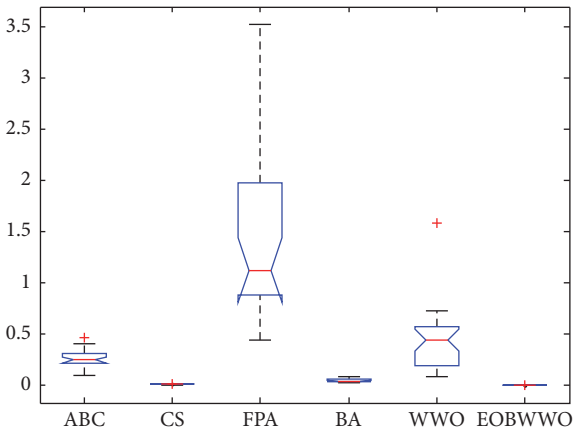

Figure 37: $D = 24$, ANOVA test of global minimum for f_{15}.

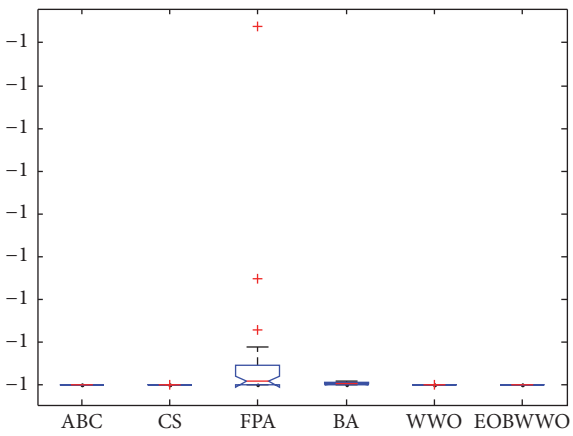

Figure 38: $D = 2$, ANOVA test of global minimum for f_{16}.

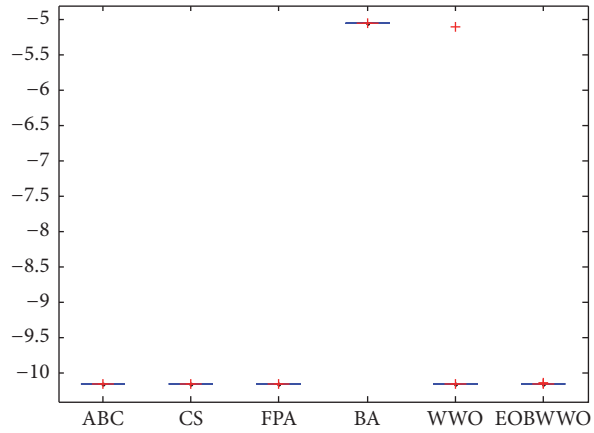

Figure 39: $D = 4$, ANOVA test of global minimum for f_{17}.

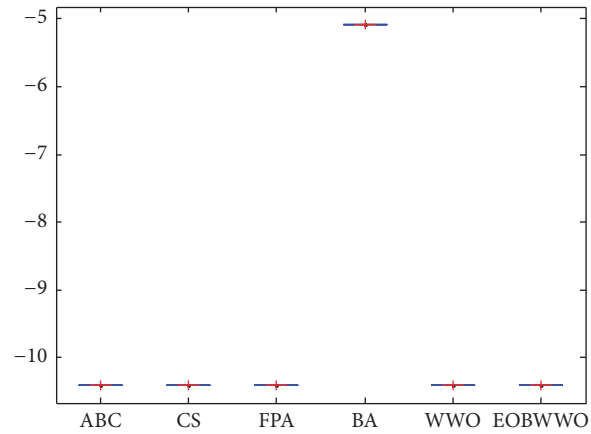

Figure 40: $D = 4$, ANOVA test of global minimum for f_{18}.

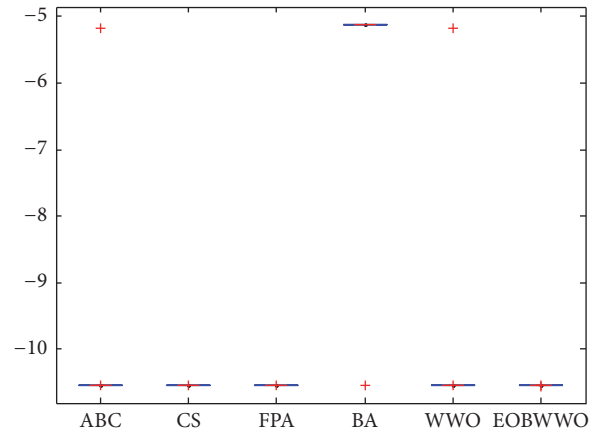

Figure 41: $D = 4$, ANOVA test of global minimum for f_{19}.

other design problems can be resolved using EOBWWO in future research, such as multidimensional knapsack problem, permutation flow shop scheduling problem [51], and graph coloring problem. On the other hand, some improvements can be introduced to EOBWWO and WWO algorithm to enhance the ability of dealing with relevant problems. More elaborate set of parameters, such as breaking coefficient β and wavelength reduction coefficient α, multiple population strategy, and combination with other optimization algorithms are some good choices. In addition, multiobjective optimization problems are also the focus in the future research.

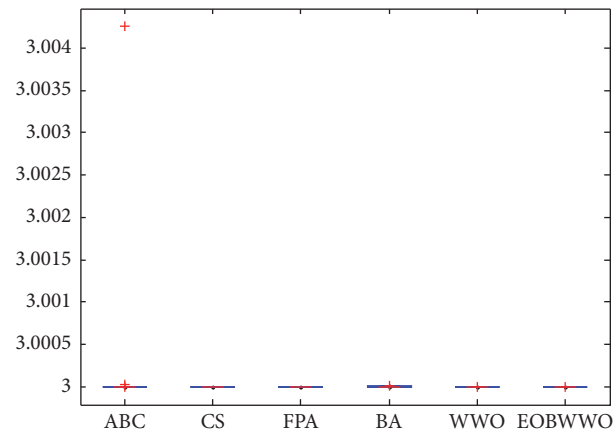

Figure 42: $D = 2$, ANOVA test of global minimum for f_{20}.

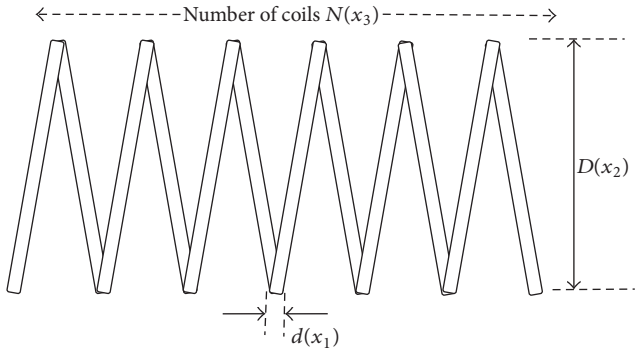

Figure 43: The tension/compression spring problem.

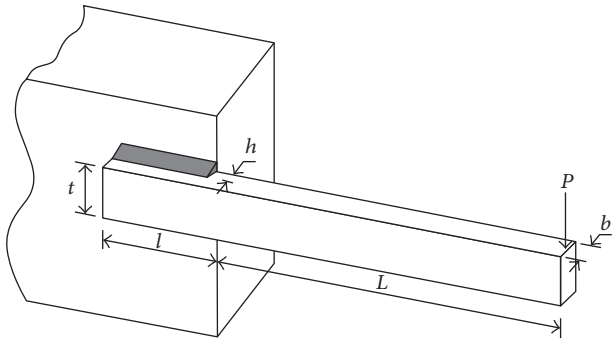

Figure 44: The welded beam problem.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the National Science Foundation of China under Grants nos. 61463007 and 61563008 and the Guangxi Natural Science Foundation under Grant no. 2016GXNSFAA380264.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, November-December 1995.

[2] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligenc*, MIT Press, Cambridge, Mass, USA, 1975.

[3] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[4] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[5] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[6] X. S. Yang, "A new metaheuristic bat-inspired algorithm," *Computer Knowledge & Technology*, vol. 284, pp. 65–74, 2010.

[7] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Engineering with Computers*, vol. 29, no. 2, pp. 175–184, 2013.

[8] X. S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation: 11th International Conference, UCNC 2012, Orléan, France, September 3–7, 2012. Proceedings*, vol. 7445 of *Lecture Notes in Computer Science*, pp. 240–249, Springer, Berlin, Germany, 2012.

[9] Y.-J. Zheng, "Water wave optimization: a new nature-inspired metaheuristic," *Computers & Operations Research*, vol. 55, pp. 1–11, 2015.

[10] H. Huang, *Dynamics of Surface Waves in Coastal Waters*, Springer, Berlin, Germany, 2009.

[11] X.-B. Wu, J. Liao, and Z.-C. Wang, "Water wave optimization for the traveling salesman problem," in *Intelligent Computing Theories and Methodologies: 11th International Conference, ICIC 2015, Fuzhou, China, August 20–23, 2015, Proceedings, Part I*, vol. 9225 of *Lecture Notes in Computer Science*, pp. 137–146, Springer, Berlin, Germany, 2015.

[12] B. Zhang, M.-X. Zhang, J.-F. Zhang, and Y.-J. Zheng, "A water wave optimization algorithm with variable population size and comprehensive learning," in *Intelligent Computing Theories and Methodologies*, vol. 9225 of *Lecture Notes in Computer Science*, pp. 124–136, Springer International, Cham, Switzerland, 2015.

[13] Y. J. Zheng and B. Zhang, "A simplified water wave optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '15)*, Sendai, Japan, May 2015.

[14] A. D. Belegundu, *A Study of Mathematical Programming Methods for Structural Optimization*, Department of Civil and Environmental Engineering, Iowa University, 1982.

[15] J. S. Arora, *Introduction to Optimum Design*, McGraw-Hill, New York, NY, USA, 1989.

[16] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[17] C. A. C. Coello, "Constraint-handling using an evolutionary multiobjective optimization technique," *Civil Engineering & Environmental Systems*, vol. 17, no. 4, pp. 319–346, 2000.

[18] C. A. C. Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Advanced Engineering Informatics*, vol. 16, no. 3, pp. 193–203, 2002.

[19] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Engineering Optimization*, vol. 36, no. 5, pp. 585–605, 2004.

[20] C. A. C. Coello and R. L. Becerra, "Efficient evolutionary optimization through the use of a cultural algorithm," *Engineering Optimization*, vol. 36, no. 2, pp. 219–236, 2004.

[21] K. H. Raj, R. S. Sharma, G. S. Mishra, A. Dua, and C. Patvardhan, "An evolutionary computational technique for constrained optimization in engineering design," *Journal of the Institution of Engineers (India): Mechanical Engineering Division*, vol. 86, pp. 121–128, 2005.

[22] A.-R. Hedar and M. Fukushima, "Derivative-free filter simulated annealing method for constrained continuous global optimization," *Journal of Global Optimization*, vol. 35, no. 4, pp. 521–549, 2006.

[23] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 1, pp. 89–99, 2007.

[24] E. M. Montes and C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *International Journal of General Systems*, vol. 37, no. 4, pp. 443–473, 2008.

[25] M. G. H. Omran and A. Salman, "Constrained optimization using CODEQ," *Chaos, Solitons and Fractals*, vol. 42, no. 2, pp. 662–668, 2009.

[26] V. S. Aragón, S. C. Esquivel, and C. A. Coello Coello, "A modified version of a T-cell Algorithm for constrained optimization problems," *International Journal for Numerical Methods in Engineering*, vol. 84, no. 3, pp. 351–378, 2010.

[27] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1001–1014, 2012.

[28] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing & Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[29] A. H. Gandomi, "Interior search algorithm (ISA): a novel approach for global optimization," *ISA Transactions*, vol. 53, no. 4, pp. 1168–1183, 2014.

[30] A. Baykasoğlu and F. B. Ozsoydan, "Adaptive firefly algorithm with chaos for mechanical design optimization problems," *Applied Soft Computing*, vol. 36, pp. 152–164, 2015.

[31] K. Deb, "Optimal design of a welded beam via genetic algorithms," *Aiaa Journal*, vol. 29, no. 11, pp. 2013–2015, 2013.

[32] J. P. B. Leite and B. H. V. Topping, "Improved genetic operators for structural engineering optimization," *Advances in Engineering Software*, vol. 29, no. 7–9, pp. 529–562, 1998.

[33] C. A. C. Coello, "Self-adaptive penalties for GA-based optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pp. 573–580, July 1999.

[34] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics & Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[35] X. Hu, R. C. Eberhart, and Y. Shi, "Engineering optimization with particle swarm," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 53–57, Indianapolis, Ind, USA, April 2003.

[36] J.-L. Liu, "Novel orthogonal simulated annealing with fractional factorial analysis to solve global optimization problems," *Engineering Optimization*, vol. 37, no. 5, pp. 499–519, 2005.

[37] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[38] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33–40, pp. 3080–3091, 2008.

[39] A. Kaveh and S. Talatahari, "Engineering optimization with hybrid particle swarm and ant colony optimization," *Asian Journal of Civil Engineering*, vol. 10, no. 6, pp. 611–628, 2009.

[40] A. Kaveh and S. Talatahari, "An improved ant colony optimization for constrained engineering design problems," *Engineering Computations*, vol. 27, no. 1, pp. 155–182, 2010.

[41] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Mixed variable structural optimization using Firefly Algorithm," *Computers & Structures*, vol. 89, no. 23-24, pp. 2325–2336, 2011.

[42] X.-Y. Zhou, Z.-J. Wu, H. Wang, K.-S. Li, and H.-Y. Zhang, "Elite opposition-based particle swarm optimization," *Acta Electronica Sinica*, vol. 41, no. 8, pp. 1647–1652, 2013.

[43] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[44] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and the International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 1, pp. 695–701, Vienna, Austria, November 2005.

[45] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, New York, NY, USA, 2010.

[46] Y. Peng and B.-L. Lu, "A hierarchical particle swarm optimizer with latin sampling based memetic algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 5, pp. 2823–2836, 2013.

[47] B. Zhang, H. Yuan, L. Sun, J. Shi, Z. Ma, and L. Zhou, "A two-stage framework for bat algorithm," *Neural Computing & Applications*, 2016.

[48] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.

[49] S. Surjanovic and D. Bingham, "Virtual library of simulation experiments: test functions and datasets," 2013, http://www.sfu.ca/screen.

[50] S. Saremi, S.-Z. Mirjalili, and S.-M. Mirjalili, "Evolutionary population dynamics and grey wolf optimizer," *Neural Computing and Applications*, vol. 26, no. 5, pp. 1257–1263, 2015.

[51] H. Bargaoui and O. B. Driss, "Multi-agent model based on tabu search for the permutation flow shop scheduling problem," *Advances in Distributed Computing & Artificial Intelligence Journal*, vol. 3, no. 8, pp. 519–527, 2014.