Research Article

Biogeography-Based Optimization with Orthogonal Crossover

Quanxi Feng,^{1,2} Sanyang Liu,¹ Guoqiang Tang,² Longquan Yong,¹ and Jianke Zhang³

¹ Department of Applied Mathematics, Xidian University, Xi'an 710071, China

² School of Science, Guilin University of Technology, Guilin 541004, China

³ School of Science, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Correspondence should be addressed to Quanxi Feng; fqx9904@gmail.com and Sanyang Liu; liusanyang@126.com

Received 6 January 2013; Revised 15 April 2013; Accepted 15 April 2013

Academic Editor: Alexander P. Seyranian

Copyright © 2013 Quanxi Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Biogeography-based optimization (BBO) is a new biogeography inspired, population-based algorithm, which mainly uses migration operator to share information among solutions. Similar to crossover operator in genetic algorithm, migration operator is a probabilistic operator and only generates the vertex of a hyperrectangle defined by the emigration and immigration vectors. Therefore, the exploration ability of BBO may be limited. Orthogonal crossover operator with quantization technique (QOX) is based on orthogonal design and can generate representative solution in solution space. In this paper, a BBO variant is presented through embedding the QOX operator in BBO algorithm. Additionally, a modified migration equation is used to improve the population diversity. Several experiments are conducted on 23 benchmark functions. Experimental results show that the proposed algorithm is capable of locating the optimal or closed-to-optimal solution. Comparisons with other variants of BBO algorithms and state-of-the-art orthogonal-based evolutionary algorithms demonstrate that our proposed algorithm possesses faster global convergence rate, high-precision solution, and stronger robustness. Finally, the analysis result of the performance of QOX indicates that QOX plays a key role in the proposed algorithm.

1. Introduction

Many problems in both industrial application field and scientific research world can be regarded as optimization problems. During the past decades, several kinds of classical methods [1, 2] have been proposed to handle optimization problem, which have made enormous progress, but these methods demand to know the property of optimization problem, such as continuity or differentiability. In recent decades, many meta heuristic algorithms sprung up, for example, genetic algorithm (GA) [3], particle swarm optimization (PSO) [4], simulated annealing algorithm (SA) [5], differential evolution (DE) [6], and biogeography-based optimization (BBO) [7]. These algorithms can solve optimization problems without using some information such as differentiability.

Biogeography-based optimization (BBO) [7], developed by Simon, is a new emerging population-based evolutionary algorithm motivated through mimic migration of species in natural biogeography. Just as species migrate back and forth between islands in biogeography, migration operator in BBO shares information between habitats in the population.

That is to say, a good solution can share their features with poor ones through migration operator, and a poor solution can improve their quality by accepting new features from good ones. Thus, BBO possesses powerful exploitation ability. However, migration operator does not produce new SIV which may lead to weak exploration ability and poor population diversity.

In evolution algorithm, the step of generating some solutions can be considered as "experiment." For example, the operation of using crossover operator to sample genes from their parents and produce new offspring, this process can be considered as sample experiment. Orthogonal experiment design is a method that is used to generate multifactor experiment uniformly, which has been developed to sample a small, but representative set of combinations for experimentation. Zhang and Leung [8] introduced orthogonal experimental design into genetic algorithm and proposed orthogonal crossover (OX). Leung and Wang [9] used quantization technique in orthogonal experimental design and proposed quantization orthogonal crossover operator

(QOX). Experimental studies show that QOX is an effective and efficient operator for numerical optimization.

Since BBO has properties of powerful exploitation ability, weak exploration ability, poor population diversity, and slow convergence rate simultaneous. QOX operator is a global search operator with systematic and rational search ability. In order to enhance the exploration ability and improve the population diversity, in this paper, an improved BBO variant, namely, biogeography-based optimization with orthogonal crossover (denoted as OXBBO), is presented for solving global optimization problems. In the proposed algorithm, QOX operator is embedded into BBO to enhance its exploration ability and accelerate its convergence rate, and a modified migration operator is used to improve the population diversity.

The rest of the paper is organized as follows. The basic BBO is introduced in Section 2, while Section 3 briefly reviews orthogonal experiment design. The proposed algorithm is introduced in Section 4 in detail. In Section 5, extensive experiments have been carried out to test OXBBO. Finally, some conclusions and future works are summarized in Section 6.

2. Biogeography-Based Optimization

BBO is a new population-based biogeography inspired global optimization algorithm, which has some features in common with other biology-based algorithms. Migration operator is the main operator of BBO which shares information among solutions. PSO and BBO solutions survive forever, while GA solutions “die” at the end of each generation. PSO solutions are more likely to clump together in similar groups, while BBO and GA solutions do not have any built-in trend to cluster.

BBO is a method motivated by geographical distribution of biological organisms. In BBO, each individual is considered as a “habitat” with a habitat suitability index (HSI). Habitats with a high HSI tend to have a large number of species, while those with a low HSI have a small number of species. Habitats with a high HSI migrate to the nearby habitats, while those with a low HSI accept features from neighbor habitats with high HSI.

Migration strategy in BBO is similar to the global recombination approach in the breeder GA and evolutionary strategy. Migration operator is a probabilistic operator that adjusts habitat based on the immigration rate and emigration rate. The probability that H is modified is proportional to its immigration rate λ , and the probability that the modification source comes from H is proportional to the emigration rate μ . Migration operator [7] can be described in Algorithm 1.

A habitat’s HSI can be changed suddenly because of random events which are modeled as mutation operator in BBO. Mutation operator randomly modifies SIV based on the habitat’s priori probability. The mutation rate m is expressed as follows:

$$m = m_{\max} \left(\frac{1 - p_s}{p_{\max}} \right), \quad (1)$$

```

(1) Begin
(2)   For  $i = 1$  to NP
(3)     Select  $X_i$  based on immigration rate  $\lambda_i$ 
(4)     If  $X_i$  is selected
(5)       For  $j = 1$  to NP
(6)         Select  $X_j$  based on emigration rate  $\mu_j$ 
(7)         If  $\text{rand}(0, 1) < \mu_j$ 
(8)            $H_i(\text{SIV}) = H_j(\text{SIV})$       (*)
(9)         EndIf
(10)      EndFor
(11)    EndIf
(12)  EndFor
(13) End

```

ALGORITHM 1: Procedure of migration operator.

```

(1) Begin
(2)   For  $k = 1$  to Dim
(3)     Use  $\lambda_k$  and  $\mu_k$  to compute the probability  $p_k$ 
(4)     Select variable  $X_{ik}$  with probability  $p_k$ 
(5)     If  $X_{ik}$  is selected
(6)       Replace  $X_{ik}$  with a randomly generated SIV
(7)     EndIf
(8)   EndFor
(9) End

```

ALGORITHM 2: Procedure of mutation operator.

where $m_{\max}$ is a user-defined parameter. In BBO, highly probable solutions will tend to be more dominant in the population without this modification. This mutation scheme tends to increase diversity among population. The mutation operator [7] can be loosely described in Algorithm 2.

As some scholars has pointed out that BBO may have some deficiencies. For example, BBO is good at exploiting the solution space, but BBO is weak at exploring the solution space. Recently, many researchers have been working on the improvement of BBO, and many variants of BBO are presented.

Improving the operator in BBO is one direction for improvement. Gong et al. [11] used a real code to represent the solution and extend BBO to the continuous domain optimization and then presented a real-coded BBO (RCBBO). Ma and Simon [16] modified the migration operator by giving a new migration formula and then proposed blend BBO (B-BBO). Li et al. [10] presented perturbing biogeography-based optimization (PBBO) through a design of perturbing migration based on the sinusoidal migration curve. Li and Yin [12] proposed another migration operation based on multiparent crossover called multiparent migration model and then presented multioperator biogeography-based optimization (MOBBO). Ma [17] generalized linear migration model to six different migration models and then used experimental study and theoretical analysis to investigate their performance.

Hybridization of EAs is another direction for improvement. Cai et al. [18] combined evolutionary programming with BBO to enhance the exploration ability and then proposed hybrid BBO (BBO-EP). Wang and Xu [19] combined differential evolution and simplex search with BBO and proposed SSBODE for solving parameter estimation problems. Lohokare et al. [20] presented intelligent biogeography-based optimization (IBBO) through hybridization of BBO with bacterial foraging algorithm. Zhu [21] used graphics hardware acceleration to present a massively parallel biogeography-based optimization-pattern search algorithm (BBO-PS). Tan and Guo [22] proposed quantum and biogeography-based optimization (QBBO) by evolving multiple quantum probability models via evolutionary strategies inspired by the mathematics of biogeography. Gong et al. [15] combined the exploration of DE with the exploitation of BBO effectively and then presented a hybrid DE with BBO, namely, DE/BBO. Boussaïd et al. [13] proposed a two stages algorithm (DBBO) through hybridization of BBO with DE, which updated the population by using BBO and DE alternately.

Moreover, theory analysis of BBO and application has also been developing. Simon [23] introduced several simplified versions of BBO and used probability theory to perform an approximate analysis. Simon et al. [24] showed that BBO is a generalization of a genetic algorithm with global uniform recombination (GA/GUR) and compared BBO and GA/GUR algorithms by using analytical Markov models. Simon [25] also introduced dynamic system models for BBO and GA/GUR.

Ma et al. [26] incorporated resampling in BBO to solve optimization problems in noisy environments. Ergezer et al. [27] employed opposition-based learning alongside migration rate of BBO and created oppositional BBO (OBBO).

It is necessary to emphasize that our work is to add a new global search operator into BBO algorithm. Meanwhile, we also aim at not increasing too many control parameters in the algorithm.

3. Orthogonal Crossover

In practical application, it is impossible to consider all combinations in large sample space, and then we can use a small part of representative samples to represent the sample space. Orthogonal design method [9], with both orthogonal array (OA) and factor analysis (FA), is used to execute multifactor experiment uniformly and sample well-distributed in solution space. OA is a fractional factorial array of numbers arranged in rows and columns which can assure a balanced comparison of levels of any factor. All columns in orthogonal array can be evaluated independently of one another, a number of such arrays can be found in <http://www2.research.att.com/~njas/oadir/>.

For example, an experiment with 4 factors and 3 levels, its orthogonal array is $L_9 (3^4)$. There are 81 combinations for all. If we apply the orthogonal array, we only consider nine combinations.

For convenience, we denote an orthogonal array for F factors with Q levels and M combinations

as $L_M(Q^F) = [a_{ij}]_{M \times N}$, where $a_{ij} \in \{1, 2, \dots, Q\}$ represents the j th factor in the i th combination. Generally,

$$M \leq \frac{Q^F - 1}{Q - 1}. \quad (2)$$

Orthogonal crossover operator was first developed by Zhang and Leung [8], which originated from the integration of orthogonal experiment design with crossover operator to generate several new solutions in the line with orthogonal array. As a matter of fact, each operation of generating a new offspring can be regarded as an experiment. For example, a crossover is a procedure for sampling several points from a defined region while orthogonal crossover operator (OX) uses orthogonal array to make crossover operator with more statistically. OX in [8] was used in combination problems. Leung and Wang [9] used OX with quantization technique and proposed new OX used for dealing with numerical optimization with continuous variables. Leung quantized the continuous domain into several levels firstly and then used orthogonal array to generate combinations.

Consider two parent habitats $x = (x_1, x_2, \dots, x_N)$ and $y = (y_1, y_2, \dots, y_N)$. x and y define a search space $[\min(x_i, y_i), \max(x_i, y_i)]$ of variable x_i . Then, the solution space is $[l_{\text{low}}, u_{\text{up}}]$, where

$$\begin{aligned} l_{\text{low}} &= [\min(x_1, y_1), \min(x_2, y_2), \dots, \min(x_N, y_N)], \\ u_{\text{up}} &= [\max(x_1, y_1), \max(x_2, y_2), \dots, \max(x_N, y_N)]. \end{aligned} \quad (3)$$

QOX quantizes the solution space $[l_{\text{low}}, u_{\text{up}}]$ into Q levels:

$$\beta_{ij} = \begin{cases} \min(x_1, y_1), & j = 1, \\ \min(x_i, y_i) + (j-1) \left(\frac{|x_i - y_i|}{(Q-1)} \right), & 2 \leq j \leq Q-1, \\ \max(x_N, y_N), & j = Q. \end{cases} \quad (4)$$

In practical application, dimension N is often much larger than orthogonal factor F , and $L_M(Q^F)$ cannot be used directly. So, we first randomly generate $F-1$ different integers between 1 and N and then divide N -dimension solution vector into F factors:

$$\begin{aligned} f_1 &= (x_1, \dots, x_{k_1}), \\ f_2 &= (x_{k_1+1}, \dots, x_{k_2}), \\ &\vdots \\ f_{F-1} &= (x_{k_{F-2}+1}, \dots, x_{k_{F-1}}), \\ f_F &= (x_{k_{F-1}+1}, \dots, x_N). \end{aligned} \quad (5)$$

Finally, M combinations are generated according to orthogonal array $L_M(Q^F)$ whose habitat suitability index is evaluated, and the two best habitats are selected from them to replace their parent habitats.

QOX has been incorporated into particle swarm optimization [4], differential evolution [28], and genetic algorithm [9] to enhance the exploration ability.

FIGURE 1: Illustrations of using crossover to generate new offspring in two-dimension space. (x_1, y_1) , (x_2, y_2) are their parents; the others are offspring. (a) GA crossover operator, (b) migration operator, and (c) orthogonal crossover operator.

- (1) **Begin**
- (2) **Input:** parameters Q and F
- (3) Select M fulfilling formula (2)
- (4) Randomly select two habitats from the population
- (5) Quantize the domain defined by the two habitats into Q levels
- (6) Randomly generate $F - 1$ different integers between 1 and dimension N
- (7) Divide N -dimension habitat vector into F factors
- (8) Apply $L_M(Q^F)$ to generate M potential offspring
- (9) Evaluate M potential offspring
- (10) Select two best habitats to replace their parents
- (11) **Output:** two best habitats
- (12) **End**

ALGORITHM 3: Procedure of QOX.

4. The Proposed Algorithm: OXBBO

4.1. Orthogonal Crossover Operator. Migration operator plays a key role in BBO. Simon et al. [24] has pointed out that BBO migration strategy is conceptually similar to the GA global uniform recombination operator, which can be considered as a special GA crossover operator [24]. Similar to GA crossover operator, migration operator only generates and evaluates the habitat which is located in the vertex of hyperrectangle defined by the emigration habitats and immigration habitats. The inner and other vertexes of the hyperrectangle are not considered in this process, which might be a promising region in the solution space. This process may lead the optimal solution to fall into local optimal solution, especially, in the initial optimizing stage. That is to say, migration operator cannot perform a systematic search in the hyperrectangle. Therefore, the exploration ability of BBO could be limited to some extent.

Orthogonal crossover operator with quantization technique (QOX) samples a small but representative set of combinations for experimentation. In the searching process, QOX not only searches the vertexes of hyperrectangle defined by parents but also searches the inner of the hyperrectangle. So QOX could execute systematic search in the hyperrectangle, which can locate the global optimal solution easy and overcome the limitation of migration operator.

As shown in Figure 1, $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are the parents (supposing that one is emigration habitat and the

other one is immigration habitat in two-dimensional solution space). We can see in Figure 1 that GA crossover operator generates two offspring randomly. Migration operator absorbs better SIVs and generates one offspring, determined by immigration and emigration rate, in the vertex consciously. And orthogonal crossover operator with $L_9(3^4)$ generates nine potential offspring which lie in both the vertexes and the inner of hyperrectangle uniformly. Obviously, orthogonal crossover operator searches the solution space more thoroughly than GA crossover operator and migration operator do.

In order to enhance the exploration ability of BBO, we embedded the QOX operator with $L_9(3^4)$ into BBO. For the sake of randomness, two random selected habitats are used as the parent of QOX, which will be replaced by two best habitats chosen from offspring after performing QOX. Note that QOX operator is more costly than migration operator and GA operator, since it needs to evaluate 9 offspring if $L_9(3^4)$ is used. Considering the limitation of the number of fitness function evaluations, we apply QOX with $L_9(3^4)$ only once at each generation to save the computation cost and to simplify the implementation procedure. The brief description of QOX is shown in Algorithm 3.

4.2. Modified Migration Operator. As analyzed before, in migration operator, the SIV of new habitat migrates from other habitats. This procedure only uses the SIV of better

```

(1) Begin
(2)   Generate initial population Pop randomly
(3)   Evaluate each habitat  $X_i$ 
(4)   While the halting criteria are not satisfied do
(5)     Sort habitats from best to worst based on the fitness values
(6)     Map the fitness value to the number of spires
(7)     Calculate the emigration rate and immigration rate of each habitat
(8)     For  $i = 1$  to NP          /* migration stage */
(9)       Select  $X_i$  based on immigration rate  $\lambda_i$ 
(10)      Generate two different integers  $r1 \neq r2$  randomly between 1 and NP
(11)      If  $X_i$  is selected
(12)        For  $j = 1$  to NP
(13)          Select  $X_j$  based on emigration rate  $\mu_j$ 
(14)          If  $\text{rand}(0, 1) < \mu_j$ 
(15)             $H_i(\text{SIV}) = H_j(\text{SIV}) + \alpha (H_{r1}(\text{SIV}) - H_{r2}(\text{SIV}))$ 
(16)          End If
(17)        End For
(18)      End If
(19)    End For          /* end of migration stage */
(20)    For  $i = 1$  to NP    /* mutation stage */
(21)      Compute the probability  $p_i$ 
(22)      Select  $X_i$  based on  $p_i$ 
(23)      If  $\text{rand}(0, 1) < m_i$ 
(24)         $H_i(\text{SIV}) = H_j(\text{SIV}) + \delta(1)$ 
(25)      End If
(26)    End For          /* end of mutation stage */
(27)    Make sure each habitat legal based on boundary constraints
(28)    Evaluate each newly generated habitat  $M_i$ 
(29)    For  $i = 1$  to NP    /* selection stage */
(30)      If  $f(X_i) < f(M_i)$ 
(31)        Reserve  $X_i$  in the next population
(32)      Else
(33)        Reserve  $M_i$  in the next population
(34)      End If
(35)    End For          /* end of selection stage */
(36)    Input: parameters  $Q$  and  $F$  /* QOX search stage */
(37)    Select  $M$  fulfilling formula (3)
(38)    Randomly select two habitats  $r3 \neq r4$  from the population
(39)    Quantize the domain defined by the two habitats into  $Q$  levels
(40)    Randomly generate  $F - 1$  different integers between 1 and dimension  $N$ 
(41)    Divide  $N$ -dimension habitat vector into  $F$  factors
(42)    Apply  $L_M(Q^F)$  to generate  $M$  potential offspring
(43)    Evaluate  $M$  potential offspring
(44)    Select two best habitats to replace habitat  $r3, r4$ 
(45)    Output: two best habitats /* end of QOX search stage */
(46)  End While
(47) End

```

ALGORITHM 4: Main Procedure of OXBBO.

habitats to replace the SIV of the worse habitats but does not produce new SIV which results in poor population diversity, especially in the later stage. For the sake of this, a new modified migration is used to improve the population diversity. In Algorithm 1, formula (*) is replaced by a new formula:

$$H_i(\text{SIV}) = H_j(\text{SIV}) + \alpha (H_{r1}(\text{SIV}) - H_{r2}(\text{SIV})), \quad (6)$$

where $r1, r2$ are two different integers generated randomly between 1 and NP (NP is population size). We can see

in formula (6), the difference generated by two randomly selected habitats are added to the emigration SIV of habitat, which can generate a random perturbation on the emigration SIV and then improve the diversity of new population.

4.3. Mutation with Mutation Operator. In order to enhance the exploration ability of BBO, a modified mutation operator, Cauchy mutation operator, is integrated into BBO to replace the randomly generated SIV.

The formula for the probability density function of the Cauchy distribution [11] is

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad (7)$$

where $x \in R$, and $t > 0$ is a scale parameter. A real-valued random variable Y is Cauchy distributed with $t > 0$ being written as

$$Y \sim \delta(t). \quad (8)$$

The Cauchy mutation ($t = 1$) in OXBBO can be described as

$$X_{ij}(\text{SIV}) = X_{ij}(\text{SIV}) + \delta(1). \quad (9)$$

4.4. Boundary Constraints. The OXBBO algorithm assumes that all the habitats in the population should be limited in an isolated and finite solution space. There are some habitats moving out of the solution space, in the optimizing process, which should be prevented. In order to keep the solution of bound-constrained problems feasible, those habitats that violate boundary constraints should be replaced by a new generate habitat. The formula is

$$X_{ij}(\text{SIV}) = l_j + \text{rand} * (u_j - l_j), \quad (10)$$

where l_j , u_j are the lower and upper bound, of the solution space, respectively.

4.5. Selection Operator. Selection operator can impel the metabolism of population and retain good habitats. Greedy selection operator is used in this paper. In other words, the population habitat will be replaced by its corresponding trial habitat if the HSI of the trial habitat is better than that of its population habitat:

$$X_i = \begin{cases} X_i & \text{if } f(X_i) < f(M_i) \\ M_i & \text{else,} \end{cases} \quad (11)$$

where X_i , M_i are the population habitat and trial habitat, respectively.

4.6. Main Procedure of OXBBO. As the analysis above, migration operator is good at exploiting the solution space and weak at exploring the solution space. QOX is good at exploring the solution space, which can offset the deficiency of BBO. Modified migration operator can increase the difference of population and improve the population diversity. Therefore, we introduce biogeography-based optimization with orthogonal crossover (viz., OXBBO) by incorporating the above-mentioned QOX and modified migration operator into BBO in this section. The pseudocode of OXBBO approach is described in Algorithm 4.

5. Experimental Study

In order to verify the performance of the proposed algorithm, several experiments are implemented on 23 benchmark

functions. These benchmark functions are chosen from [29] and briefly summarized in Table 1, detailed descriptions about these functions can be found in [29].

Functions f_1 – f_5 are unimodal functions. Where function f_5 has a very sharp narrow ridge running around a parabola. Function f_6 is a discontinuous step function with one minimum. Function f_7 is a low-dimensional function with a noisy perturbation. Functions f_8 – f_{13} are multimodal functions where the number of local minima increases exponentially with dimension. Functions f_{14} – f_{23} are low-dimensional functions with only a few local minima.

5.1. Experimental Setting. For OXBBO, we have chosen a reasonable set of values. For all experimental tests, we use the following parameters setting unless a change is mentioned:

- (1) population size: $NP = 100$;
- (2) maximum immigration rate: $I = 1$;
- (3) maximum emigration rate: $E = 1$;
- (4) mutation probability: $m_{\max} = 0.005$;
- (5) value to reach (VTR) = 10^{-8} , except for f_7 of VTR = 10^{-2} ;
- (6) maximum number of fitness function evaluations (Max_NFFEs): 150,000 for f_1 , f_6 , f_{10} , f_{12} , and f_{13} ; 200,000 for f_2 and f_{11} ; 300,000 for f_7 , f_8 , and f_9 ; 500,000 for f_3 , f_4 , and f_5 ; 10,000 for f_{14} , f_{16} – f_{19} , f_{21} , and f_{22} ; 40,000 for f_{15} ; 20,000 for f_{20} .

All the algorithms are evaluated on performance criteria [14, 15] or similar to those. (1) Successful rate (SR) is The ratio of the number of successful runs to total runs. (2) NEFFs the number of fitness function evaluations is needed when reach VTR. (3) Acceleration rate (AR): $AR = \text{NEFFs}_{\text{other}} / \text{NEFFs}_{\text{OXBBO}}$. AR is used to compare the convergence speed between OXBBO and other algorithms.

Note. For convenience's sake of comparison, the best function values and their standard variation are compared in Section 5.2, and the best function error values and their corresponding standard variation are compared in other sections. Moreover, high-dimensional functions f_1 – f_{13} with dimension 10, 50, 100, and 200 are optimized in Section 5.5. Functions f_1 – f_{13} with 30 dimensions are optimized in other sections.

All experimental tests in this paper are implemented on a computer with 1.86 GHz Inter-core Processor, 1 GB of RAM, and Windows XP operating system in Matlab software 7.6.

5.2. Comparison with Improved BBO Algorithms. In order to validate the performance of OXBBO, we first compare OXBBO with BBO [11] and improved BBO algorithms: Perturb BBO with Gaussian mutation (PBBO-G) [10], real code BBO with Gaussian mutation (RCBBO-G) [11], and Multioperator BBO (MOBBO) [12]. OXBBO executed 50 independent runs on 23 benchmark functions; mean fitness values and their standard variations are recoded. Results of other algorithms are taken from corresponding papers directly. Comparison results are summarized in Table 2.

TABLE 1: Benchmark functions used in our experimental tests.

Function	Name	Dim	Search space	Optima
$f_1(x) = \sum_{i=1}^n x_i^2$	Sphere	n	$[-100, 100]$	0
$f_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	Schwefel's problem 2.22	n	$[10, 10]$	0
$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	Schwefel's problem 1.2	n	$[-100, 100]$	0
$f_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	Schwefel's problem 2.21	n	$[-100, 100]$	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Rosenbrock function	n	$[-30, 30]$	0
$f_6(x) = \sum_{i=1}^n (\lfloor x_i + 0.5 \rfloor)^2$	Step function	n	$[-100, 100]$	0
$f_7(x) = \sum_{i=1}^n i x_i^4 + \text{random}[0, 1]$	Quartic function	n	$[-1.28, 1.28]$	0
$f_8(x) = -\sum_{i=1}^n x_i \sin(\sqrt{ x_i })$	Schwefel function	n	$[-512, 512]$	-12569.5
$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	Rastrigin function	n	$[-5.12, 5.12]$	0
$f_{10}(x) = -20 \exp\left(-0.2 \sqrt{(1/n) \sum_{i=1}^n x_i^2}\right) - \exp((1/n) \sum_{i=1}^n \cos 2\pi x_i) + 20 + e$	Ackley function	n	$[-32, 32]$	0
$f_{11}(x) = (1/4000) \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(x_i/\sqrt{i}) + 1$	Griewank function	n	$[-600, 600]$	0
$f_{12}(x) = (\pi/n) \{10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^n u(x_i, 10, 100, 4),$ $y_i = 1 + (x_i - 1)/4$	Penalized function 1	n	$[-50, 50]$	0
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^2, & x_i > a \\ 0, & -a \leq x_i \leq a \\ k(-x_i - a)^2, & x_i < -a \end{cases}$				
$f_{13}(x) = 0.1 \{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \} + \sum_{i=1}^n u(x_i, 5, 100, 4),$	Penalized function 2	n	$[-50, 50]$	0
$f_{14}(x) = \left[1/500 + \sum_{j=1}^{25} 1/(j + \sum_{i=1}^2 (x_i - a_{ij})^6) \right]^{-1}$	Shekel Foxhole	2	$[-65.536, 65.536]$	1
$f_{15}(x) = \sum_{i=1}^{11} [a_i - (x_1(b_i^2 + b_i x_2)) / (b_i^2 + b_i x_3 + x_4)]^2$	Kowalik's function	4	$[-5, 5]$	0.003075
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-Hump camel-Back function	4	$[-5, 5]$	-1.0316
$f_{17}(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - 1/8\pi) \cos x_1 + 10$	Branin function	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	Goldstein-Price function	2	$[-2, 2]$	3
$f_{19}(x) = -\sum_{i=1}^4 c_i \exp[-\sum_{j=1}^4 a_{ij}(x_j - p_{ij})^2]$	Hartman's function 1	3	$[0, 1]$	-3.86
$f_{20}(x) = -\sum_{i=1}^4 c_i \exp[-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2]$	Hartman's function 2	6	$[0, 1]$	-3.32
$f_{21}(x) = -\sum_{i=1}^5 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's Function 1	4	$[0, 10]$	-10.1532
$f_{22}(x) = -\sum_{i=1}^7 [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's Function 2	4	$[0, 10]$	-10.4029
$f_{23}(x) = -\sum_{i=1}^{10} [(x - a_i)(x - a_i)^T + c_i]^{-1}$	Shekel's Function 3	4	$[0, 10]$	-10.5364

Note: Dim n means these functions with changeable dimension.

TABLE 2: Comparison of improved BBO algorithms and OXBBO.

	PBBO-G [10]	RCBBO-G [11]	BBO [11]	MOBBO [12]	OXBBO
	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)	Mean (Std)
f_1	5.61E-11 (4.92E-11)	1.39E-03 (5.50E-04)	1.63E-03 (6.60E-04)	7.13E-50 (8.47E-50)	3.96E-28 (2.47E-28)
f_2	3.10E-07 (1.80E-07)	7.99E-02 (1.44E-02)	8.04E-02 (1.42E-02)	8.64E-42 (3.69E-42)	1.12E-19 (4.01E-20)
f_3	5.93E-02 (1.84E-01)	2.27E+01 (1.03E+01)	4.80E+01 (2.19E+01)	2.00E-03 (3.30E-03)	3.21E-03 (1.96E-03)
f_4	1.28E-02 (2.00E-03)	3.09E-02 (7.27E-03)	2.68E-02 (5.09E-03)	1.67E-03 (7.00E-03)	2.61E-15 (8.83E-15)
f_5	7.55E+00 (1.02E+01)	5.54E+01 (3.52E+01)	5.27E+01 (3.91E+01)	2.10E+01 (6.85E+00)	1.89E+01 (1.84E-01)
f_6	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
f_7	1.16E-02 (3.9E-03)	1.75E-02 (6.43E-03)	1.87E-02 (5.11E-03)	3.49E-03 (6.30E-03)	2.01E-03 (6.45E-04)
f_8	-12569 (4.168E-12)	-12569.5 (2.20E-05)	-12569.5 (1.93E-05)	-12569.5 (1.05E-12)	-12569.5 (0.00E+00)
f_9	1.74E-11 (1.12E-11)	2.62E-02 (9.76E-03)	2.77E-02 (1.02E-02)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
f_{10}	3.35E-06 (2.31E-06)	2.51E-02 (5.51E-03)	2.89E-02 (5.15E-03)	6.22E-15 (0.00E+00)	6.79E-15 (1.50E-15)
f_{11}	8.40E-03 (1.49E-03)	8.49E-02 (5.44E-02)	2.99E-02 (3.20E-02)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
f_{12}	7.81E-14 (1.20E-13)	3.28E-05 (3.33E-05)	2.73E-05 (2.49E-05)	1.57E-32 (2.89E-48)	7.06E-30 (3.95E-30)
f_{13}	5.33E-12 (9.44E-12)	3.72E-04 (4.63E-04)	5.84E-04 (8.82E-04)	1.42E-32 (1.56E-33)	7.45E-29 (3.00E-29)
f_{14}	0.9980 (3.67E-11)	0.998017 (5.23E-05)	0.998 (2.74E-05)	0.998 (1.96E-16)	0.998004 (2.34E-16)
f_{15}	7.28E-04 (1.94E-04)	7.86E-04 (1.80E-04)	9.00E-04 (2.68E-04)	7.26E-04 (7.39E-05)	6.29E-04 (4.79E-05)
f_{16}	-1.0316 (5.92E-07)	-1.03101 (9.01E-04)	-1.03095 (1.09E-03)	-1.0316 (7.17E-14)	-1.03163 (4.52E-12)
f_{17}	0.3979 (1.03E-06)	0.398414 (6.77E-04)	0.398327 (4.26E-04)	0.3979 (0)	0.397888 (2.41E-06)
f_{18}	3.0000 (1.76E-08)	3.009504 (1.12E-02)	3.007858 (9.57E-03)	3.000 (9.87E-15)	3.000 (8.30E-15)
f_{19}	-3.8628 (7.30E-09)	-3.86248 (3.65E-04)	-3.86253 (2.62E-04)	-3.8628 (4.19E-16)	-3.86278 (2.40E-15)
f_{20}	-3.2744 (0.0614)	-3.31691 (2.36E-02)	-3.30741 (3.90E-02)	-3.3101 (0.0376)	-3.31962 (0.0168)
f_{21}	-6.6284 (3.1750)	-5.51341 (3.35E+00)	-4.49193 (3.34E+00)	-6.6598 (3.7522)	-7.69866 (3.011)
f_{22}	-7.4888 (3.8490)	-6.80022 (3.52E+00)	-6.73583 (3.40E+00)	-9.8714 (1.6808)	-10.3598 (0.3048)
f_{23}	-6.3403 (3.7343)	-7.28480 (3.38E+00)	-7.80261 (3.29E+00)	-9.8663 (2.119)	-10.4040 (0.810)

Note: according to the setting, these algorithms use the same maximum generations for each function, but population size of MOBBO is $65 * 3 = 195$ while that of other algorithms is 100, so the MAX_NFFE of MOBBO are twice those of the others.

We make a comparison of OXBBO with original BBO firstly. In Table 2, the results of OXBBO are better than those of BBO in 22 out of 23 benchmark functions except f_6 . Either higher-dimensional functions or lower-dimensional functions, OXBBO outperforms BBO. This means that the integration of QOX and modified operator in BBO can improve the performance greatly.

Compared with PBBO-G, from Table 2, we can see that OXBBO is better than PBBO-G on 21 out of 23 functions. For unimodal function f_1 – f_6 , it is obvious that OXBBO performs better than or similar to PBBO-G on 5 out of 6 functions, and PBBO-G is better than OXBBO only on function f_5 . For higher-dimensional multimodal functions, OXBBO performs better than PBBO-G for all the multimodal functions. For lower dimension functions, OXBBO outperforms PBBO-G except function f_{17} , and both algorithms obtain similar results on function f_{17} . Moreover, the results of OXBBO are better than those of PBBO-G by several orders of magnitude on functions f_1 , f_2 , f_4 , f_8 , f_9 , f_{10} , f_{11} , f_{12} , f_{13} , f_{14} , f_{16} , f_{18} , and f_{19} . The analysis shows that the combination of QOX and modified migration operator into BBO is much better than combination of perturbing migration and sinusoidal migration curve in BBO.

What is more is that similar results are obtained when we compare OXBBO with RCBBO-G. OXBBO outperforms RCBBO-G in 22 test functions, and OXBBO, RCBBO-G

obtain the same results on test function f_6 . When compared with MOBBO, OXBBO is better than or similar to MOBBO in 12 functions out of 23 functions. According to the parameters setting, MOBBO and OXBBO are tested with population size ($65 * 3 = 195$ and 100; both of the two algorithms are tested with the same maximum generations. That is to say, the maximum number of fitness function evaluations of MOBBO is almost twice as that of OXBBO. Meanwhile, MOBBO is better than OXBBO in precision in some functions that excellent results are also obtained by OXBBO. The above analysis indicates that OXBBO not only has better exploitation ability but also has better exploration ability.

5.3. Comparison with Hybrid BBO Algorithms. In order to verify the performance of OXBBO further, comparisons of OXBBO with hybrid BBO algorithms, such as BBO-EP [11], DBBO [13], eBBO [14], and DE/BBO [15], are made. The algorithms we adopt in this section to conduct the comparison are to directly cite their numerical results available in the according papers. OXBBO are executed 50 independent runs on 13 high-dimensional functions. The numerical results of 13 high-dimensional benchmark functions are listed in Table 3.

From Table 3, OXBBO outperforms BBO-EP, DBBO, eBBO, and DE/BBO in 9, 9, 8, and 6 out of 13 test functions, respectively. In addition, BBO-EP, DBBO, eBBO, and DE/BBO surpass OXBBO in 3, 3, 3, and 3 out of 13 test

TABLE 3: Comparison of hybrid BBO algorithms and OXBBO for high-dimension of functions.

	BBO-EP [11]	DBBO [13]	eBBO [14]	DE/BBO [15]	OXBBO
	Mean (Std)	Mean (Std)	Mean (std)	Mean (std)	Mean (Std)
f_1	$2.35E-06$ ($3.17E-06$)	$7.76E-36$ ($3.80E-35$)	$6.22E-27$ ($2.17E-26$)	$8.66E-28$ ($5.21E-28$)	$3.96E-28$ ($2.47E-28$)
f_2	$5.03E-07$ ($3.17E-08$)	$6.14E-04$ ($2.64E-03$)	$4.46E-23$ ($1.28E-22$)	$0.00E+00$ ($0.00E+00$)	$1.12E-19$ ($4.01E-20$)
f_3	$1.59E-03$ ($2.90E-03$)	$1.52E-11$ ($1.86E-11$)	$4.98E+01$ ($8.04E+01$)	$2.26E-03$ ($1.58E-03$)	$3.21E-03$ ($1.96E-03$)
f_4	$6.66E-04$ ($4.62E-05$)	$9.02E-02$ ($6.10E-02$)	$7.68E-08$ ($1.37E-07$)	$1.89E-15$ ($8.85E-16$)	$2.61E-15$ ($8.83E-15$)
f_5	$2.42E+00$ ($2.60E+00$)	$6.71E+01$ ($2.61E+01$)	$5.16E+01$ ($4.37E+01$)	$1.90E+01$ ($7.52E+00$)	$1.89E+01$ ($1.84E-01$)
f_6	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)
f_7	$1.70E-03$ ($5.90E-04$)	$1.54E-03$ ($3.41E-04$)	$1.88E-03$ ($3.60E-03$)	$3.44E-03$ ($8.27E-04$)	$2.01E-03$ ($6.45E-04$)
f_8	-12569.5 ($2.65E-06$)	$2.10E-01$ ($2.91E-01$)	$8.13E-09$ ($2.21E-14$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)
f_9	$3.97E-04$ ($5.32E-04$)	$2.24E-03$ ($3.25E-03$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)
f_{10}	$9.40E-04$ ($5.84E-05$)	$4.63E-03$ ($1.47E-02$)	$1.45E-14$ ($9.93E-15$)	$1.07E-14$ ($1.90E-15$)	$6.79E-15$ ($1.50E-15$)
f_{11}	$6.70E-08$ ($9.55E-09$)	$1.33E-17$ ($6.53E-17$)	$6.57E-03$ ($9.00E-03$)	$0.00E+00$ ($0.00E+00$)	$0.00E+00$ ($0.00E+00$)
f_{12}	$1.57E-08$ ($2.40E-09$)	$9.64E-06$ ($3.68E-05$)	$3.75E-32$ ($6.58E-32$)	$7.16E-29$ ($6.30E-29$)	$7.06E-30$ ($3.95E-30$)
f_{13}	$2.52E-07$ ($1.10E-07$)	$4.51E-04$ ($2.15E-03$)	$1.08E-26$ ($2.76E-26$)	$9.81E-27$ ($7.10E-27$)	$7.45E-29$ ($3.00E-29$)

Note: the result of f_8 obtained by BBO-EP is fitness value but is fitness value error by the others.

TABLE 4: NFFEs required to obtain accuracy levels less than VTR.

	eBBO		DE/BBO		OXBBO		3 versus 1	3 versus 2
	Mean (std)	SR	Mean (std)	SR	Mean (std)	SR	AR	AR
f_1	56110 (3559.251)	100%	59926 (745.5)	100%	44398 (438.77)	100%	1.26	1.35
f_2	73670 (4640.076)	100%	82004 (983.9)	100%	84490 (1085.61)	100%	0.87	0.97
f_3	NA (NA)	0	NA (NA)	0	NA (NA)	0	NA	NA
f_4	491498 (14486.144)	44%	296572 (4969.9)	100%	214889 (4628.80)	100%	2.29	1.38
f_5	NA (NA)	0	NA (NA)	0	NA (NA)	0	NA	NA
f_6	17138 (2141.132)	100%	21590 (573.3)	100%	16055 (545.62)	100%	1.07	1.34
f_7	43864 (39947.48)	100%	109574 (21005.8)	100%	58173 (19577.82)	100%	0.75	1.88
f_8	59948 (2831.762)	100%	95952 (3126.7)	100%	63213 (1639.74)	100%	0.95	1.52
f_9	72220 (16458.023)	100%	170226 (8379.0)	100%	124639 (5195.17)	100%	0.58	1.37
f_{10}	84870 (4800.818)	100%	91308 (922.7)	100%	78839 (853.99)	100%	1.08	1.16
f_{11}	120422 (71312.35)	54%	62042 (1219.6)	100%	46312 (798.55)	100%	2.6	1.3
f_{12}	45416 (2520.314)	100%	54482 (873.3)	100%	40646 (641.83)	100%	1.19	1.34
f_{13}	54302 (2238.94)	100%	64772 (1133.4)	100%	43970 (605.58)	100%	1.23	1.47
Average	$1.02E+05$ ($1.50E+04$)	77%	$1.01E+05$ ($3.99E+03$)	85%	$7.41E+04$ ($3.27E+03$)	85%	1.26	1.37

functions, respectively. Thus, we can conclude that, overall, OXBBO is better than BBO-EP, DBBO, and eBBO and is very competitive with DE/BBO. The above discussion means that OXBBO is an effective global function optimizer.

5.4. Convergence Speed When Compared with BBO Algorithms. From Section 5.3, we can see that, though OXBBO outperforms four algorithms, the results of eBBO, DE/BBO are also very competitive. According to [30], more than three quarters of the computational cost is consumed by the fitness function evaluation in evolutionary search. Hence, in solving real-world problems, NFFEs overwhelm the algorithm overhead. For these reasons, we compare the convergence speed of OXBBO with eBBO and DE/BBO by using the concept acceleration rate (AR). From the definition of AR, we know that $AR > 1$ means that OXBBO is faster than the compared algorithm, and vice versa.

Table 4 summarizes the mean NFFEs and SR of eBBO, DE/BBO, and OXBBO for solving 13 test functions, and ARs between algorithm OXBBO and eBBO, OXBBO, and DE/BBO.

OXBBO requires fewer NFFEs to reach the VTR than DE/BBO does on 10 functions out of 11 functions with a successful run, which can indicate that our algorithm is faster than DE/BBO. Similar results can be obtained when compared with eBBO. For example, tests on function f_1 show that the average numbers of NFFEs of 56110, 59926, and 44398 are needed by the eBBO, DE/BBO, and OXBBO, respectively, to reach the VTR. However, OXBBO only needs 44398 NFFEs, which means that its CPU time is the shortest among the three algorithms. Moreover, it can be seen from Table 4 that the average AR between algorithms OXBBO and eBBO, OXBBO, and DE/BBO in the successful run are 1.26 and 1.37 (the total AR divided by 11); that is to say, the total

TABLE 5: Comparison of DE/BBO and OXBBO with differential dimensions.

	DE/BBO		SR	OXBBO		SR	DE/BBO		SR	OXBBO		SR
	Mean (Std)	Dim = 10		Mean (Std)	Dim = 10		Mean (Std)	Dim = 50		Mean (Std)	Dim = 50	
f_1	0.00E + 00 (0.00E + 00)	100%	100%	3.71E - 42 (3.49E - 42)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	5.45E - 64 (3.30E - 64)	100%	
f_2	0.00E + 00 (0.00E + 00)	100%	100%	4.67E - 23 (2.37E - 23)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	2.07E - 33 (7.28E - 34)	100%	
f_3	6.07E - 14 (9.60E - 14)	100%	100%	3.12E - 12 (3.65E - 12)	100%	100%	5.20E + 02 (2.48E + 02)	0	0	2.61E + 01 (7.33E + 00)	0	
f_4	1.58E - 16 (9.88E - 17)	100%	100%	3.62E - 13 (1.48E - 13)	100%	100%	3.42E - 03 (2.28E - 02)	6%	6%	2.82E - 04 (8.21E - 04)	0	
f_5	3.40E + 00 (8.03E - 01)	0	0	5.85E + 00 (6.70E - 01)	0	0	4.43E + 01 (1.39E + 01)	0	0	3.92E + 01 (2.20E - 01)	0	
f_6	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	
f_7	1.11E - 03 (3.96E - 04)	100%	100%	1.31E - 03 (6.90E - 04)	100%	100%	4.05E - 03 (9.20E - 04)	100%	100%	2.13E - 03 (5.79E - 04)	100%	
f_8	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	2.37E + 00 (1.67E + 01)	98%	98%	1.82E - 11 (0.00E + 00)	100%	
f_9	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	1.52E - 03 (1.01E - 02)	96%	
f_{10}	5.89E - 16 (0.00E - 00)	100%	100%	2.66E - 15 (0.00E + 00)	100%	100%	5.92E - 15 (1.79E - 15)	100%	100%	5.01E - 15 (1.70E - 15)	100%	
f_{11}	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	
f_{12}	4.71E - 32 (0.00E + 00)	100%	100%	4.71E - 32 (1.66E - 47)	100%	100%	9.42E - 33 (0.00E + 00)	100%	100%	9.42E - 33 (1.38E - 48)	100%	
f_{13}	1.35E - 32 (0.00E + 00)	100%	100%	1.35E - 32 (1.11E - 47)	100%	100%	1.35E - 32 (0.00E + 00)	100%	100%	1.35E - 32 (1.11E - 47)	100%	
Dim = 100												
f_1	6.16E - 34 (1.97E - 33)	100%	100%	1.48E - 80 (1.00E - 80)	100%	100%	3.07E - 32 (2.48E - 32)	100%	100%	2.87E - 95 (2.24E - 95)	100%	
f_2	0.00E + 00 (0.00E + 00)	100%	100%	1.91E - 42 (6.31E - 43)	100%	100%	5.83E - 17 (8.11E - 17)	100%	100%	4.64E - 54 (1.75E - 54)	100%	
f_3	3.10E + 04 (1.12E + 04)	0	0	3.04E + 02 (5.06E + 01)	0	0	2.22E + 05 (5.90E + 04)	0	0	1.31E + 03 (1.68E + 02)	0	
f_4	2.71E + 00 (2.58E + 00)	0	0	2.90E + 00 (8.10E - 01)	0	0	1.59E + 01 (3.43E + 00)	0	0	1.25E + 01 (9.66E - 01)	0	
f_5	1.19E + 02 (3.38E + 01)	0	0	8.15E + 01 (8.52E - 01)	0	0	2.95E + 02 (4.48E + 01)	0	0	1.80E + 02 (2.47E + 01)	0	
f_6	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	3.60E - 01 (8.60E - 01)	76%	
f_7	6.87E - 03 (1.15E - 03)	98%	98%	2.76E - 03 (6.30E - 04)	100%	100%	1.56E - 02 (2.82E - 03)	0	0	7.17E - 03 (1.01E - 03)	100%	
f_8	7.11E + 00 (2.84E + 01)	94%	94%	1.09E - 10 (0.00E + 00)	100%	100%	2.01E + 02 (1.68E + 02)	46%	46%	2.91E - 10 (0.00E + 00)	100%	
f_9	7.36E - 01 (8.48E - 01)	46%	46%	7.37E + 00 (3.79E + 00)	100%	0	1.76E + 01 (2.89E + 00)	0	0	5.21E + 01 (1.02E + 01)	0	
f_{10}	7.84E - 15 (7.03E - 16)	100%	100%	9.06E - 15 (3.29E - 15)	100%	100%	1.09E - 14 (1.12E - 15)	100%	100%	2.47E - 14 (3.84E - 15)	100%	
f_{11}	0.00E + 00 (0.00E + 00)	100%	100%	0.00E + 00 (0.00E + 00)	100%	100%	1.11E - 16 (0.00E + 00)	100%	100%	1.20E - 16 (5.48E - 17)	100%	
f_{12}	4.71E - 33 (0.00E + 00)	100%	100%	4.84E - 33 (8.76E - 34)	100%	100%	2.36E - 33 (0.00E + 00)	100%	100%	2.94E - 33 (8.35E - 34)	100%	
f_{13}	1.76E - 32 (2.68E - 32)	100%	100%	1.39E - 32 (2.79E - 33)	100%	100%	8.36E - 32 (1.44E - 31)	100%	100%	4.37E - 32 (2.52E - 32)	100%	

TABLE 6: Comparison of OX algorithms.

	OXDE		OLPSO		OXBBO	
	Mean (Std)	SR	Mean (Std)	SR	Mean (Std)	SR
f_1	$1.58E - 16$ ($1.41E - 16$) ⁺	100%	$1.55E - 41$ ($3.32E - 41$) ⁺	100%	$8.84E - 64$ ($8.92E - 64$)	100%
f_2	$4.38E - 12$ ($1.93E - 12$) ⁺	100%	$5.76E - 31$ ($5.46E - 31$) ⁺	100%	$5.25E - 46$ ($3.50E - 46$)	100%
f_3	$6.41E - 07$ ($4.98E - 07$) ⁺	0%	$2.17E - 04$ ($2.72E - 04$) ⁺	0%	$1.81E - 11$ ($1.77E - 11$)	100%
f_4	$1.49E + 00$ ($9.62E - 01$) ⁺	0%	$2.81E + 00$ ($1.47E + 00$) ⁺	0%	$2.39E - 01$ ($2.58E - 01$)	0%
f_5	$1.59E - 01$ ($7.97E - 01$) ⁻	96%	$4.78E - 01$ ($1.32E + 00$) ⁻	88%	$2.61E + 01$ ($9.85E - 01$)	0%
f_6	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%
f_7	$2.95E - 03$ ($1.32E - 03$) ⁺	100%	$1.35E - 03$ ($1.40E - 03$)	100%	$9.41E - 04$ ($3.76E - 04$)	100%
f_8	$0.00E + 00$ ($0.00E + 00$)	100%	$2.84E + 01$ ($6.19E + 01$) ⁺	80%	$0.00E + 00$ ($0.00E + 00$)	100%
f_9	$4.06E + 00$ ($1.95E + 00$) ⁺	0%	$1.99E - 01$ ($4.06E - 01$)	80%	$1.23E - 01$ ($4.05E - 01$)	84%
f_{10}	$2.99E - 09$ ($1.54E - 09$) ⁺	100%	$3.52E - 15$ ($1.55E - 15$) ⁺	100%	$2.66E - 15$ ($0.00E + 00$)	100%
f_{11}	$1.48E - 03$ ($3.02E - 03$) ⁺	80%	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%
f_{12}	$6.94E - 18$ ($7.55E - 18$) ⁺	100%	$1.57E - 32$ ($5.59E - 48$)	100%	$1.57E - 32$ ($5.59E - 48$)	100%
f_{13}	$4.07E - 17$ ($2.68E - 17$) ⁺	100%	$1.35E - 32$ ($5.59E - 48$)	100%	$1.35E - 32$ ($5.59E - 48$)	100%
f_{14}	$1.24E - 16$ ($1.12E - 16$) ⁺	100%	$3.64E - 16$ ($3.95E - 16$) ⁺	100%	$8.88E - 18$ ($4.44E - 17$)	100%
f_{15}	$5.12E - 19$ ($1.49E - 19$) ⁻	100%	$1.56E - 04$ ($1.29E - 04$) ⁻	0%	$3.35E - 04$ ($6.74E - 05$)	0%
f_{16}	$1.81E - 15$ ($6.04E - 15$)	100%	$4.44E - 16$ ($0.00E + 00$) ⁺	100%	$3.46E - 16$ ($1.12E - 16$)	100%
f_{17}	$1.62E - 11$ ($7.29E - 11$)	100%	$1.67E - 16$ ($0.00E + 00$)	100%	$1.67E - 16$ ($0.00E + 00$)	100%
f_{18}	$4.97E - 15$ ($1.65E - 15$) ⁺	100%	$9.02E - 15$ ($4.37E - 15$) ⁺	100%	$2.91E - 15$ ($1.35E - 15$)	100%
f_{19}	$4.44E - 16$ ($0.00E + 00$)	100%	$3.59E - 14$ ($5.28E - 14$) ⁺	100%	$0.00E + 00$ ($0.00E + 00$)	100%
f_{20}	$9.51E - 03$ ($3.29E - 02$)	92%	$6.47E - 08$ ($3.23E - 07$)	96%	$9.51E - 03$ ($3.29E - 02$)	92%
f_{21}	$2.03E - 01$ ($2.77E - 01$) ⁻	0%	$7.79E - 02$ ($2.25E - 01$) ⁻	0%	$2.34E + 00$ ($3.07E + 00$)	24%
f_{22}	$1.68E - 07$ ($2.20E - 07$)	12%	$2.71E - 01$ ($1.33E + 00$)	0%	$2.67E - 01$ ($1.34E + 00$)	92%
f_{23}	$2.78E - 07$ ($4.58E - 07$)	16%	$3.67E - 03$ ($1.43E - 02$)	0%	$2.94E - 01$ ($1.14E + 00$)	92%

⁺-The value of t with 48 degrees of freedom is significant at 0.05 confidence level by two-tailed test.

⁻The corresponding algorithm is better than our proposed OXBBO method.

convergence rate of OXBBO is faster than that of DE/BBO and eBBO.

From Table 4, we can also find that the average SR of eBBO, DE/BBO, and OXBBO are 76.8%, 84.6%, and 84.6%, respectively. OXBBO and DE/BBO reach the VTR with a successful ratio of 100% on 11 functions except function f_3 and f_5 , while eBBO on 9 functions. So, OXBBO and DE/BBO are more robust than eBBO.

5.5. Effect of Dimensionality on the Performance. From the analysis above, OXBBO is a BBO algorithm with robust effective performance. In order to investigate the influence of scalability on the performance of OXBBO, we carry out a scalability study comparison with DE/BBO for the scalable functions. For functions f_1 – f_{13} , Dim = 10, 50, 100, and 200. The results of 50 independent runs are recorded in Table 5 after Dim $\times$ 10000 NFFEs.

We can find, from Table 5, that the overall SR is decreasing for both DE/BBO and OXBBO upon increasing the dimension. For OXBBO, the average SR for solving function with dimension 10, 50, 100, and 200 is 92%, 77%, 69%, and 67%, respectively, whereas the SR of DE/BBO for solving these functions with the same dimension is 92%, 77%, 72%, and 57%, respectively. Obviously, the average SR of DE/BBO decreases more than that of OXBBO does upon increasing dimension, which can also be proved when we carefully

compare the mean fitness values accordingly. By carefully looking at the results, we can recognize that for unimodal function f_1 , DE/BBO is better than OXBBO at dimensions 10 and 50. However, DE/BBO is outperformed by OXBBO at high dimensions (100 and 200). Moreover, the precision of f_1 obtained by DE/BBO is decreased very much upon increasing dimension, while that obtained by OXBBO is increased upon increasing dimension. Another example, for multimodal function f_8 , which is very difficult to locate its global optima position, OXBBO can find its global solution at different dimensions in Table 5, but DE/BBO cannot do it. The experimental test shows that OXBBO can also solve higher dimensionality function effectively. Therefore, we can know that the operators used in OXBBO can enhance the exploration ability but do not offset its exploitation ability.

5.6. Comparison with Other OX-Based Algorithms. In this section, we compare the performance of OXBBO with two excellent OX-based algorithms: OXDE [28], which is proposed by embedding QOX into DE, and OLPSO [4], which is presented through using an orthogonal learning strategy for PSO to overcome the “oscillation” phenomenon of traditional PSO.

Now, we apply OX-based algorithms to solve 13 test functions adopted in this paper under 25 independent runs.

TABLE 7: Comparison of OX operator.

	OXBBO		OXBBO1		OXBBO versus OXBBO1
	Mean (Std)	SR	Mean (Std)	SR	<i>t</i> -test
f_1	$3.10E - 28$ ($1.31E - 28$)	100%	$1.22E - 19$ ($3.73E - 20$)	100%	-3.274
f_2	$1.28E - 19$ ($4.05E - 20$)	100%	$7.50E - 14$ ($1.64E - 14$)	100%	-4.564
f_3	$3.23E - 03$ ($1.92E - 03$)	0%	$1.28E + 02$ ($2.44E + 01$)	0%	-5.251
f_4	$6.66E - 16$ ($1.32E - 15$)	100%	$3.01E - 08$ ($9.00E - 09$)	0%	-3.346
f_5	$1.89E + 01$ ($2.07E - 01$)	0%	$1.81E + 01$ ($2.35E - 01$)	0%	2.831
f_6	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%	NaN
f_7	$1.80E - 03$ ($5.54E - 04$)	100%	$4.63E - 03$ ($1.59E - 03$)	100%	-1.677
f_8	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%	NaN
f_9	$0.00E + 00$ ($0.00E + 00$)	100%	$3.01E - 01$ ($8.26E - 01$)	12%	-0.364
f_{10}	$6.79E - 15$ ($1.33E - 15$)	100%	$8.17E - 11$ ($1.83E - 11$)	100%	-4.465
f_{11}	$0.00E + 00$ ($0.00E + 00$)	100%	$0.00E + 00$ ($0.00E + 00$)	100%	NaN
f_{12}	$6.21E - 30$ ($3.12E - 30$)	100%	$3.96E - 21$ ($1.61E - 21$)	100%	-2.459
f_{13}	$7.28E - 29$ ($3.56E - 29$)	100%	$2.92E - 20$ ($8.52E - 21$)	100%	-3.428
f_{14}	$3.38E - 16$ ($1.13E - 16$)	100%	$3.20E - 16$ ($1.29E - 16$)	100%	0.103
f_{15}	$3.27E - 04$ ($4.46E - 05$)	0%	$3.35E - 04$ ($4.35E - 05$)	0%	-0.130
f_{16}	$1.66E - 15$ ($4.85E - 15$)	100%	$8.88E - 16$ ($8.01E - 16$)	100%	0.157
f_{17}	$3.81E - 12$ ($1.32E - 11$)	100%	$2.83E - 14$ ($9.17E - 14$)	100%	0.287
f_{18}	$1.84E - 14$ ($8.71E - 15$)	100%	$2.21E - 14$ ($9.13E - 15$)	100%	-0.296
f_{19}	$6.93E - 16$ ($2.25E - 16$)	100%	$8.35E - 16$ ($2.66E - 16$)	100%	-0.407
f_{20}	$4.80E - 16$ ($1.23E - 16$)	100%	$7.82E - 16$ ($3.46E - 16$)	100%	-0.823
f_{21}	2.089174 (2.753900)	12%	3.990006 (3.399740)	12%	-0.434
f_{22}	$4.68E - 08$ ($9.97E - 08$)	76%	$9.24E - 03$ ($4.62E - 02$)	92%	-0.200
f_{23}	$7.92E - 05$ ($3.96E - 04$)	84%	$1.94E - 07$ ($7.49E - 07$)	64%	0.200

In order to compare the performance fairly, all control parameters are kept the same with their corresponding paper except the population size. The population size is 50 in this section. Results for OX-based algorithms with 25 independent runs are summarized in Table 6. The convergence curves and boxplot figures are listed in Figures 2 and 3 separately.

From Table 6, we can find that the results of OXBBO are significantly better than those of OXDE, OLPSO in 12, and 10 out of 23 functions, while this number for OXDE, OLPSO are 3, 3 out of 23 functions. In terms of the success rate, the average SR of OXDE, OLPSO, and OXBBO is 74%, 71%, and 82%, respectively. By carefully looking at the results in Table 6, one can note that OXDE, OLPSO, and OXBBO achieve a 100% success rate for f_1 , f_2 , f_6 , f_7 , f_8 , f_{10} , f_{11} , f_{12} , f_{13} , f_{16} , f_{17} , f_{18} , and f_{19} . Additionally, for function f_4 , OXBBO surpass OXDE and OLPSO. For f_9 , OXBBO and OLPSO achieve the optima, but the successful rate of OXBBO is higher than that of OLPSO. For f_5 , OXDE and OLPSO can reach the optima, but OXBBO failed.

From Figures 2 and 3, we can see that OXBBO is not only converging faster than both OXDE and OLPSO, but also converging more robust than both two. From the analysis, we can conclude that the overall performance of OXBBO is better than OXDE and OLPSO.

5.7. Analysis the Performance of QOX. From the analysis above, we know that the proposed algorithm OXBBO possesses very good performance through comparison with

improved BBO algorithms, hybrid BBO algorithms, and some state-of-the-art orthogonal-based algorithms. Does QOX really play a key role of OXBBO? For these reasons, in this section, a detailed analysis is made to analyze the performance of QOX operator in BBO. We consider two variants of OXBBO: OXBBO with QOX operator, OXBBO without QOX operator. These algorithms are denoted as OXBBO, OXBBO1. The experimental test performs on 23 test functions with 25 independent runs. The some parameter setting suggested in Section 5.1. The mean and standard deviation of the function error values and *t*-test values of 25 independent runs for each algorithm have been listed in Table 7. The convergence curves and boxplot figures are listed in Figures 4 and 5.

According to the *t*-test, the results of OXBBO are significantly better than those of OXBBO1 in 11 out of 23 test functions, and the results of OXBBO are similar to those of OXBBO1 in 11 out of 23 functions. OXBBO1 surpasses OXBBO on only one function. From this, we know that QOX has played a very important role in enhancing the performance of OXBBO, which not only improves the solution precision but also enhances the reliability. This can also be demonstrated in Figures 4 and 5.

6. Conclusions and Future Work

In this paper, a new improved BBO algorithm (OXBBO) is present through the combination of the QOX operator

(a)

FIGURE 2: Continued.

(b)

FIGURE 2: (a) Convergence curves of mean fitness error values of OXBBO, OXDE, and OLPSO for functions $f_1, f_2, f_3, f_4, f_5, f_7, f_8, f_9, f_{10}$, and f_{11} . (b) Convergence curves of mean fitness error values of OXBBO, OXDE, and OLPSO for functions $f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{21}$, and f_{22} .

(a)

FIGURE 3: Continued.

(b)

FIGURE 3: (a) ANOVA tests of fitness error values of functions $f_1, f_3, f_4, f_5, f_7, f_8, f_9, f_{10}, f_{12}$, and f_{13} for OXBBO, OXDE, and OLPSO. (b) ANOVA tests of fitness error values of functions $f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, f_{20}, f_{21}, f_{22}$, and f_{23} for OXBBO, OXDE, and OLPSO.

FIGURE 4: Convergence curves of mean fitness error values of OXBBO and OXBBO1 for functions $f_1, f_3, f_5, f_7, f_8, f_9, f_{11}, f_{15}, f_{18}$, and f_{22} .

FIGURE 5: ANOVA tests of fitness error values of functions $f_1, f_5, f_7, f_9, f_{12}, f_{14}, f_{15}, f_{16}, f_{20}$, and f_{21} for OXBBO and OXBBO1.

into BBO algorithm, which is designed to overcome the shortcoming that migration operator only visits one vertex of the hyperrectangle defined by the immigration and emigration vectors. In OXBBO, QOX is able to make a systematic and rational search in the region and enhance the exploration ability. Moreover, a modified migration operator is used in OXBBO to improve the population diversity. In this paper, extensive experiments have been implemented to compare the performance of OXBBO with state-of-the-art BBO algorithms and orthogonal crossover-based evolutionary algorithms. We have also experimentally studied the effect of QOX on the performance of OXBBO.

We have observed that the results obtained by OXBBO differ very much with different population sizes, which maybe due to the search frequency of QOX, similar results in [28]. In the future, we will study the effect of the search frequency of QOX on the performance in BBO framework for large scale dimension test functions. Furthermore, we would like to point out that experimental tests are based on the no-rotation functions; studying the performance on rotation functions is also our future work.

Acknowledgments

The authors would like to thank Doctor Zhan for providing the OLPSO code. The authors wish to make the acknowledgement of the area editor and the anonymous reviewers for their valuable comments and suggestions on this paper. This work is proudly supported in part by the National Natural Science Foundation of China (no. 60974082, 11101101, 11226219, and 51174263), Foundation of National Natural Science Foundation of Guangxi Region (no. 2013GXNSFBA019008, 2013GXNSFAA019003), and Project supported by Program to Sponsor Teams for Innovation in the Construction of Talent Highlands in Guangxi Institutions of Higher Learning.

References

- [1] E. L. Lawler and D. E. Wood, "Branch-and-bound methods: a survey," *Operations Research*, vol. 14, pp. 699–719, 1966.
- [2] J. Vlček and L. Lukšan, "A conjugate directions approach to improve the limited-memory BFGS method," *Applied Mathematics and Computation*, vol. 219, no. 3, pp. 800–809, 2012.
- [3] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," *Evolutionary Computation*, vol. 1, pp. 82–87, 1994.
- [4] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.
- [5] M. Dorigo, V. Maniezzo, and A. Colnari, "The ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.
- [6] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [7] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.
- [8] Q. Zhang and Y. W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 53–62, 1999.
- [9] Y. W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.
- [10] X. Li, J. Wang, J. Zhou, and M. Yin, "A perturb biogeography based optimization with mutation for global numerical optimization," *Applied Mathematics and Computation*, vol. 218, no. 2, pp. 598–609, 2011.
- [11] W. Gong, Z. Cai, C. X. Ling, and H. Li, "A real-coded biogeography-based optimization with mutation," *Applied Mathematics and Computation*, vol. 216, no. 9, pp. 2749–2758, 2010.
- [12] X. Li and M. Yin, "Multi-operator based biogeography based optimization with mutation for global numerical optimization," *Computers & Mathematics with Applications*, vol. 64, no. 9, pp. 2833–2844, 2012.
- [13] I. Boussaid, A. Chatterjee, P. Siarry, and M. Ahmed-Nacer, "Two-stage update biogeography-based optimization using differential evolution algorithm (DBBO)," *Computers and Operations Research*, vol. 38, no. 8, pp. 1188–1198, 2011.
- [14] M. R. Lohokare, S. S. Pattnaik, S. Devi, and S. Das, "Extrapolated biogeography-based optimization for global numerical optimization and micro strip patch antenna design," *International Journal of Applied Evolutionary Computation*, vol. 1, no. 3, pp. 1–26, 2010.
- [15] W. Gong, Z. Cai, and C. X. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.
- [16] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 517–525, 2011.
- [17] H. Ma, "An analysis of the equilibrium of migration models for biogeography-based optimization," *Information Sciences*, vol. 180, no. 18, pp. 3444–3464, 2010.
- [18] Z. H. Cai, W. Y. Gong, and C. X. Ling, "Research on a novel biogeography-based optimization algorithm based on evolutionary programming," *System Engineering Theory and Practice*, vol. 30, no. 6, pp. 1106–1112, 2010.
- [19] L. Wang and Y. Xu, "An effective hybrid biogeography-based optimization algorithm for parameter estimation of chaotic systems," *Expert Systems with Applications*, vol. 38, no. 12, pp. 15103–15109, 2011.
- [20] M. R. Lohokare, S. S. Pattnaik, S. Devi, B. K. Panigrahi, S. Das, and K. M. Bakwad, "Intelligent biogeography-based optimization for discrete variables," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NABIC '09)*, pp. 1088–1093, Coimbatore, India, December 2009.
- [21] W. Zhu, "Parallel biogeography-based optimization with GPU acceleration for nonlinear optimization," in *Proceedings of the ASME, International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pp. 1–9, Montreal, Quebec, Canada, August 2010.
- [22] L.-X. Tan and L. Guo, "Quantum and Biogeography based Optimization for a Class of Combinatorial Optimization," in

Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 969–972, New York, NY, USA, 2009.

- [23] D. Simon, “A probabilistic analysis of a simplified biogeography-based optimization algorithm,” *Evolutionary Computation*, vol. 19, no. 2, pp. 167–188, 2011.
- [24] D. Simon, R. Rarick, and M. Ergezer, “Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms,” *Information Sciences*, vol. 181, no. 7, pp. 1224–1248, 2011.
- [25] D. Simon, “A dynamic system model of biogeography-based optimization,” *Applied Soft Computing*, vol. 11, no. 8, pp. 5652–5661, 2011.
- [26] H. Ma, M. Fei, D. Simon, and M. Yu, “Biogeography-based optimization in noisy environments,” Tech. Rep., 2012, <http://academic.csuohio.edu/simond/bbo/noisy/>.
- [27] M. Ergezer, D. Simon, and D. Du, “Oppositional biogeography-based optimization,” in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1035–1040, San Antonio, Tex, USA, 2009.
- [28] Y. Wang, Z. Cai, and Q. Zhang, “Enhancing the search ability of differential evolution through orthogonal crossover,” *Information Sciences*, vol. 185, no. 1, pp. 153–177, 2012.
- [29] X. Yao, Y. Liu, and G. Lin, “Evolutionary programming made faster,” *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.
- [30] E. Mezura-Montes and C. A. Coello Coello, “A simple multi-membered evolution strategy to solve constrained optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 1, pp. 1–17, 2005.