Hindawi
Complexity
Volume 2019, Article ID 4182148, 20 pages
https://doi.org/10.1155/2019/4182148

Research Article

Improved Monarch Butterfly Optimization Algorithm Based on Opposition-Based Learning and Random Local Perturbation

Lin Sun[1], **Suisui Chen**[1], **Jiucheng Xu**[1] **and Yun Tian**[2]

[1]*College of Computer and Information Engineering, Henan Normal University, Xinxiang 453007, China*
[2]*College of Information Science and Technology, Beijing Normal University, Beijing 100875, China*

Correspondence should be addressed to Jiucheng Xu; jiuchxu@gmail.com

Received 4 July 2018; Revised 25 October 2018; Accepted 19 November 2018; Published 10 February 2019

Guest Editor: Yimin Zhou

Many optimization problems have become increasingly complex, which promotes researches on the improvement of different optimization algorithms. The monarch butterfly optimization (MBO) algorithm has proven to be an effective tool to solve various kinds of optimization problems. However, in the basic MBO algorithm, the search strategy easily falls into local optima, causing premature convergence and poor performance on many complex optimization problems. To solve the issues, this paper develops a novel MBO algorithm based on opposition-based learning (OBL) and random local perturbation (RLP). Firstly, the OBL method is introduced to generate the opposition-based population coming from the original population. By comparing the opposition-based population with the original population, the better individuals are selected and pass to the next generation, and then this process can efficiently prevent the MBO from falling into a local optimum. Secondly, a new RLP is defined and introduced to improve the migration operator. This operation shares the information of excellent individuals and is helpful for guiding some poor individuals toward the optimal solution. A greedy strategy is employed to replace the elitist strategy to eliminate setting the elitist parameter in the basic MBO, and it can reduce a sorting operation and enhance the computational efficiency. Finally, an OBL and RLP-based improved MBO (OPMBO) algorithm with its complexity analysis is developed, following on which many experiments on a series of different dimensional benchmark functions are performed and the OPMBO is applied to clustering optimization on several public data sets. Experimental results demonstrate that the proposed algorithm can achieve the great optimization performance compared with a few state-of-the-art algorithms in most of the test cases.

1. Introduction

Many real-world tasks, which can be transferred to optimization problems, have become increasingly complex and are difficult to solve using the traditional optimization algorithms [1]. Recently, a lot of nature-inspired metaheuristic algorithms have been proposed and applied to deal with various optimization problems [2]. Then, the researches on tackling by optimization techniques in many applications have become a fruitful field of research, especially those interested in solving global optimization problems. The swarm intelligence optimization (SIO) algorithm is a kind of bionic random method inspired by natural phenomena and biological behaviors and can deal with certain high-dimensional complex and variable optimization problems because of its better computing performance and simple model [3, 4].

Over the past several decades, SIO has become an attractive research area which leads to the emergence of a large variety of intelligent optimization algorithms. Kennedy and Eberhart [5] proposed a particle swarm optimization (PSO) algorithm derived from the simulation of bird foraging behaviors. However, the PSO often faces premature convergence problem, especially in multimodal problems as it may get stuck in specific point [6]. Wu and Yang [7] presented an elitist transposon quantum-based PSO to solve economic dispatch problems. Eusuff and Lansey [8] presented a shuffled frog-leaping algorithm (SFLA), which is inspired from the memetic evolution of frogs seeking food in a pond. It has been shown to be competitive with PSO, but the SFLA is good at exploration but poor at exploitation and easily gets trap in local optima when solving partial complex multimodal problems. Meanwhile, its convergence speed is

slower [9]. Tan and Zhu [10] designed a fireworks algorithm (FWA) for the global optimization of complex functions. The FWA has powerful global optimization capabilities to solve classification problems, but there is no direct interaction among the solutions found during the optimization process of FWA; its convergence speed is slow and the computational cost is high [11]. Yin et al. [12] introduced a hybrid FWA-based parameter optimization into the nonlinear hypersonic vehicle dynamics control to satisfy the design requirements with high probability. Gandomi and Alavi [13] developed a krill herd algorithm (KHA) which mimics the herding behavior of ocean krill individuals. The herding of the krill individuals is a multiobjective process, and the position of an individual krill is time dependent. Unfortunately, the performance of KHA is degraded by the poor exploitation capability, and the basic KHA has a low convergence speed and accuracy [14]. Singh and Khan [15] proposed an artificial shark optimization (ASO) method to remove the limitation of existing algorithms for solving the economical operation problem of microgrid. Mirjalili et al. [16] established a grey wolf optimizer (GWO) metaheuristic based on grey wolves. The GWO algorithm is considered for learning method due to its advantages, including high accuracy, effectiveness, and competitiveness [17]. The paramount challenge in GWO is that it is prone to stagnation in local optima [18]. Differential evolution (DE) as a popular stochastic optimizer, proposed by Storn and Price [19], is to exhibit consistent and reliable performance in nonlinear and multimodal environment and has proven to be effective for constrained optimization problems [20]. Some empirical studies have shown that DE outperforms PSO [21]. However, setting different parameters has great impacts on the performances of DE algorithm when solving various global optimization problems or even the same problem at different evolutionary stages [22]. The fruit fly optimization algorithm (FOA) as a global optimization method was proposed by Pan [23], who was inspired by the foraging behavior of fruit flies. The FOA is simple in structure and easy to implement [24]. However, the basic FOA often derives a local extreme when solving high-dimensional functions and large-scale combinational optimization problems [25]. The idea of ant colony optimization (ACO) is to mimic the way that real ants find the shortest route between a food source and their nest. Recently, the ACO algorithm and its versions have been investigated to tackle combinatorial optimization problems [26]. But the efficiency of ACO is unsatisfactory since each ant needs to search for a complete solution, and the runtime is rather long [27]. Abedinia et al. [28] introduced a shark smell optimization (SSO) algorithm, which is applied for the solution of load frequency control problem in electrical power systems. The monarch butterfly optimization (MBO) algorithm was first presented by Wang et al. [29], and it simulates the migration behaviors of monarch butterflies in nature. Although most of these heuristic techniques have the ability to provide fast and efficient solution, sometimes they suffer from discovering global optimal solution, slow convergence rate, and several parameters tuning [30]. Until now, the MBO algorithm has become one of the most widely used SIO algorithms, and it has two important operators, the migration operator and the butterfly adjusting operator [31]. The former provides a certain local search capability and the latter gives a global search capability. The search direction of monarch butterflies is mainly determined by the migration operator and the butterfly adjusting operator in MBO. Since the migration operator and the butterfly adjusting operator can be implemented simultaneously, MBO is ideally suited for parallel processing and is capable of making trade-offs between intensification and diversification [32]. In addition, the MBO algorithm has simple calculation process, requires less computational parameters, and is easy to implement by a program. Furthermore, some advantages of MBO are incomparable to many other intelligent optimization algorithms. Therefore, the MBO algorithm and its versions have been widely used in many fields, such as dynamic vehicle routing problem [30], 0-1 knapsack problem [31], neural network straining [32], optimal power flow problem [33], and prevention of osteoporosis [34].

In the last few years, in order to improve the performance of MBO, the scholars have made many improvements. In the basic MBO algorithm, after implementing the migration operator, the generated monarch butterfly will be accepted as a new monarch butterfly in the next generation regardless of whether it is better or worse. Then, Hu et al. [35] used self-adaptive and greedy strategies to improve the performance of the basic MBO. However, it suffers greatly from worse standard deviations and average fitness on some benchmarks. Wang et al. [36] developed a different version of MBO with greedy strategy and self-adaptive crossover operator (GCMBO), in which the greedy strategy can accelerate convergent speed and the self-adaptive crossover operator can significantly improve the diversity of population at later run phase of the search. Feng et al. [37] proposed a chaotic MBO algorithm, in which the chaos theory was employed to enhance its global optimization ability. Feng et al. [38] introduced neighborhood mutation with crowding and Gaussian perturbation into MBO algorithm, in which the first strategy enhances the global search ability, while the second strategy strengthens local search ability and prevents premature convergence during the evolution process. At present, the MBO is usually combined with other SIO methods to improve the optimization performance. The main objective is to improve the balance between the characteristics of exploration and exploitation in those algorithms in order to address the issues of trapping in local optimal solution, slow convergence, and low accuracy in complex optimization problems [39]. Ghanem and Jantan [40] presented a metaheuristic algorithm that combined artificial bee colony optimization with the MBO. Ghetas et al. [41] introduced the harmony search algorithm into MBO to enhance the search ability of MBO, in which mutation operators were added to the process of adjusting operator to enhance the exploitation and exploration ability and speed up the convergence rate of MBO. Strumberger et al. [42] incorporated the search mechanism of firefly algorithm (FA) into MBO to overcome this deficiency that in early iterations exceedingly directs the search process toward the current best solution in MBO. However, most of the abovementioned MBO algorithms still easily fall into local optima and are rather slow in convergence. This inspires

the authors to investigate new nature-inspired optimization algorithm about MBO.

Based on the above analyses of MBO, it is clear that falling into local optima easily is one of the typical disadvantages of MBO, and there are many ways to improve this drawback. The OBL method, proposed by Tizhoosh [43], is one of the most effective methods. It can prevent the algorithm from falling into local optima to some degree. For example, Shang et al. [44] introduced OBL, dynamic inertia weight, and a postprocedure to improve PSO with mutual information as its fitness function to detect SNP-SNP interactions, in which OBL enhances the global explorative ability. Since the poor exploration capabilities of SFLA sometimes get trapped in local optima, which results in poor convergence, Sharma and Pant [45] embedded the OBL into the memeplexes before the frog initiates foraging, which enhances the local search mechanism of SFLA but also improves the diversity. Ahandani and Alavi-Rad [46] used new versions of the SFLA which on the one hand employed the OBL to accelerate the SFLA without making premature convergence and on the other hand used the OBL strategy to diversify search moves of SFLA. Yu et al. [47], inspired by the OBL, improved the performance of the FA, in which the worst firefly is forced to escape from the normal path after OBL operation and can help it to escape from local optima. Yang et al. [48] presented an improved artificial bee colony algorithm based on OBL to overcome the shortcomings of the slow convergence rate and sinking into local optima. Shan et al. [49] embedded the OBL into the bat algorithm to enhance the diversity and convergence capability. Park and Lee [50] combined differential evolution with OBL to obtain a high-quality solution with low-computational effort. Kumar and Sahoo [51] presented a cat swarm optimization (CSO) algorithm via the OBL, which can enhance the diversity of the algorithm. Sarkhel et al. [52] applied OBL to the harmony search algorithm to overcome slow convergence to the globally optimal solutions. Zhang et al. [53] merged the OBL into the biogeography-based optimization algorithm to prevent the algorithm from falling into the local optima. Zhang et al. [54] added the OBL to the GWO (OGWO) to prevent the algorithm from falling into the local optima. In recent years, the OBL has become a widely used technique in optimization algorithms. It is noted that the OBL can increase population diversity and enhance global search ability [55]. If the current best candidate solution falls into the local optima, it may mislead the other candidate solutions into the local optima. However, its opposite is often far from the local optima. Therefore, this paper introduces the OBL into the initial phase of MBO, effectively avoiding the algorithm falling into the local optimum. Furthermore, the opposition-based individuals are generated by the OBL such that the best individual can be accepted. This operation can efficiently prevent the algorithm from falling into the local optima to some extent.

What is more, there exists much insufficiency for MBO about its solution search mechanism which may bring the premature convergence and the low search accuracy when solving complex optimization problems [49]. Then, considering that MBO converges very slowly, a perturbation operator strategy can be used to ensure the diversity of monarch butterfly against the premature convergence. For example, Liu et al. [9] designed a perturbation operator strategy in a convergence state to help the best frog to jump out of possible local optima to further increase the performance of SFLA. Wang et al. [56] proposed an improved FOA using swarm collaboration and random perturbation strategy to enhance the performance. Li et al. [57] developed an artificial bee colony algorithm with random perturbations for numerical optimization, in which the self-adaptive population perturbation strategy for the current colony is used by random perturbation to enhance the population diversity. Yu et al. [58] presented a teaching-learning-based optimization with a chaotic perturbation mechanism, which produces many solutions around the current best solution and thereby enhances the searching ability and global convergence. Li et al. [59] utilized the uniformity of Anderson chaotic mapping and performed chaos perturbation to part of particles based on the information of variance of the population's fitness to avoid the untimely aggregation of particle swarm. Based on the ideas of random perturbation, a novel RLP operator is proposed to prevent premature convergence in this paper, and merged into the migration operator. The improved migration operator with RLP shares the information of excellent individuals, which is conducive to guiding individuals to approach an optimal solution and accelerate convergence, and the works are not considered in previous MBO.

The remainder of this paper is organized as follows: Section 2 reviews some related theory of the MBO algorithm. In Section 3, the OBL method and RLP-based migration operator are investigated, and the main procedure of improved MBO and its complexity analysis are given. Section 4 describes the experimental results and analysis. Finally, the conclusion is summarized in Section 5.

2. Related Work

The theory of the MBO algorithm can be found in [29, 36]. In MBO, all monarch butterfly individuals are idealized and located in only two lands as follows: the northern United States and southern Canada (Land1), and Mexico (Land2). Then, the location of monarch butterflies is updated in two ways, namely, the migration operator and the butterfly adjusting operator. Firstly, the offspring are generated (location update) through the migration operator. Secondly, the location of other monarch butterflies is updated by the butterfly adjusting operator. Thus, the search direction of the monarch butterfly individual is determined by the migration operator and the butterfly adjusting operator. Moreover, the two operators can be performed simultaneously. Therefore, the MBO algorithm is suitable for parallel processing, and it has a good balance of strengthening and diversification. The MBO algorithm abides by the following ideal rules:

(1) All the monarch butterflies are located only in Land1 and Land2. Namely, the population of the entire monarch butterflies is composed by the monarch butterflies in Land1 and Land2.

(2) The offspring of each monarch butterfly are generated only by the migration operator in Land1 or Land2.

(3) To keep the population constant, once a descendant monarch butterfly is produced, a corresponding parent monarch butterfly will disappear.

(4) Monarch butterfly individuals with the best fitness automatically enter the next generation without any operation, and then it ensures that the quality of the monarch butterfly population does not decline as the number of iterations increases.

The MBO algorithm contains two important operators, which are described as follows.

The first operator is the migration operator, whose purpose is to update the migration of the monarch butterflies between Land1 and Land2. The total number of monarch butterflies is NP, and the numbers of monarch butterflies in Land1 and Land2 are $NP1$= ceil $(p \times NP)$ and $NP2 = NP - NP1$, respectively, where p is the migration rate of monarch butterflies with $p = 5/12$ in MBO, ceil(x) rounds x to the nearest integer greater than or equal to x, the subpopulation of Land1 is denoted as Subpopulation1, and the subpopulation of Land2 is denoted as Subpopulation2. Then, the migration operator is expressed as

$$x_{i,k}^{t+1} = \begin{cases} x_{r_1,k}^{t}, & r \le p \\ x_{r_2,k}^{t}, & r > p, \end{cases} \tag{1}$$

where $x_{i,k}^{t+1}$ is the kth element of x_i in generation $t + 1$; similarly, $x_{r_1,k}^{t}$ denotes the kth element of x_{r_1} in generation t, and $x_{r_2,k}^{t}$ is the kth element of x_{r_2} in generation t; the current generation number is t, and the monarch butterflies r_1 and r_2 are randomly selected from Subpopulation1 and Subpopulation2, respectively. Here, r is calculated by $r = rand \times peri$, where $peri$ is the migration period, which is equal to 1. 2 in MBO and $rand$ is a random number in $[0, 1]$.

The second operator is the butterfly adjusting operator, which is used to update the position of monarch butterfly in Subpopulation2. The formula is described as

$$x_{j,k}^{t+1} = \begin{cases} x_{best,k}^{t}, & rand \le p \\ x_{r_3,k}^{t}, & rand > p \ \& \ rand \le BAR \\ x_{i,k}^{t+1} + \alpha \times \left(dx_k - 0.5\right), & rand > p \ \& \ rand > BAR, \end{cases} \tag{2}$$

where $x_{j,k}^{t+1}$ is the kth element of x_j in generation t+1; similarly $x_{best,k}^{t}$ is the kth element of x_{best} in generation t, which is the best location for monarch butterflies in Land1 and Land2, $x_{r_3,k}^{t}$ is the kth element of x_{r_3} in generation t, the monarch butterfly r_3 is randomly selected from Subpopulation2, and BAR is the adjustment rate. If BAR is less than the random number $rand$, the kth element of x_j at $t + 1$ is updated, where α is the weighting factor, and $\alpha = S_{\max}/t^2$, where S_{max} is the maximum walk step. In (2), dx is the walk step of butterflies j that can be calculated by the Levy flight such that $dx = Levy(x_j^t)$.

3. Improved MBO Algorithm Based on OBL and RLP

3.1. Motive of Improving MBO Algorithm. In the MBO algorithm, the migration operator and the butterfly adjusting operator can ensure the search direction of monarch butterflies. Furthermore, the migration operator and the butterfly adjusting operator can be executed simultaneously. The advantages of MBO algorithm include its simplicity and easy implementation. However, the drawbacks of MBO algorithm cause poor optimization efficiency in solving complex optimization problems, which are mainly described from two aspects as follows: First, from (1), the monarch butterflies r_1 and r_2 are randomly selected from Subpopulation1 and Subpopulation2, respectively. A worse monarch butterfly may be selected to share its features with a better one, leading to the population degenerating. Second, from the main process of MBO, when the elitist strategy is adopted, the population must be sorted twice during each generation, thus causing high time complexity. Thus, in order to overcome the above drawbacks and improve the optimization efficiency of MBO, several creative improvements are developed in this paper.

3.2. Opposition-Based Learning Method. In the fields of computational intelligence, the OBL is usually used to improve the convergence rate of many optimization algorithms. Its main idea is to take into account the current population as well as its opposite population at the same time and further obtain better candidate solution [55]. In recent years, the scholars have applied the OBL method in population-based optimization technique to enhance the convergence rate. It can be concluded that an opposite candidate solution has a better chance to be closer to the global optimum solution than a random candidate solution [51]. The opposite solution of OBL is denoted by the mirror point of the solution from the center of the search space, and then its formula can be mathematically expressed as

$$x_{i,j}{}' = a_i + b_i - x_{i,j}, \tag{3}$$

where $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,\mathrm{D}})$ is a feasible solution in a D-dimensional search space, $x_{i,j} = [a_i, b_j]$, $j = 1, 2, \ldots, D$, and its opposition-based solution is $x_i' = (x_{i,1}', x_{i,2}', \ldots, x_{i,\mathrm{D}}')$.

Note that if the OBL approach is introduced into the initialization of the MBO algorithm, it can produce the opposition-based population. Then, the better individuals are selected to participate in the evolution from the union of the original populations and the opposition-based populations. Thus, this operation increases the population diversity and expands the exploration scope of MBO.

3.3. Random Local Perturbation-Based Migration Operator. To overcome the shortcoming of the premature convergence of MBO, a novel RLP is constructed and merged into the migration operator of the MBO. For this, the RLP strategy can be defined as

$$x_{i,k}^{t+1} = x_{p,d}^{t} + rand \times \left(x_{q,d}^{t} - x_{p,d}^{t}\right), \tag{4}$$

```
for i = 1 to NP1 do
  for k = 1 to D do
    R = 0. 5.
    Set r as a random number in [0, 1].
    if r < R then
      Calculate x_{i,k}^{t+1} by Eq. (4).
    else
      Calculate x_{i,k}^{t+1} by Eq. (1).
    end if
  end for
  Calculate x_{i,new}^{t+1} with greedy strategy by Eq. (5).
end for
```

Algorithm 1

where x_p^t is an optimal solution in generation t, x_q^t is a suboptimal solution in generation t, $x_{i,k}^{t+1}$ is the kth element of the ith individual in generation $t+1$, $x_{p,d}^t$ is the dth element of the optimal solution in generation t, $x_{q,d}^t$ is the dth element of the suboptimal solution in generation t, and d can be calculated by $d = \lceil D \times rand \rceil$.

Equation (4) shares the information of the optimal solution and the suboptimal solution, which is conducive to guiding the current individual to move toward the optimal solution and the suboptimal solution. Then, the convergence speed can be effectively accelerated. Meanwhile, to maintain the diversity of the MBO search, a control parameter R is set as $R = 0.5$, where $R = 0.5$ is determined through many experiments, and a random number r from 0 to 1 is generated. When $r<R$, the location updating is performed according to (4); otherwise the location updating is performed according to (1).

For the basic MBO, the parameters of the elitist strategy need to be set. In each generation, the population will be sorted twice, which brings about much computation complexity. If the greedy selection method is adopted in MBO, the population at each generation is just sorted once. Thus, the elitist strategy can be replaced by the greedy selection method in the SIO algorithms [53]. Hence, during each generation, the new generated monarch butterflies are compared with the corresponding old ones, and the better one is selected. So, this replacement eliminates the elitist parameters, gets rid of a sorting, and further improves the operation efficiency. It follows that the greedy strategy is introduced into the improved migration operator with RLP, and the superior candidate solution is retained by the principle of survival of the fittest. Here, the greedy strategy can be expressed as

$$x_{i,new}^{t+1} = \begin{cases} x_i^{t+1}, & f\left(x_i^{t+1}\right) < f\left(x_i^t\right) \\ x_i^t, & \text{otherwise,} \end{cases} \tag{5}$$

where $x_{i,new}^{t+1}$ is the generation $t+1$ of new monarch butterfly individuals, and $f(x_i^{t+1})$ and $f(x_i^t)$ represent the fitness values of two monarch butterflies x_i^{t+1} and x_i^t, respectively.

The special steps of the improved migration operator with RLP are described in Algorithm 1.

For Algorithm 1, the improved migration operator with RLP shows that our proposed method with sharing information can make full use of the information of the high-quality individuals in the current population, and improve the local optimization ability. Moreover, the greedy strategy only retains individuals who have a better fitness, which efficiently enhances the convergence rate.

3.4. Main Procedure of Improved MBO Algorithm. All the above improvements can enhance the optimization performance of the MBO algorithm. The main process of OBL and RLP-based improved MBO (OPMBO) algorithm can be illustrated in Figure 1. The special steps of the OPMBO algorithm are provided in Algorithm 2.

3.5. Complexity Analysis. Under the same software and hardware on all systems, the computational complexity of the optimization algorithm is mainly composed of two parts as follows: one is the complexity of the objective function, and the other is the complexity of the algorithm process. In the comparison experiment, six kinds of SIO algorithms, namely, the MBO algorithm [29], the GCMBO algorithm [36], the OPMBO algorithm, the FOA based on hybrid location information exchange mechanism (HFOA) [60], the GWO algorithm [16], and the OGWO algorithm [54], have the same population number and maximum number of iterations, so that their maximum function evaluation times are equal. Thus, the complexity of OPMBO algorithm mainly focuses on its operation process. For the OPMBO algorithm, the time complexity is polynomial. Assume that the maximum number of iterations of the OPMBO is *MaxGen*, the population size is *NP*, the Subpopulation1 is *NP*1, the Subpopulation2 is *NP*2 with $NP2 = NP - NP1$, and the dimension is D. In an effort to avoid confusion and awkward phrasing, *MaxGen* is replaced by T, *NP* is replaced by N, and *NP*1 is replaced by $N1$. According to Figure 1, the time complexity of the algorithm is mainly determined by each iteration cycle. The detailed analysis of time complexity for OPMBO is as follows: The first step is to calculate the fitness value of the monarch butterflies, and then the time complexity is $O(N)$. The second step is sorting, and the time complexity of Quicksort algorithm in [31] is $O(N \log N)$. The third step is to divide the population into two subpopulations, and the time complexity is $O(N)$. The fourth step is to firstly run the improved migration operator, which has two inner loops, and then the time complexity is $O(N1 \times D)$. And secondly for the butterfly adjusting operator, there are two inner loops, whose time complexity is $O((N - N1) \times D)$. Therefore, the total time complexity of OPMBO algorithm is $T(n) = O(f(n)) = O(T \times (N + N \log N + N + (N1 \times D) + ((N - N1) \times D))) = O(T \times (2N + N \log N + N \times D)) = O(T \times N \log N)$. In the OPMBO algorithm, since the variable storage space is affected by the population size N and the variable dimension D, the space complexity can be calculate by $S(n) = O(f(n)) = O(N \times D)$.

4. Experimental Results and Analysis

4.1. Experiment Preparation. To verify the optimization performance of OPMBO, a series of experiments are performed

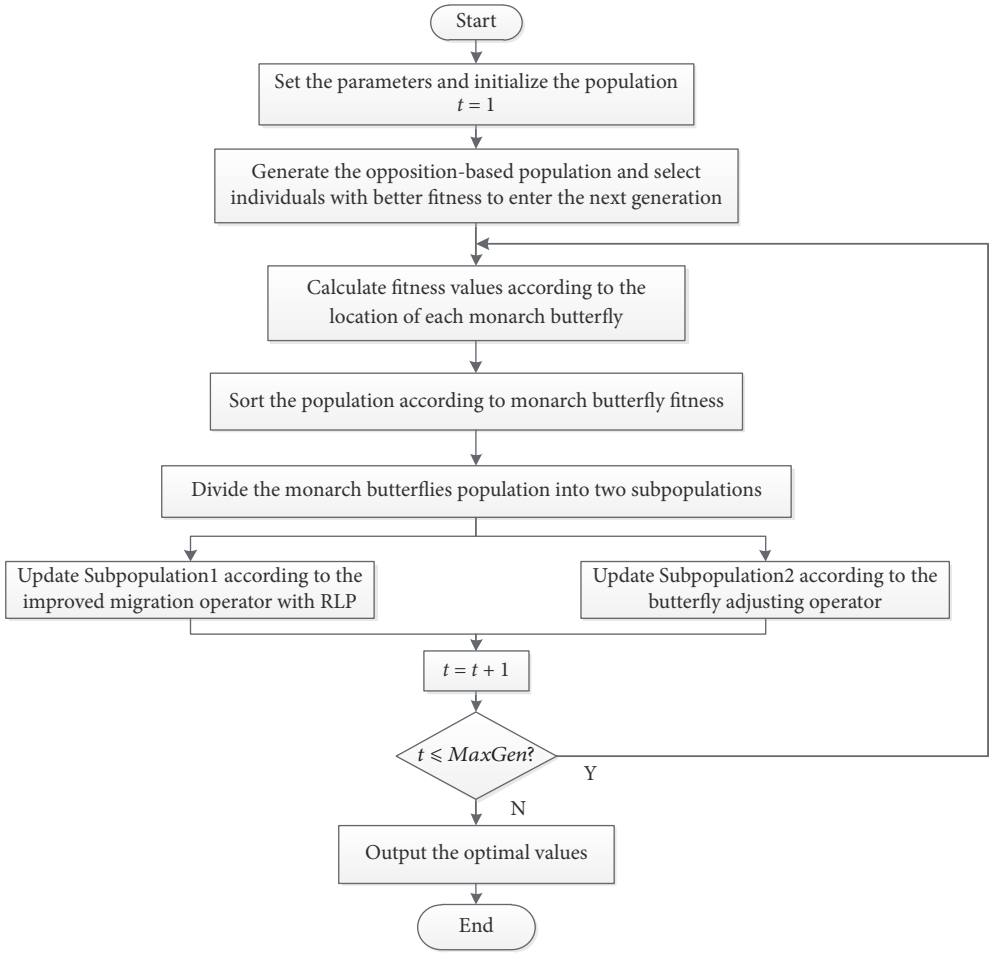

Figure 1: The main process of the OPMBO algorithm.

Step 1. Set the population quantity NP, the maximum generation $MaxGen$, the dimensions D, the max walk step size S_{max}, the adjusting rate BAR, the migration period $peri$ and the migration rate p. Let the current cycle counter $t = 1$. //Initialization operation
Step 2. Generate the opposition-based population according to OBL. Select the individuals with better fitness to enter the next generation from the original and opposition-based populations.
Step 3. Calculate their fitness values according to the location of each monarch butterfly. //Fitness evaluation
Step 4. **While** $t \leq MaxGen$ **do**
Sort the population according to monarch butterfly fitness using Quicksort algorithm in [31].
Divide the monarch butterfly population into two subpopulations, i.e., Subpopulation1 and Subpopulation2.
for $i = 1$ to $NP1$ **do**
Update Subpopulation1 using Algorithm 1.
end for
for $j = 1$ to $NP2$ **do**
Update Subpopulation2 by Eq. (2).
end for
Merge two new subpopulations into a new population.
Recalculate the fitness values of each monarch butterfly according to the updated position.
Let $t = t + 1$.
Step 5. **end while**
Step 6. Output the optimal values.

Algorithm 2

on various benchmark functions. The test of benchmark functions is a common and popular method to verify the performance of intelligent algorithms. For example, Wang et al. [29] introduced 38 benchmark functions to demonstrate the superior performance of the MBO algorithm, and the results clearly exhibit the capability of the MBO toward finding the enhanced function values on most of the benchmark problems. Wang et al. [36] employed 18 benchmark functions to test the GCMBO algorithm, and the results indicate that GCMBO significantly outperforms the basic MBO method on almost all the test cases. Zhang et al. [53] marked 21 benchmark functions to verify the efficiency of biogeography-based optimization algorithm. Zhang et al. [54] used 30 benchmark functions to illustrate the performance of hybrid algorithm based on biogeography-based optimization and GWO. In our experiments, the typical 12 benchmark functions are selected from [29, 36, 53, 54] to test the performance of our OPMBO algorithm, which can be rigorous to verify the effectiveness of all of the compared algorithms. The information for 12 benchmark functions is shown in Table 1, where the F_{min} is the minimum value (ideal optimal value) of the function. These benchmark functions can be classified into two different types, unimodal functions f_1-f_7 and multimodal functions f_8-f_{12}.

The experiments were performed on a personal computer running Windows 7 with an Intel(R) Core(TM)CPU operating at 3.10 GHz and 4 GB memory. All the simulation experiments were implemented in MATLAB R2014a.

4.2. Comparison of OPMBO with MBO and GCMBO on Different Dimensions. The objective of the following experiments is to show the comparison results of 12 benchmark functions on different dimensions. The two state-of-the-art algorithms, the MBO [29] and the GCMBO [36], are selected as the comparison algorithms to evaluate the effectiveness of the OPMBO. Following the experimental techniques developed by Wang et al. [36] and Zhang et al. [53], the three dimensions of the functions are set as follows: the low-dimensional (20 dimensions), the medium-dimensional (50 dimensions), and the high-dimensional (100 dimensions). As the dimension increases, the difficulty of the problem increases, which verifies that OPMBO has the ability to handle the complex optimization problems. Then, the optimization experiments of the three algorithms (MBO, GCMBO, and OPMBO) are performed on the three different dimensions for the 12 benchmark functions to verify the optimization performance of the OPMBO. The related parameter values for testing the three algorithms are shown in Table 2. Following the experimental techniques designed by Wang et al. [29], the D describes the dimension, the *MaxGen* is the maximum number of iterations, the *NP* describes the number of monarch butterfly population, and the *Num* represents the independent running times of each optimization problem. It is known that along with the increase of the dimension of the benchmark function, the maximum number of iterations *MaxGen* will increase. The monarch butterfly population of the three algorithms is uniformly set to 50, and the other parameters are the same as in [29, 36]. In order to reduce the random error, each method is run 30 times independently for each optimization problem, and the results are the average value of 30 time evaluations. The experimental results for different dimensions (20, 50, and 100 dimensions) on 12 benchmark functions are listed in Tables 3–5, respectively, where the best values are in bold font. Following the experimental techniques designed in [29, 36, 53, 54, 60, 61], the optimal value as *Best*, the worst value as *Worst*, the mean value as *Mean*, and the standard deviation as *Std* of the fitness values of benchmark functions for the 30 independent experimental results are employed to test the MBO, GCMBO, and OPMBO algorithms. Here, Zhang et al. [53] and Wang et al. [36] declared that the lower *Mean* and *Std* values indicate a better algorithm with respect to search ability and stability in their experimental analysis.

The first part of this experiment is conducted on the 20 dimensions, and the results are illustrated in Table 3. It can be seen from Table 3 that on f_1-f_4, f_8-f_{10}, and f_{12}, the OPMBO algorithm achieves the best optimization results on the *Best*, *Worst*, *Mean*, and *Std* values. In particular, the theoretical optimal value is obtained by OPMBO on f_{12}. On f_5 and f_7, although the *Best* value of OPMBO is not the best, it has achieved the best optimization results on the *Worst*, *Mean*, and *Std* values. The *Best* value of GCMBO is the same as that of OPMBO, but OPMBO achieves the best results for the other values on f_{11}. On f_6, the three algorithms achieve the best value, but the *Mean* and *Std* values of OPMBO are better. Therefore, the experimental results on the 20-dimensional benchmark function show that the performance of OPMBO is excellent for the low-dimensional functions.

In what follows, this portion of our experiment is to be performed on the 50 dimensions, and the results are shown in Table 4. The values of OPMBO are not as well as that of GCMBO on f_1. However, the OPMBO algorithm achieves the best results on f_2-f_4, f_8, and f_9. Both the *Mean* and *Std* values of OPMBO are optimal on f_5 and f_{10}-f_{12}. As the dimensions increase, the optimization performances of MBO and GCMBO decrease severely, whereas OPMBO does not decrease on f_6. To more intuitively demonstrate the convergence speed and the local and global search ability of the three algorithms, the convergence curves of the three algorithms of the experimental results on the 50-dimensional functions in Table 4 are shown in Figure 2.

Figure 2 shows that for f_1, the convergence rate of OPMBO is better than those of MBO and GCMBO in the initial iterations, whereas the convergence of OPMBO is slower than that of GCMBO in later iterations. Overall, both GCMBO and OPMBO have a much better effect than MBO on f_1. The convergence speed of OPMBO is clearly faster than those of MBO and GCMBO on f_2-f_{11}. For f_2, when iterating to 200 times, the MBO and GCMBO algorithms have stopped, but the OPMBO has been updating the search. So, the convergence of OPMBO is faster. For f_3, although the convergence rate of GCMBO is the best at the beginning, the convergence of OPMBO gradually shows an optimal trend as the iteration progresses. For f_4, the convergence curves of MBO and GCMBO are basically the same, but the curve of OPMBO shows obviously a faster convergence speed. For f_5, from the iteration of 100 times, both MBO and GCMBO are caught in the search stagnation, but the OPMBO gradually

Table 1: Overview of 12 benchmark functions.

Function	Formula	Search range	F_{min}
Sphere	$f_1(x) = \sum_{i=1}^{d} x_i^2$	$[-5.12, 5.12]^D$	0
Quartic	$f_2(x) = \sum_{i=1}^{d} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
Schwefel 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^D$	0
Schwefel 2.21	$f_4(x) = \max_i (\lvert x_i \rvert,\ 1 \le i \le d)$	$[-100, 100]^D$	0
Schwefel 2.22	$f_5(x) = \sum_{i=1}^{d} \lvert x_i \rvert + \prod_{i=1}^{d} \lvert x_i \rvert$	$[-10, 10]^D$	0
Step	$f_6(x) = \sum_{i=1}^{d} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0
Rosenbrock	$f_7(x) = \sum_{i=1}^{d-1} \left((x_i - 1)^2 + 100 \left(x_i^2 - x_{i+1} \right)^2 \right)$	$[-10, 10]^D$	0
Penalized 1	$f_8(x) = \frac{\pi}{d} \left(10 \sin^2 (\pi y_i) + \sum_{i=1}^{d-1} (y_i - 1)^2 \left(1 + 10 \sin^2 (\pi y_{i+1}) \right) + (y_d - 1)^2 \right) + \sum_{i=1}^{d} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$, and $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50, 50]^D$	0
Penalized 2	$f_9(x) = 0.1 \left(\sin^2 (\pi x_1) + \sum_{i=1}^{d-1} (x_i - 1)^2 \left(1 + \sin^2 (3\pi x_{i+1}) \right) (x_d - 1) \left(1 + \sin^2 (2\pi x_d) \right) \right) + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$	$[-50, 50]^D$	0
Ackley	$f_{10}(x) = -20 \exp \left(-\frac{1}{5} \sqrt{\frac{1}{d} \sum_{i=1}^{d} x_i^2} \right) - \exp \left(\frac{1}{d} \sum_{i=1}^{d} \cos (2\pi x_i) \right) + 20 + e$	$[-30, 30]^D$	0
Schwefel 2.26	$f_{11}(x) = 418.98288727243369 \times d - \sum_{i=1}^{d} x_i \sin \left(\sqrt{x_i} \right)$	$[-500, 500]^D$	0
Rastrigin	$f_{12}(x) = \sum_{i=1}^{d} \left(x_i^2 - 10 \cos (2\pi x_i) + 10 \right)$	$[-5.12, 5.12]^D$	0

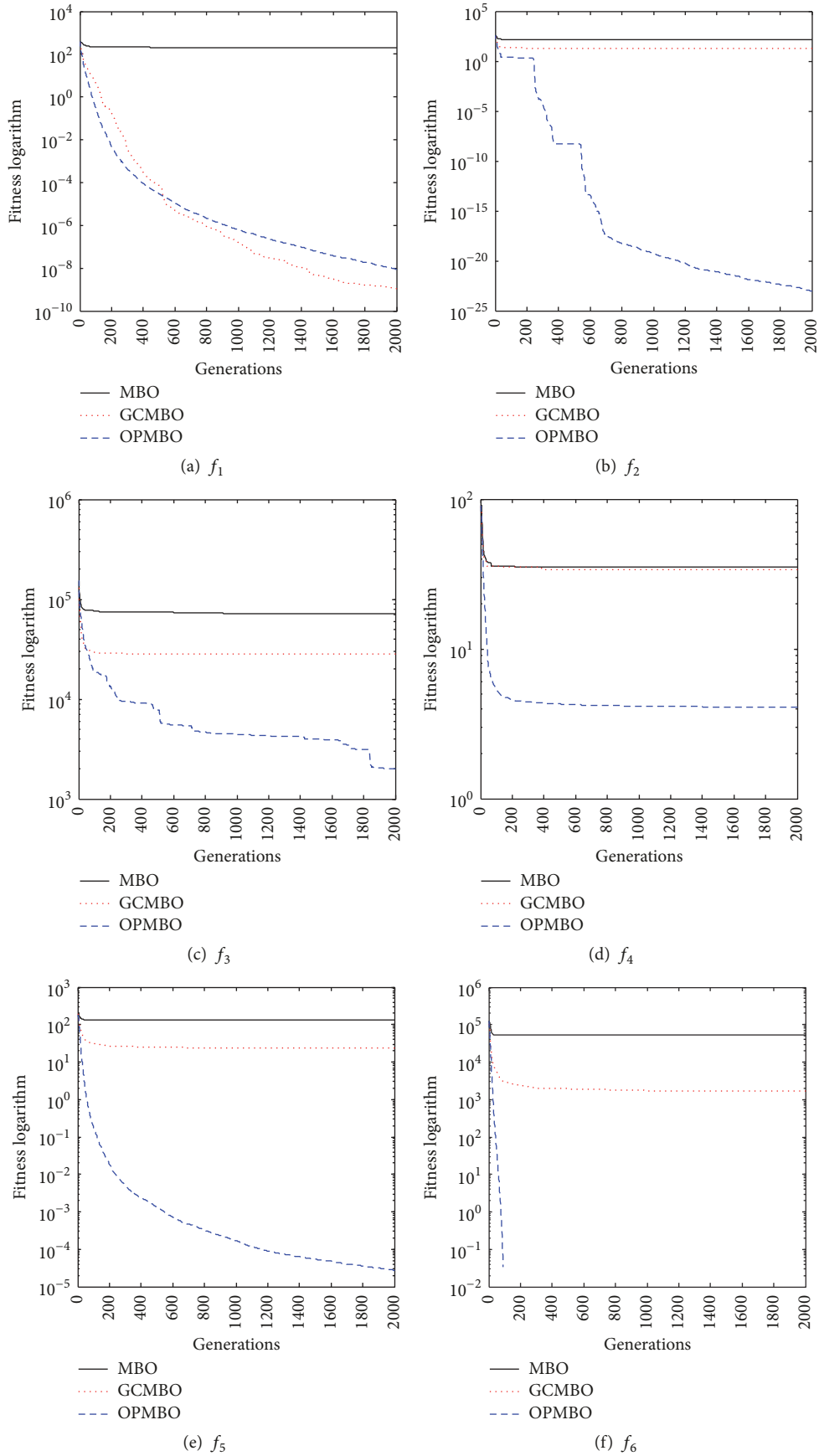

Figure 2: Continued.

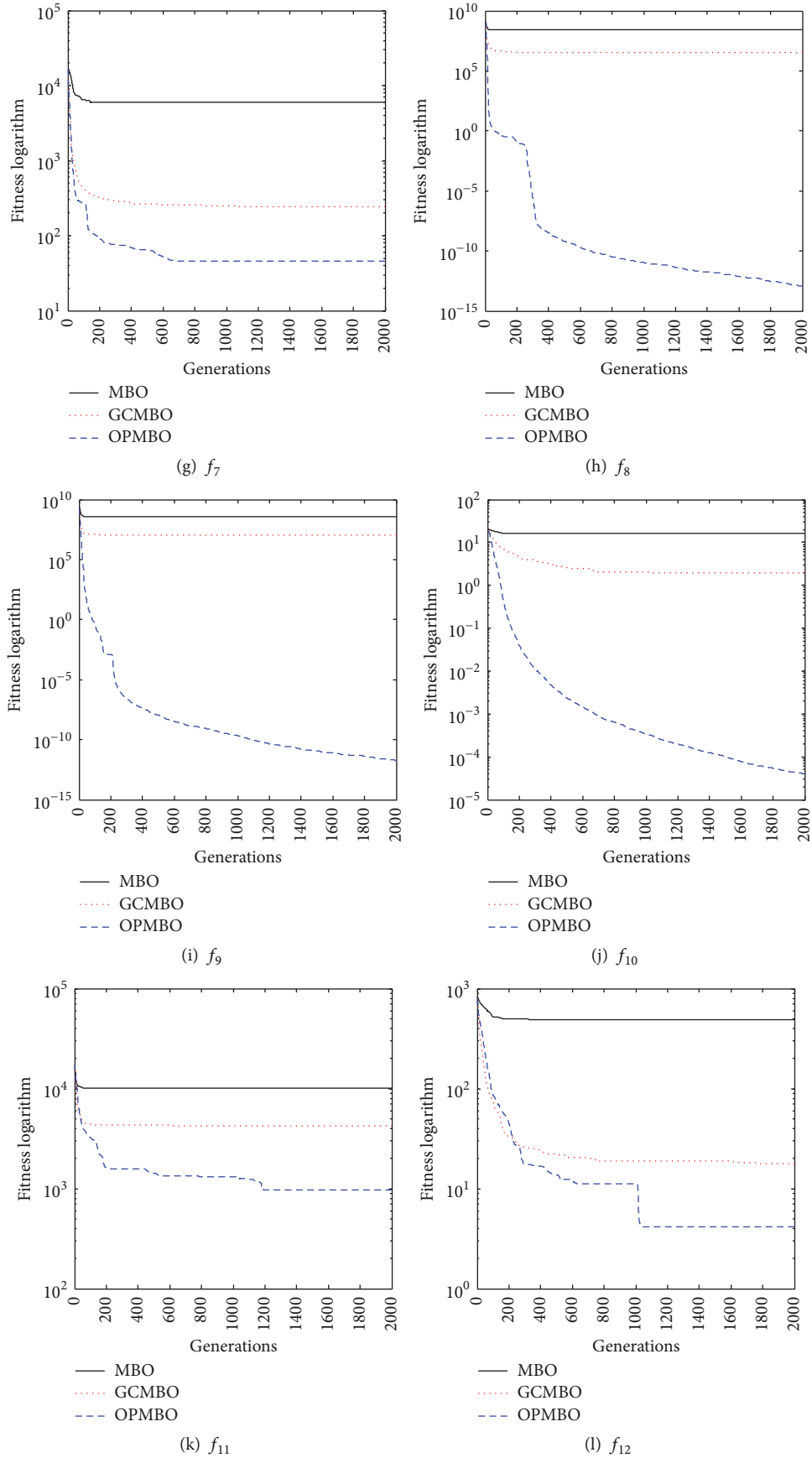

Figure 2: The convergence curves of the three algorithms on the 50-dimensional functions.

Table 2: The related parameter values for testing the three algorithms.

Dimension	*D*	*MaxGen*	*NP*	*Num*
low	20	500	50	30
medium	50	2000	50	30
high	100	5000	50	30

Table 3: The comparison results of the three algorithms on the 20-dimensional benchmark functions.

Function	Algorithm	*Best*	*Worst*	*Mean*	*Std*
f_1	MBO	2.1817e-08	1056671	10.2897	24.1542
	GCMBO	1.4305e-12	2.2191e-08	4.0376e-09	6.1440e-09
	OPMBO	**1.7124e-18**	**2.8958e-09**	**1.9961e-10**	**5.6832e-10**
f_2	MBO	2.0696e-05	45.8467	14.1370	14.5827
	GCMBO	8.5651e-31	2.7743	0.1784	0.5239
	OPMBO	**1.1985e-32**	**1.9515e-25**	**9.6741e-27**	**3.6316e-26**
f_3	MBO	467.5414	2.6400e+04	9.9164e+03	7.0557e+03
	GCMBO	5.2744	1.1659e+04	5.7437e+03	3.6304e+03
	OPMBO	**4.7218e-16**	**105.8061**	**12.9713**	**25.4996**
f_4	MBO	0.0656	69.6874	34.4202	23.9322
	GCMBO	0.8478	65	21.8416	18.9365
	OPMBO	**6.4794e-08**	**9.7491**	**1.3394**	**2.5359**
f_5	MBO	2.4859e-04	76.0287	19.5472	24.7662
	GCMBO	**0**	18.8264	2.2732	5.8333
	OPMBO	1.2577e-10	**6.6586e-06**	**1.7297e-06**	**1.8097e-06**
f_6	MBO	**0**	38189	8.2961e+03	1.3038e+04
	GCMBO	**0**	2463	9.72000	449.5788
	OPMBO	**0**	**0**	**0**	**0**
f_7	MBO	3.2274e-06	3.0575e+03	636.4260	1.0172e+03
	GCMBO	**1.7541e-07**	342.4249	71.0668	96.0405
	OPMBO	1.7164e-05	**18.6066**	**14.1968**	**7.9660**
f_8	MBO	6.8753e-12	3.3320e+08	4.1706e+07	8.7357e+07
	GCMBO	1.0427e-14	4.4128e+06	1.6156e+05	8.0584e+05
	OPMBO	**1.5705e-32**	**1.0369e-12**	**7.8430e-14**	**2.4871e-13**
f_9	MBO	4.3862e-10	4.9682e+08	5.5076e+07	1.3686e+08
	GCMBO	2.3991e-14	1.1729e+07	7.9503e+05	2.5028e+06
	OPMBO	**5.6159e-28**	**5.2997e-12**	**3.5047e-13**	**1.0677e-12**
f_{10}	MBO	5.0008e-04	20.1443	8.9263	7.9682
	GCMBO	1.9918e-06	5.5369	0.2610	1.0806
	OPMBO	**1.6130e-10**	**1.6511e-05**	**6.9485e-06**	**5.0036e-06**
f_{11}	MBO	2.5817e-04	4.8952e+03	2.3067e+03	1.6744e+03
	GCMBO	**2.5455e-04**	3.0729e+03	1.1757e+03	1.0464e+03
	OPMBO	**2.5455e-04**	**2.3009e+03**	**76.6964**	**420.0823**
f_{12}	MBO	1.1262e-04	255.8194	97.2892	96.4221
	GCMBO	1.9541e-08	74.1824	10.7346	15.3041
	OPMBO	**0**	**2.4978e-07**	**2.9660e-08**	**5.3340e-08**

has a very fast speed in convergence. For f_6, combining the convergence graph with Table 4, the OPMBO has obtained an ideal optimal value when iterating 100 times. For f_8, although the OPMBO appears to stagnate at 100 iterations, the OPMBO jumps out of the local optimum at 200 iterations. The reason is that the migration operator with RLP of OPMBO makes the algorithm jump out of local optimum to some extent. For f_9-f_{10}, the convergence of OPMBO is much better than those of MBO and GCMBO. Though the convergence of OPMBO is better than MBO and GCMBO on f_{11} and f_7, it is not as well as f_{10}. Thus, the convergence speed of OPMBO is significantly faster than those of the other algorithms, especially on f_4-f_6 and f_9. The convergence of GCMBO is better than those of MBO and OPMBO in

Table 4: The comparison results of the three algorithms on the 50-dimensional benchmark functions.

Function	Algorithm	*Best*	*Worst*	*Mean*	*Std*
f_1	MBO	0.1477	398.2232	209.3775	139.9068
	GCMBO	**1.8428e-13**	**1.0670e-08**	**1.1156e-09**	**2.1500e-09**
	OPMBO	1.5567e-10	2.7548e-08	8.8554e-09	7.5075e-09
f_2	MBO	0.0034	415.5476	153.3220	142.7727
	GCMBO	1.9171e-07	133.2460	19.7500	29.8177
	OPMBO	**1.2181e-28**	**6.6022e-23**	**9.4507e-24**	**1.6826e-23**
f_3	MBO	0.0010	1.6716e+05	7.1776e+04	5.1240e+04
	GCMBO	2.3034e+03	7.4799e+04	2.8123e+04	1.7575e+04
	OPMBO	**9.7152e-10**	**4.7017e+04**	**1.9873e+03**	**8.6490e+03**
f_4	MBO	0.0071	90.5485	35.3305	26.7642
	GCMBO	0.0299	85.3000	33.9794	21.5405
	OPMBO	**1.4047e-06**	**22.7838**	**4.0685**	**6.5366**
f_5	MBO	0.1832	227.1000	130.7104	79.0919
	GCMBO	**0**	110.2839	23.6658	35.6746
	OPMBO	1.0957e-06	**7.7733e-05**	**2.7397e-05**	**1.6504e-05**
f_6	MBO	**0**	132209	5.1815e+04	5.1859e+04
	GCMBO	**0**	12346	1696	2.9548e+03
	OPMBO	**0**	**0**	**0**	**0**
f_7	MBO	1.3455e-04	1.8360e+04	5.9326e+03	6.2480e+03
	GCMBO	**3.3925e-09**	1.4881e+03	243.2459	353.8553
	OPMBO	1.2070e-05	**306.7838**	**46.0623**	**53.5043**
f_8	MBO	3.4589e-11	1.1141e+09	2.4720e+08	4.0804e+08
	GCMBO	2.0184e-17	5.4740e+07	3.2179e+06	1.0925e+07
	OPMBO	**4.5879e-19**	**9.7859e-13**	**1.1863e-13**	**2.1688e-13**
f_9	MBO	6.5719e-10	1.9497e+09	3.6131e+08	6.4416e+08
	GCMBO	9.8724e-16	1.4513e+08	1.0977e+07	3.1121e+07
	OPMBO	**1.2994e-16**	**9.2119e-12**	**1.9405e-12**	**2.1819e-12**
f_{10}	MBO	3.0017	20.7867	15.9858	6.1167
	GCMBO	**6.6066e-07**	19.8083	1.9658	4.6606
	OPMBO	1.0861e-06	**9.8438e-05**	**4.0379e-05**	**2.2417e-05**
f_{11}	MBO	2.4511e+03	1.6770e+04	1.0050e+04	4.1486e+03
	GCMBO	**6.3638e-04**	8.0438e+03	4.2775e+03	2.7433e+03
	OPMBO	**6.3638e-04**	**6.0967e+03**	**970.1854**	**2.2072e+03**
f_{12}	MBO	26.7258	851.3496	489.7041	275.3819
	GCMBO	**1.9838e-10**	**98.6456**	17.8949	31.4580
	OPMBO	7.4206e-09	124.9711	**4.1657**	**22.8165**

early iterations, whereas the convergence of OPMBO is faster than those of the other algorithms in the later iterations on f_{12}. As a result, the OPMBO algorithm outperforms the MBO and GCMBO algorithms from the convergence curves. In conclusion, the performance of OPMBO on the medium-dimensional benchmark function is excellent for the experimental results in Table 4.

The third part of this experiment is to be carried out on the 100 dimensions, and the results are demonstrated in Table 5. The OPMBO algorithm achieves the excellent performance from Table 5. Although OPMBO is not as good as GCMBO for the *Std* on f_{11}, it is the best in terms of both the *Mean* and *Std* values on the other functions. Most importantly, the performance of OPMBO does not decrease with increasing dimensions on f_6. Therefore, OPMBO can obtain the best optimization performance on the high-dimensional.

From Tables 3–5, the GCMBO and OPMBO algorithms have similar results in three different dimensions on the identical benchmark functions, and both are better than MBO on f_1. For f_2, the results of OPMBO in all three dimensions are the best. In particular, there is no decline in optimization performance as the dimension rises in the 20-dimensional and 50-dimensional functions. On f_3 and f_4, although the OPMBO has achieved the best results, the gap between them is not obvious. The results of OPMBO are superior to the other two algorithms, especially in the 20-dimensional and 50-dimensional functions on f_5. It is worth

Table 5: The comparison results of the three algorithms on the 100-dimensional benchmark functions.

Function	Algorithm	*Best*	*Worst*	*Mean*	*Std*
	MBO	2.9655e-04	873.4768	483.6526	327.2392
f_1	GCMBO	**1.6826e-11**	200.0985	10.0288	40.1674
	OPMBO	1.4014e-08	**138.6132**	**4.8784**	**25.2979**
	MBO	0.0082	1.8812e+03	942.8625	653.6147
f_2	GCMBO	4.6673e-05	552.4148	139.0795	154.2869
	OPMBO	**2.7599e-23**	**301.3835**	**15.2540**	**59.7389**
	MBO	326.4544	8.2019e+05	3.6314e+05	1.7811e+05
f_3	GCMBO	4.5214e+03	2.4929e+05	1.1272e+05	6.3874e+04
	OPMBO	**3.8963e-08**	**1.4750e+05**	**9.4295e+03**	**2.8488e+04**
	MBO	2.3891	90.9853	38.9181	25.9349
f_4	GCMBO	0.7494	90.9000	40.2048	27.9953
	OPMBO	**1.0715e-05**	**33.5396**	**7.8364**	**9.4976**
	MBO	0.0807	484.7000	266.4036	175.1478
f_5	GCMBO	**5.0299e-07**	250.5469	56.1723	71.6008
	OPMBO	5.4517e-05	**2.2668**	**0.0760**	**0.4138**
	MBO	433	288070	1.7473e+05	1.0058e+05
f_6	GCMBO	**0**	106666	1.7420e+04	2.8532e+04
	OPMBO	**0**	**3**	**0.1000**	**0.5477**
	MBO	0.1632	4.2320e+04	1.3569e+04	1.4322e+04
f_7	GCMBO	**1.0042e-08**	8.8990e+03	1.0016e+03	2.0665e+03
	OPMBO	1.9180e-05	**6.8818e+03**	**432.4594**	**1.2535e+03**
	MBO	4.5486e-11	2.9628e+09	8.5431e+08	1.1962e+09
f_8	GCMBO	**1.1964e-19**	3.6385e+08	4.6229e+07	9.5275e+07
	OPMBO	8.3732e-13	**5.9931e-12**	**2.9010e-12**	**1.4209e-12**
	MBO	0.9237	5.0861e+09	1.4982e+09	1.7699e+09
f_9	GCMBO	**3.2011e-15**	9.8268e+08	1.3463e+08	2.7277e+08
	OPMBO	9.2390e-12	**0.4056**	**0.0135**	**0.0741**
	MBO	3.7907	20.8227	18.6499	3.9959
f_{10}	GCMBO	**5.7380e-07**	19.9186	8.6313	9.1342
	OPMBO	4.5998e-05	**1.0344**	**0.0393**	**0.1897**
	MBO	8.2058e+03	3.6580e+04	2.3230e+04	8.9660e+03
f_{11}	GCMBO	**0.0013**	**1.8954e+04**	9.3635e+03	**5.8928e+03**
	OPMBO	**0.0013**	3.1905e+04	**6.5790e+03**	1.0764e+04
	MBO	262.6423	1.8211e+03	1.4277e+03	422.1270
f_{12}	GCMBO	**7.4610e-09**	670.2994	113.0140	155.9463
	OPMBO	2.1383e-06	**665.6871**	**57.3632**	**152.5890**

mentioning that OPMBO has achieved an ideal optimal value of f_6 in three dimensions, which can prove the excellent performance of OPMBO. On f_8, the *Mean* of OPMBO in 20 dimensions is 7.8430e-14, the *Mean* in 50 dimensions is 1.1863e-13, and the *Mean* in 100 dimensions is 2.9010e-12. It follows that the optimization performance of OPMBO does not decrease as the dimension increases. So, the results of the OPMBO are much better than the other two algorithms, regardless of the three dimensions on f_9 and f_{10}. On f_{11}, OPMBO is better than the other two algorithms, except for 100 dimensions. Furthermore, the *Mean* and *Std* values of OPMBO are the best in three dimensions on f_7 and f_{12}. Therefore, the results show that the performance of OPMBO is more outstanding than the other two algorithms on most of the 12 benchmark functions under three different dimensions.

According to the above experimental results and analysis, it can be proven that OPMBO not only has great convergence accuracy but also shows better global search capabilities on multimodal functions of three different dimensions, and it is superior to the contrast algorithms with relatively obvious advantages. Since the OPMBO embeds the OBL, the diversity of population has been enhanced and the global exploration ability has been improved. Moreover, the RLP is added to the migration operator such that the information of the better individuals in the population is applied effectively, and the convergence rate of the OPMBO algorithm is greatly improved. In summary, OPMBO has better optimization

Table 6: The two-tailed t-test of the experimental results on 100-dimensional benchmark functions.

Function	OPMBO with MBO		OPMBO with GCMBO	
	t	p	t	p
f_1	791.0854	0	8.8112	1.4443e-18
f_2	946.4590	0	112.1480	0
f_3	692.0171	0	165.3924	0
f_4	395.9505	0	431.6215	0
f_5	938.4122	0	157.6091	0
f_6	1.2251e+03	0	97.7934	0
f_7	353.7764	0	15.2470	6.6025e-52
f_8	590.1854	0	32.2386	5.0606e-217
f_9	546.0571	0	48.7033	0
f_{10}	715.0020	0	272.3707	0
f_{11}	298.9920	0	16.7654	3.0813e-62
f_{12}	618.5786	0	10.6642	2.0751e-26
Better	12		12	
Equal	0		0	
Worse	0		0	

Table 7: The comparison results of OPMBO with HFOA on 30 dimensions with a population size of 50.

Function	Value	HFOA	OPMBO
f_1	*Mean*	1.30e−05	**1.2160e-08**
	Std	6.37e−07	**1.4345e-08**
f_5	*Mean*	1.82e−02	**2.3977e-05**
	Std	5.45e−04	**2.1619e-05**
f_6	*Mean*	**0**	**0**
	Std	**0**	**0**
f_{10}	*Mean*	2.65e−03	**2.7394e-06**
	Std	5.67e−05	**3.7822e-06**
f_{12}	*Mean*	2.60e−03	**6.2990e-05**
	Std	1.34e−04	**4.4079e-05**

performance than the other two algorithms whether it is on the unimodal function or the multimodal function. Therefore, all the experimental results show that the OPMBO algorithm is effective and feasible.

4.3. Two-Tailed t-Test of the Experimental Results on 100-Dimensional Benchmark Functions. For the experimental results of the 100 dimensions on the 12 benchmark functions in Table 5, following the experimental techniques designed by Zhang et al. [61], Table 6 shows the values of t and p for a two-tailed t-test with a 5% level of significance between the OPMBO algorithm and the other optimization algorithms (MBO and GCMBO), in which "Better", "Equal", and "Worse" indicate that OPMBO is better than, equal to, or worse than the compared methods in this case, respectively.

Table 6 shows that OPMBO is significantly better than MBO and GCMBO at a 95% confidence level, which demonstrates the excellent performance of the OPMBO algorithm.

4.4. Comparison of OPMBO with HFOA on 30 Dimensions and Population Size of 50. This portion of our experiments further concerns the optimization performance of OPMBO on the 30 dimensions and the population size of 50. The OPMBO algorithm is compared with the HFOA algorithm in [60]. Both algorithms have the 30 dimensions on the representative 5 benchmark functions selected from Table 1, and the population size is set as 50. Following the experimental techniques in [60], the results are shown in Table 7, where the bold font indicates the best.

Both HFOA and OPMBO obtain a theoretical optimal value on f_6. However, on the other 4 benchmark functions, OPMBO obviously outperforms the HFOA on the *Mean* and *Std* values. Hence, the results indicate that the optimization ability of OPMBO is strong. In summary, the OPMBO algorithm outperforms the HFOA algorithm on 30 dimensions and population size of 50.

4.5. Comparison of OPMBO with GWO and OGWO on 30 Dimensions and Population Size of 20. To further illustrate the optimization performance of the OPMBO algorithm, similar to Section 4.4, the OPMBO is compared with the GWO algorithm [16] and the OGWO algorithm [54]. Both OPMBO and OGWO use the OBL method and have strong comparability. Three algorithms have the 30 dimensions

Table 8: The comparison of OPMBO with two GWOs on 30 dimensions with a population size of 20.

Function	Value	GWO	OGWO	OPMBO
f_1	*Mean*	**0**	**0**	1.2050e-11
	Std	**0**	**0**	1.7997e-11
f_2	*Mean*	1.1120e-03	4.6172e-04	**1.5053e-29**
	Std	7.6128e-04	5.6910e-04	**2.5417e-29**
f_4	*Mean*	**0**	**0**	1.5287
	Std	**0**	**0**	2.2393
f_7	*Mean*	2.6505e+01	**2.4281e+01**	31.0143
	Std	5.2458e-01	**4.6388e-01**	38.9259
f_8	*Mean*	4.6138e-02	1.7570e-06	**1.4067e-15**
	Std	2.1483e-02	5.9634e-07	**3.6039e-15**

on the representative 5 benchmark functions selected from Table 1, and the population size is 20. Following the experimental techniques designed by Mirjalili et al. [16] and Zhang et al. [54], the experimental results are demonstrated in Table 8, where the best values are in bold font.

Table 8 shows that although the results of OPMBO are not as well as those of GWO and OGWO on f_1, f_4, and f_7, OPMBO is better than GWO and OGWO on f_2 and f_8. In conclusion, the optimization performance of OPMBO is better than those of GWO and OGWO on benchmark functions.

4.6. Comparison of Clustering Optimization with Six SIO Algorithms. Clustering is a way of the grouping of objects or data according to similar criteria, which is done by a group of data items close to each other based on several criteria [62]. Data clustering refers to maximizing the similarity of the members in a group (or cluster) as much as possible and minimizing the similarity of the members in two different groups as much as possible [53]. One of the well-known techniques to handle the clustering problem is to convert the clustering problem into an optimization problem. Then, the clustering problem can be solved by any optimization algorithm [63]. That is, clustering itself can be stated as an optimization problem, so it can be solved by the SIO algorithms [54]. Many scholars have been devoted to solving clustering problem using SIO techniques [62–70].

For a data space, let X be the data set with n number of objects or patterns, and each object is of m dimension, where $X = \{X_1, X_2, \ldots, X_n\}$, and $X \in R^{n\times m}$. Then, a clustering C can be represented as K clusters $\{G_1, G_2, \ldots, G_k\}$, such that it must satisfy the following conditions:

(1) $\bigcup_{i=1}^{K} G_i = X$, where G_i refers to the i^{th} cluster.

(2) $G_i \neq \emptyset$, where $i = 1, 2, \ldots, K$.

(3) $G_i \cap G_j = \emptyset$, where $i \neq j$, $i, j = 1, 2, \ldots, K$.

(4) $sim(X_1, X_2) > sim(X_1, Y_1)$, where $X_1, X_2 \in G_i$ and $Y_1 \in G_j$, $i \neq j$.

It is known that the clustering can be achieved by various well-known similarity measures. In this experiment, when the numeric data is analyzed, the Euclidean distance is introduced to measure the similarity degree between two objects. Since each individual of OPMBO is viewed as a candidate solution for the clustering optimization problem, a fitness value of the individual is used as an objective function value of the corresponding candidate solution. What is more, when the solution minimizes the objective function value, the best can be achieved. Then, the formula of the objective function [53] is described as

$$f = \sum_{k=1}^{K} \sum_{X_i \in G_k} d\left(X_i, Z_k\right), \tag{6}$$

where Z_k represents the kth clustering center, and $d(X_i, Z_k) = \|X_i - Z_k\|$ indicates the Euclidean distance between two m dimensional objects X_i and Z_k.

In Table 9, 13 data sets are adopted to test the clustering optimization performance of our proposed algorithm. The specifications of the 13 data sets are shown in Table 9, in which Mydata can be downloaded at http://yarpiz.com/64/ypml101-evolutionary-clustering, and the other 12 data sets are taken from UCI Machine Learning Repository, which can be downloaded at http://archive.ics.uci.edu/ml/datasets.htm.

The first section of this experiment is to illustrate the efficiency of clustering optimization on the selected 10 data sets from Table 9 with three MBO algorithms, which include the MBO, GCMBO, and OPMBO algorithms. The common parameter values of all three algorithms are set as follows: the population size N is 50, the maximum number of iterations *MaxGen* is 200, and the independent run number on each data set is 30. Following the experimental techniques designed in [29, 36], the *Mean* indicates the average minimum distance between each spatial point and the center of each cluster, the *Std* describes the standard deviation value, and then the comparison results are shown in Table 10, where the bold font indicates the best.

From Table 10, the OPMBO obtains the best *Mean* and *Std* values on all the 10 data sets. The *Mean* values of OPMBO are about nearly 5-6% larger than GCMBO and nearly 4-7% on the Heart, Liver Disorders, and Statlog data sets, respectively. However, the different values of *Std* with the OPMBO, MBO, and GCMBO algorithms on the three data sets are approximately 1%. Although the OPMBO achieves the largest values of *Mean* and *Std* on the other seven data sets, it is obvious that the values of OPMBO compared with MBO and GCMBO are very close.

Table 9: Overview of the thirteen data sets.

Data set	Samples	Attributes	Clusters
Mydata	300	2	3
Abalone	4177	8	3
Balance Scale	625	4	3
Car	1728	6	4
Heart	270	13	2
Liver Disorders	345	7	2
Soy bean	47	35	4
Statlog	270	13	2
Yeast	2417	103	14
Zoo	101	17	7
Iris	150	4	3
Wine	178	13	3
Glass	214	9	6

Table 10: The comparison results of clustering optimization on the three different MBO algorithms.

Dataset	Value	MBO	GCMBO	OPMBO
Mydata	*mean*	1.8159	1.7644	**1.5715**
	std	0.1740	0.1710	**0.1382**
Abalone	*mean*	1.8487	1.8430	**1.6846**
	std	0.1259	0.1135	**0.0904**
Balance Scale	*mean*	2.3613	2.3647	**2.3250**
	std	0.0237	0.0238	**0.0147**
Car	*mean*	2.0465	2.0478	**2.0125**
	std	0.0227	0.0214	**0.0193**
Heart	*mean*	52.6973	53.0394	**48.2396**
	std	3.5966	3.5950	**3.3478**
Liver Disorders	*mean*	46.1680	44.4722	**39.9735**
	std	5.0187	4.2370	**3.5294**
Soy bean	*mean*	3.5825	3.6082	**3.3619**
	std	0.0852	0.0891	**0.0715**
Statlog	*mean*	92.6813	92.3759	**86.4422**
	std	12.2853	12.2129	**10.6546**
Yeast	*mean*	0.3879	0.4006	**0.3579**
	std	0.0352	0.0402	**0.0286**
Zoo	*mean*	2.3883	2.3986	**2.3412**
	std	0.0725	0.0716	**0.0666**

The following part of this experiment further concerns the clustering optimization performance on the selected three data sets from Table 9 with the four kinds of different optimization algorithms, which include the OPMBO, the PSO algorithm [5], the FA algorithm [42, 69], and the cuckoo search (CS) algorithm [70]. The related parameter values of OPMBO are the same as [5, 69, 70], where the population size N is 40, the maximum number of iterations *MaxGen* is 200, and the independent run number on each data set is 20. Following the experimental techniques designed in [5, 42, 69, 70], similar to Section 4.2, the *Best*, *Worst*, *Mean*, and *Std* of the fitness values can be obtained. The comparison results are illustrated in Table 11, where the bold font indicates the best.

It can be observed from Table 11 that OPMBO exhibits the largest values of *Best*, *Worst*, *Mean*, and *Std* on the Wine and Glass data sets, except for Iris. For Wine, the values of *Best*, *Worst*, *Mean*, and *Std* achieved by the PSO, FA, and CS algorithms are 1000 times larger than those of the OPMBO algorithm. For Glass, the values of *Best*, *Worst*, and *Mean* achieved by the FA are 400-450 larger than those of OPMBO. However, the value of *Std* of OPMBO is close to that of FA. Meanwhile, the value of *Std* of the PSO is nearly 100 larger than that of OPMBO. For Iris, the value of *Std* of the OPMBO is 3-8 less than that of PSO, FA, and CS. However, the values of *Best*, *Worst*, and *Mean* obtained by the OPMBO are 40-80 larger than those of PSO, FA, and CS. The reason is that the Iris data set may have too few attributes so that the OPMBO

Table 11: The comparison of the clustering optimizations on the four different optimization algorithms.

Data set	Algorithm	*Best*	*Worst*	*Mean*	*Std*
Iris	PSO	**80.576**	153.232	116.128	22.435
	FA	81.176	**152.347**	106.476	19.547
	CS	80.647	152.348	**100.45**	17.623
	OPMBO	142.8859	199.5001	184.8284	**14.7271**
Wine	PSO	16529651	17292036	17019639	185396
	FA	16695182	17963968	16962938	172644
	CS	16209233	17325656	16940862	262499
	OPMBO	**1.6856e+04**	**2.0190e+04**	**1.8219e+04**	**913.9988**
Glass	PSO	687.347	1140.445	882.098	115.025
	FA	1023.533	1110.698	1079.931	22.597
	CS	729.658	1095.603	957.901	81.275
	OPMBO	**612.3864**	**688.3899**	**653.4392**	**21.1229**

algorithm performs very poorly in *Mean*. In general, the OPMBO algorithm can obtain satisfactory results in solving the clustering optimization problems.

5. Conclusion and Future Work

Recently, SIO algorithms have been widely employed for solving complex optimization problems. MBO as one of the most promising SIO methods is proposed to tackle global optimization problems. In order to improve the optimization efficiency of MBO algorithm and obtain an algorithm with strong universal applicability, this paper introduces OBL into MBO and proposes RLP to improved basic MBO algorithm. The improvements are mainly from two aspects, one is to enhance the optimization performance and the other is to reduce the computation complexity. The OBL firstly prevents the algorithm from falling into the local optima to some degree. A new RLP is merged into the migration operator, which can enhance the local search ability and accelerate the convergence speed. Then, the greedy strategy is used instead of the elitist strategy, and it can decrease the setting of elite parameters and eliminate a sorting operation. Finally, the OPMBO algorithm is proposed, and to verify the optimization performance of OPMBO, a series of experiments are performed on 12 benchmark functions and 13 public data sets. The results verify that our OPMBO algorithm typically outperforms the other algorithms considered. Based on the comparison and analysis of our scheme with other schemes, the contribution of our proposed method can be summarized as follows:

(1) The OBL as a widely used technique in optimization algorithms is introduced into the MBO algorithm, an opposite candidate solution generated by OBL has a better chance to be closer to the global optimum solution than a random candidate solution, and then this process can efficiently avoid the MBO from falling into a local optimum.

(2) The RLP is proposed to merge into the migration operator, which shares the information of the optimal solution and the suboptimal solution, and it is helpful to guide the current individual to move toward the optimal solution and the suboptimal solution. Then, the premature convergence of MBO can be eliminated effectively.

(3) A greedy strategy is introduced into the improved migration operator with RLP, and the superior candidate solution is retained by the principle of survival of the fittest. This operation eliminates the elitist parameters, gets rid of a sorting, and further improves the operation efficiency.

(4) Since each individual of OPMBO is viewed as a candidate solution for clustering optimization problem, a fitness value of the individual is used as an objective function value of the corresponding candidate solution. When the solution minimizes the objective function value, the best can be achieved. Then, the OPMBO algorithm can be applied to solve clustering optimization on public data sets.

It is well-known that there is not any SIO algorithm that can effectively solve all optimization problems, and then in our experiments the OPMBO algorithm cannot obtain the satisfactory results on some benchmark functions. In our future work, some interesting problems can be further studied. We intend to hybridize MBO with the latest SIO methods, further improve the OPMBO algorithm, and use it to resolve multiobjective optimization problems. Furthermore, the OPMBO will be applied to other engineering fields.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Grants 61772176, 61402153, 61472042, and 11601130), the China Postdoctoral Science Foundation (Grant 2016M602247), the Plan for Scientific Innovation Talent of Henan Province (Grant 184100510003),

the Key Project of Science and Technology Department of Henan Province (Grants 182102210362, 182102210078), the Young Scholar Program of Henan Province (Grant 2017GGJS041), the Key Scientific and Technological Project of Xinxiang City (Grant CXGG17002), the Ph.D. Research Foundation of Henan Normal University (Grants qd15132, qd15129), the Natural Science Foundation of Henan Province (Grants 182300410130, 182300410368), and the Natural Science Project of Henan Province Education Department (Grant 17A520039).

References

[1] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, "A hybrid particle swarm optimization algorithm using adaptive learning strategy," *Information Sciences*, vol. 436/437, pp. 162–177, 2018.

[2] X. Lv, D. Zhou, Y. Tang, and L. Ma, "An improved test selection optimization model based on fault ambiguity group isolation and chaotic discrete PSO," *Complexity*, vol. 2018, Article ID 3942723, 10 pages, 2018.

[3] L. Sun, X. Zhang, Y. Qian, J. Xu, S. Zhang, and Y. Tian, "Joint neighborhood entropy-based gene selection method with Fisher score for tumor classification," *Applied Intelligence*, 2018.

[4] L. Wang, H. Hu, X.-Y. Ai, and H. Liu, "Effective electricity energy consumption forecasting using echo state network improved by differential evolution algorithm," *Energy*, vol. 153, pp. 801–815, 2018.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Encyclopedia of Machine Learning*, pp. 760–766, 2011.

[6] Z. Zheng, N. Saxena, K. K. Mishra, and A. K. Sangaiah, "Guided dynamic particle swarm optimization for optimizing digital image watermarking in industry applications," *Future Generation Computer Systems*, vol. 88, pp. 92–106, 2018.

[7] A. Wu and Z. L. Yang, "An elitist transposon quantum-based particle swarm optimization algorithm for economic dispatch problems," *Complexity*, vol. 2018, Article ID 7276585, 15 pages, 2018.

[8] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources Planning and Management*, vol. 129, no. 3, pp. 210–225, 2003.

[9] C. Liu, P. Niu, G. Li, Y. Ma, W. Zhang, and K. Chen, "Enhanced shuffled frog-leaping algorithm for solving numerical function optimization problems," *Journal of Intelligent Manufacturing*, vol. 29, no. 5, pp. 1133–1153, 2018.

[10] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Proceedings of the International Conference in Swarm Intelligence*, pp. 355–364, 2010.

[11] S. Gholizadeh and A. Milany, "An improved fireworks algorithm for discrete sizing optimization of steel skeletal structures," *Engineering Optimization*, vol. 50, no. 11, pp. 1829–1849, 2018.

[12] X. Yin, X. Wei, L. Liu, and Y. Wang, "Improved hybrid fireworks algorithm-based parameter optimization in high-order sliding mode control of hypersonic vehicles," *Complexity*, vol. 2018, Article ID 9098151, 16 pages, 2018.

[13] A. H. Gandomi and A. H. Alavi, "Krill herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.

[14] R. Jensi and G. W. Jiji, "An improved krill herd algorithm with global exploration capability for solving numerical function optimization problems and its application to data clustering," *Applied Soft Computing*, vol. 46, pp. 230–245, 2016.

[15] P. Singh and B. Khan, "Smart microgrid energy management using a novel artificial shark optimization," *Complexity*, vol. 2017, Article ID 2158926, 22 pages, 2017.

[16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[17] W.-L. Mao, Suprapto, and C.-W. Hung, "Type-2 fuzzy neural network using grey wolf optimizer learning algorithm for nonlinear system identification," *Microsystem Technologies*, vol. 24, no. 10, pp. 4075–4088, 2018.

[18] C. Lu, L. Gao, and J. Yi, "Grey wolf optimizer with cellular topological structure," *Expert Systems with Applications*, vol. 107, pp. 89–114, 2018.

[19] 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.

[20] N. Noman and H. Iba, "Differential evolution for economic load dispatch problems," *Electric Power Systems Research*, vol. 78, no. 8, pp. 1322–1331, 2008.

[21] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 2018.

[22] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, "Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations," *Computers & Operations Research*, vol. 67, pp. 155–173, 2016.

[23] W.-T. Pan, "A new fruit fly optimization algorithm: taking the financial distress model as an example," *Knowledge-Based Systems*, vol. 26, pp. 69–74, 2012.

[24] L. Wang, R. Liu, and S. Liu, "An effective and efficient fruit fly optimization algorithm with level probability policy and its applications," *Knowledge-Based Systems*, vol. 97, pp. 158–174, 2016.

[25] L. Wang, S. Lv, and Y. Zeng, "Effective sparse adaboost method with ESN and FOA for industrial electricity consumption forecasting in China," *Energy*, vol. 155, pp. 1013–1031, 2018.

[26] Y. Mokhtari and D. Rekioua, "High performance of Maximum Power Point Tracking Using Ant Colony algorithm in wind turbine," *Journal of Renewable Energy*, vol. 126, pp. 1055–1063, 2018.

[27] F. Min, Z.-H. Zhang, and J. Dong, "Ant colony optimization with partial-complete searching for attribute reduction," *Journal of Computational Science*, vol. 25, pp. 170–182, 2018.

[28] O. Abedinia, N. Amjady, and A. Ghasemi, "A new metaheuristic algorithm based on shark smell optimization," *Complexity*, vol. 21, no. 5, pp. 97–116, 2016.

[29] G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing and Applications*, 2015.

[30] S. Chen, R. Chen, and J. Gao, "A monarch butterfly optimization for the dynamic vehicle routing problem," *Algorithms*, vol. 10, no. 3, pp. 1–19, 2017.

[31] Y. Feng, G.-G. Wang, S. Deb, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problem by a novel binary monarch butterfly optimization," *Neural Computing and Applications*, vol. 28, no. 7, pp. 1619–1634, 2017.

[32] H. Faris, I. Aljarah, and S. Mirjalili, "Improved monarch butterfly optimization for unconstrained global search and neural network training," *Applied Intelligence*, vol. 48, no. 2, pp. 445–464, 2018.

[33] V. Yadav and S. P. Ghoshal, "Optimal power flow for IEEE 30 and 118-bus systems using monarch butterfly optimization," in *Proceedings of the IEEE International Conference on Technologies for Smart-City Energy Security and Power*, pp. 1–6, Bhubaneswar, India, March 2018.

[34] D. Devikanniga and R. Joshua Samuel Raj, "Classification of osteoporosis by artificial neural network based on monarch butterfly optimisation algorithm," *Healthcare Technology Letters*, vol. 5, no. 2, pp. 70–75, 2018.

[35] H. Hu, Z. Cai, S. Hu, Y. Cai, J. Chen, and S. Huang, "Improving monarch butterfly optimization algorithm with self-adaptive population," *Algorithms*, vol. 11, no. 5, article 71, 2018.

[36] G.-G. Wang, X. Zhao, and S. Deb, "A novel monarch butterfly optimization with greedy strategy and self-adaptive," in *Proceedings of the IEEE Second International Conference on Soft Computing and Machine Intelligence*, pp. 45–50, 2015.

[37] Y. Feng, J. Yang, C. Wu, M. Lu, and X.-J. Zhao, "Solving 0–1 knapsack problems by chaotic monarch butterfly optimization algorithm with Gaussian mutation," *Memetic Computing*, vol. 10, no. 2, pp. 135–150, 2018.

[38] Y. Feng, G. Wang, W. Li, and N. Li, "Multi-strategy monarch butterfly optimization algorithm for discounted (0-1) knapsack problem," *Neural Computing and Applications*, vol. 30, no. 10, pp. 3019–3036, 2018.

[39] S. M. Abd-Elazim and E. S. Ali, "A hybrid particle swarm optimization and bacterial foraging for power system stability enhancement," *Complexity*, vol. 21, no. 2, pp. 245–255, 2015.

[40] W. A. Ghanem and A. Jantan, "A novel hybrid artificial bee colony with monarch butterfly optimization for global optimization problems," in *Modeling, Simulation, and Optimization, EAI/Springer Innovations in Communication and Computing*, P. Vasant, I. Litvinchev, and J. Marmolejo-Saucedo, Eds., pp. 27–38, 2018.

[41] M. Ghetas, C. H. Yong, and P. Sumari, "Harmony-based monarch butterfly optimization algorithm," in *Proceedings of the IEEE International Conference on Control System, Computing and Engineering*, pp. 156–161, 2015.

[42] I. Strumberger, M. Sarac, D. Markovic, and N. Bacanin, "Hybridized monarch butterfly algorithm for global optimization problems," *International Journal of Computers*, vol. 3, pp. 63–68, 2018.

[43] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation*, pp. 695–701, 2005.

[44] J. L. Shang, Y. Sun, S. J. Li et al., "An improved opposition-based learning particle swarm optimization for the detection of SNP-SNP interaction," *Biomed Research International*, vol. 2015, Article ID 524821, 12 pages, 2015.

[45] T. K. Sharma and M. Pant, "Opposition based learning ingrained shuffled frog-leaping algorithm," *Journal of Computational Science*, vol. 21, pp. 307–315, 2017.

[46] M. A. Ahandani and H. Alavi-Rad, "Opposition-based learning in shuffled frog leaping: An application for parameter identification," *Information Sciences*, vol. 291, no. C, pp. 19–42, 2015.

[47] S. Yu, S. Zhu, Y. Ma, and D. Mao, "Enhancing firefly algorithm using generalized opposition-based learning," *Computing: Archives for Scientific Computing*, vol. 97, no. 7, pp. 741–754, 2015.

[48] C. Yang, J. K. Zhang, and L. X. Guo, "Investigation on the inversion of the atmospheric duct using the artificial bee colony algorithm based on opposition-based learning," *International Journal of Antennas and Propagation*, vol. 2016, Article ID 2749035, 10 pages, 2016.

[49] X. Shan, K. Liu, and P. L. Sun, "Modified bat algorithm based on lévy flight and opposition based learning," *Scientific Programming*, vol. 2016, Article ID 8031560, 13 pages, 2016.

[50] S.-Y. Park and J.-J. Lee, "Stochastic opposition-based learning using a beta distribution in differential evolution," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2184–2194, 2016.

[51] Y. Kumar and G. Sahoo, "An improved cat swarm optimization algorithm based on opposition-based learning and Cauchy operator for clustering," *Journal of Information Processing Systems*, vol. 13, no. 4, pp. 1000–1013, 2017.

[52] R. Sarkhel, N. Das, A. K. Saha, and M. Nasipuri, "An improved Harmony Search Algorithm embedded with a novel piecewise opposition based learning algorithm," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 317–330, 2018.

[53] X. M. Zhang, Q. Kang, Q. Tu et al., "Efficient and merged biogeography-based optimization algorithm for global optimization problems," *Soft Computing*, 2018.

[54] X. Zhang, Q. Kang, J. Cheng, and X. Wang, "A novel hybrid algorithm based on Biogeography-Based Optimization and Grey Wolf Optimizer," *Applied Soft Computing*, vol. 67, pp. 197–214, 2018.

[55] B. Mandal and P. K. Roy, "Optimal reactive power dispatch using quasi-oppositional teaching learning based optimization," *International Journal of Electrical Power & Energy Systems*, vol. 53, no. 1, pp. 123–134, 2013.

[56] L. Wang, Y. Shi, and S. Liu, "An improved fruit fly optimization algorithm and its application to joint replenishment problems," *Expert Systems with Applications*, vol. 42, no. 9, pp. 4310–4323, 2015.

[57] M. D. Li, H. Zhao, X. W. Weng, and H. Q. Huang, "Artificial bee colony algorithm with comprehensive search mechanism for numerical optimization," *Journal of Systems Engineering and Electronics*, vol. 26, no. 3, pp. 603–617, 2015.

[58] K. Yu, X. Wang, and Z. Wang, "An improved teaching-learning-based optimization algorithm for numerical and engineering optimization problems," *Journal of Intelligent Manufacturing*, vol. 27, no. 4, pp. 831–843, 2016.

[59] M.-X. Li, R.-Q. Liao, and Y. Dong, "The particle swarm optimization algorithm with adaptive chaos perturbation," *International Journal of Computers, Communications & Control*, vol. 11, no. 6, pp. 804–818, 2016.

[60] S.-X. Lv, Y.-R. Zeng, and L. Wang, "An effective fruit fly optimization algorithm with hybrid information exchange and its applications," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 10, pp. 1623–1648, 2018.

[61] X. M. Zhang, Q. Kang, X. Wang, and J. F. Cheng, "Particle swarm optimization algorithm with cross opposition learning and particle-based social learning," *Journal of Computer Applications*, vol. 37, no. 11, pp. 3194–3200, 2017.

[62] L. Sun, R. Liu, J. Xu, S. Zhang, and Y. Tian, "An affinity propagation clustering method using hybrid kernel function with LLE," *IEEE Access*, vol. 6, pp. 68892–68909, 2018.

[63] A. Bouyer and A. Hatamlou, "An efficient hybrid clustering method based on improved cuckoo optimization and modified particle swarm optimization algorithms," *Applied Soft Computing*, vol. 67, pp. 172–182, 2018.

[64] L. Sun, J. Xu, and Y. Tian, "Feature selection using rough entropy-based uncertainty measures in incomplete decision systems," *Knowledge-Based Systems*, vol. 36, pp. 206–216, 2012.

[65] P. Das, D. K. Das, and S. Dey, "A modified Bee Colony Optimization (MBCO) and its hybridization with k-means for an application to data clustering," *Applied Soft Computing*, vol. 70, pp. 590–603, 2018.

[66] L. Sun, J. Xu, S. Liu, S. Zhang, Y. Li, and C. Shen, "A robust image watermarking scheme using Arnold transform and BP neural network," *Neural Computing and Applications*, vol. 30, no. 8, pp. 2425–2440, 2018.

[67] L. Sun, X. Zhang, J. Xu, W. Wang, and R. Liu, "A gene selection approach based on the fisher linear discriminant and the neighborhood rough set," *Bioengineered*, vol. 9, no. 1, pp. 144–151, 2018.

[68] R. Wang, S. Lai, G. Wu, L. Xing, L. Wang, and H. Ishibuchi, "Multi-clustering via evolutionary multi-objective optimization," *Information Sciences*, vol. 450, pp. 128–140, 2018.

[69] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, 2008.

[70] X. S. Yang and S. Deb, "Cuckoo search via Levy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing*, pp. 210–214, 2010.