Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 1423930, 22 pages
http://dx.doi.org/10.1155/2016/1423930

Research Article

Lévy-Flight Moth-Flame Algorithm for Function Optimization and Engineering Design Problems

Zhiming Li,[1] Yongquan Zhou,[1,2] Sen Zhang,[1] and Junmin Song[1]

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

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

Received 18 April 2016; Accepted 12 July 2016

Academic Editor: Jose J. Muñoz

The moth-flame optimization (MFO) algorithm is a novel nature-inspired heuristic paradigm. The main inspiration of this algorithm is the navigation method of moths in nature called transverse orientation. Moths fly in night by maintaining a fixed angle with respect to the moon, a very effective mechanism for travelling in a straight line for long distances. However, these fancy insects are trapped in a spiral path around artificial lights. Aiming at the phenomenon that MFO algorithm has slow convergence and low precision, an improved version of MFO algorithm based on Lévy-flight strategy, which is named as LMFO, is proposed. Lévy-flight can increase the diversity of the population against premature convergence and make the algorithm jump out of local optimum more effectively. This approach is helpful to obtain a better trade-off between exploration and exploitation ability of MFO, thus, which can make LMFO faster and more robust than MFO. And a comparison with ABC, BA, GGSA, DA, PSOGSA, and MFO on 19 unconstrained benchmark functions and 2 constrained engineering design problems is tested. These results demonstrate the superior performance of LMFO.

1. Introduction

Optimization is a process of finding the best possible solution(s) for a given problem. In real world, many problems can be viewed as optimization problems. Since the complexity of problems increases, the need for new optimization techniques becomes more evident than before. Over the past several decades, some kinds of methods have been proposed to solve optimization problems and have made great progress. For example, mathematical optimization techniques used to be the only tool for optimizing problems before the proposal of heuristic optimization techniques. However, these methods need to know the property of optimization problem, such as continuity or differentiability. In recent years, metaheuristic optimization algorithms have become more and more popular in optimization techniques. Some popular algorithms in this field are Genetic Algorithms (GA) [1, 2], Particle Swarm Optimization (PSO) [3], Ant Colony Optimization (ACO) [4], Evolutionary Strategy (ES) [5], Differential Evolution (DE) [6], and Evolutionary Programming (EP) [7]. The application of these algorithms can be found in different branches of science and industry as well. Despite the merits of these optimizers, there is a fundamental question here whether there is any optimizer for solving all optimization problems. According to the No-Free-Lunch (NFL) theorem [8] for optimization, researchers are allowed to develop new algorithms solving optimization problems more effectively. Some of the latest algorithms are Artificial Bee Colony (ABC) algorithm [9], Bat Algorithm (BA) [10], Cuckoo Search (CS) algorithm [11], Cuckoo Optimization Algorithm (COA) [12], Gravitational Search Algorithm (GSA) [13], Charged System Search (CSS) [14], Firefly Algorithm (FA) [15], and Ray Optimization (RO) [16], and Dragonfly Algorithm (DA) [17].

Moth-flame optimization (MFO) [18] algorithm is a new metaheuristic optimization method through imitating the navigation method of moths in nature called transverse orientation. In this algorithm, moths and flames are both solutions. The inventor of this algorithm, Seyedali Mirjalili, proved that this algorithm is able to show very competitive results compared with other state-of-the-art metaheuristic optimization algorithms. However, the MFO algorithm has

been in research stage so far, and convergence speed and calculation accuracy of this algorithm can be further advanced. To improve the performance of MFO, a Lévy-flight moth-flame optimization (LMFO) algorithm is proposed.

We know that Lévy-flight [11, 19] has a strong ability of strengthening global search and overcoming the problem of being trapped in local minima. In order to make use of the good performance of Lévy-flight, we propose a Lévy-flight moth-flame optimization. MFO and Lévy-flight have complementary advantages, so the proposed algorithm can lead to a faster and more robust method. The proposed algorithm is verified on nineteen benchmark functions and two engineering problems.

The rest of the paper is organized as follows: Section 2 presents a brief introduction to MFO and Lévy-flight. An improved version of MFO algorithm, LMFO, is proposed in Section 3. The experimental results of test functions and engineering design problem are showed in Sections 4 and 5, respectively. Results and discussion are provided in Section 6. Finally, Section 7 concludes the work.

2. Related Works

In this section, a background about the moth-flame optimization algorithm and Lévy-flight will be provided briefly.

2.1. MFO Algorithm. Moth-flame optimization [18] algorithm is a new metaheuristic optimization method, which is proposed by Seyedali Mirjalili and based on the simulation of the behavior of moths for their special navigation methods in night. They utilize a mechanism called transverse orientation for navigation. In this method, a moth flies by maintaining a fixed angle with respect to the moon, which is a very effective mechanism for travelling long distance in a straight path because the moon is far away from the moth. This mechanism guarantees that moths fly along straight line in night. However, we usually observe that moths fly spirally around the lights. In fact, moths are tricked by artificial lights and show such behaviors. Since such light is extremely close to the moon, hence, maintaining a similar angle to the light source causes a spiral fly path of moths.

In the MFO algorithm, the set of moths is represented in a matrix M. For all the moths, there is an array OM for storing the corresponding fitness values. The second key components in the algorithm are flames. A matrix F similar to the moth matrix is considered. For the flames, it is also assumed that there is an array OF for storing the corresponding fitness values.

The MFO algorithm is a three-tuple that approximates the global optimal of the optimization problems and defined as follows:

$$\text{MFO} = (I, P, T). \tag{1}$$

I is a function that creates a random population of moths and corresponding fitness values. The methodical model of this function is as follows:

$$I : \phi \longrightarrow \{M, OM\}. \tag{2}$$

The P function, which is the main function, moves the moths around the search space. This function received the matrix of M and returns its updated one eventually:

$$P : M \longrightarrow M. \tag{3}$$

The T function returns true if the termination criterion is satisfied and false if the termination criterion is not satisfied:

$$T : M \longrightarrow \{\text{true}, \text{false}\}. \tag{4}$$

With I, P, and T, the general framework of the MFO algorithm is defined as follows:

$M = I(\,)$;
while $T(M)$ is equal to false
    $M = P(M)$;
end

After the initialization, the P function is iteratively run until the T function returns true. For the sake of simulating the behavior of moths mathematically, the position of each moth is updated with respect to a flame using the following equation:

$$M_i = S\left(M_i, F_j\right), \tag{5}$$

where M_i indicate the ith moth, F_j indicates the jth flame, and S is the spiral function.

Any types of spiral can be utilized here subject to the following conditions:

(1) Spiral's initial point should start from the moth.

(2) Spiral's final point should be the position of the flame.

(3) Fluctuation of the range of spiral should not exceed the search space.

Considering these points, a logarithmic spiral is defined for the MFO algorithm as follows:

$$S\left(M_i, F_j\right) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j, \tag{6}$$

where D_i indicates the distance of the ith moth for the jth flame, b is a constant for defining the shape of the logarithmic spiral, and t is a random number in $[-1, 1]$.

D is calculated as follows:

$$D_i = \left|F_j - M_i\right|, \tag{7}$$

where M_i indicate the ith moth, F_j indicates the jth flame, and D_i indicates the distance of the ith moth for the jth flame.

Equation (6) describes the spiral flying path of moths. From this equation, the next position of a moth is defined with respect to a flame. The t parameter in the spiral equation defines how much the next position of the moth should be close to the flame ($t = -1$ is the closest position to the flame, while $t = 1$ shows the farthest).

A question that may rise here is that the position updating in (6) only requires the moths to move towards a flame, yet

```
(1)  Initialize the position of moths
(2)  While (Iteration <= Max_iteration)
(3)  Update flame no using (8)
(4)  OM = FitnessFunction(M);
(5)  if iteration = = 1
(6)       F = sort(M);
(7)       OF = sort(OM);
(8)  else
(9)       F = sort(M_t − 1, M_t);
(10)      OF = sort(M_t−1, M_t);
(11) end
(12) for i = 1 : n
(13)      for j = 1 : d
(14)           Update r and t
(15)           Calculate D using (7) with respect to the corresponding moth
(16)           Update M(i, j) using (5) and (6) with respect to the corresponding moth
(17)      end
(18) end
```

Algorithm 1: MFO algorithm.

it causes the MFO algorithm to be trapped in local optima quickly. In order to prevent this, each moth is obliged to update its position using only one of the flames in (6). Another concern here is that the position updating of moths with respect to n different locations in the search space may degrade the exploitation of the best promising solutions. To resolve this concern, an adaptive mechanism provided the number of flames. The following formula is utilized in this regard:

$$\text{flame no} = \text{round}\left(N - l * \frac{N-1}{T}\right), \tag{8}$$

where l is the current number of iteration, N is the maximum number of flames, and T indicates the maximum number of iterations.

The gradual decrement in number of flames balances exploration and exploitation of the search space. After all, the general steps of the P function can be described in Algorithm 1.

As described in Algorithm 1, the P function is executed until the T function returns true. After termination of the P function, the best moth is returned as the best obtained approximation of the optimum.

Note that the Quicksort method is utilized in MFO and the sort's computational complexity is $O(n \log n)$ and $O(n^2)$ in the best and worst case, respectively (where n is the number of moths).

2.2. Lévy-Flight. Lévy-flight was originally introduced by the French mathematician in 1937 named Paul Lévy. Lévy-flight is a statistical description of motion that extends beyond the more traditional Brownian motion discovered over one hundred years earlier. A diverse range of both natural and artificial phenomena are now being described in terms of Lévy statistics [19].

Generally speaking, animals looking for food is random, from one place to another place. A large number of studies have shown that flight behavior of many animals and insects has demonstrated the typical characteristics of randomness. However, the choice of the direction relies only on a mathematical model [20], which is called Lévy-flight. For instance, many studies have shown that flight behavior of many animals and insects has revealed the typical characteristics of Lévy-flight [21–24]. According to [24], we can know that fruit flies or *Drosophila melanogaster* explore their landscape utilizing a series of straight flight paths punctuated by a sudden 90° turn, resulting in a Lévy-flight-style fitful scale-free pattern. Studies on human behavior such as the Ju/'hoansi hunter-gatherer foraging patterns [21] also show the typical feature of Lévy-flight. Pavlyukevich has used Lévy-flight in his research to present and theoretically justify a new stochastic algorithm for global optimization. Even the light can be related to Lévy-flight [20]. Subsequently, Lévy-flight have been applied to optimization and optimal search, and preliminary results show its promising capability [22, 25].

3. The Proposed LMFO Approach

In order to increase the diversity of population against premature convergence and accelerate the convergence speed, this paper proposes an improved Lévy-flight moth-flame optimization (LMFO) algorithm. Lévy-flight has the prominent properties to increase the diversity of population, sequentially, which can make the algorithm effectively jump out of the local optimum. In other words, this approach is beneficial to obtain a better trade-off between the exploration and exploitation ability of MFO. So, we let each moth perform once Lévy-flight using (9) after the position updating, which is formulated as follows [11, 26]:

$$X_i^{t+1} = X_i^t + u \,\text{sign}\,[\text{rand} - 0.5] \oplus \text{Levy}\,(\beta), \tag{9}$$

```
(1)  Initialize the position of moths
(2)  While (Iteration <= Max_iteration)
(3)  Update flame no using (8)
(4)  OM = FitnessFunction(M);
(5)  if iteration == 1
(6)        F = sort(M);
(7)        OF = sort(OM);
(8)  else
(9)        F = sort(M_t − 1, M_t);
(10)       OF = sort(M_t−1, M_t);
(11) end
(12) for i = 1 : n
(13)       for j = 1 : d
(14)             Update r and t
(15)             Calculate D using (7) with respect to the corresponding moth
(16)             Update M(i, j) using (5) and (6) with respect to the corresponding moth
(17)       end
(18)       for each search agent
(19)             Update the position of the current search agent using Lévy-flight
(20)       end
(21)       Iteration = Iteration + 1;
(22) end
```

Algorithm 2: LMFO algorithm.

where X_i^t is the ith moth or solution vector X_i at iteration t, u is a random parameter which conforms to a uniform distribution, $\oplus$ is the dot product (entrywise multiplications), and rand is a random number in $[0, 1]$. It should be noted here that sign[rand $-$ 0.5] takes only three values 1, 0, and -1. And in (9) the combination of u sign[rand $-$ 0.5] and Lévy-flight can make moth walk more random. That is to say, to get rid of local minima and improve global search capability are ensured via this combination in the basic MFO. Lévy-flight are a kind of random walk in which the steps are determined by the step lengths, and the jumps conform to a Lévy distribution as follows [11, 27]:

$$\text{Levy}(\beta) \sim \mu = t^{-1-\beta}, \quad (0 \le \beta \le 2). \tag{10}$$

Formula (11) is calculated as Lévy random numbers:

$$\text{Levy}(\beta) \sim \frac{\phi \times \mu}{|\nu|^{1/\beta}}, \tag{11}$$

where μ and ν are both standard normal distributions, Γ is a standard Gamma function, $\beta = 1.5$, and ϕ is defined as follows:

$$\phi = \left[\frac{\Gamma(1+\beta) \times \sin(\pi \times \beta/2)}{\Gamma(((1+\beta)/2) \times \beta \times 2^{(\beta-1)/2})} \right]^{1/\beta}. \tag{12}$$

To sum up, global search ability of the proposed algorithm is strengthened using random walk with Lévy-flight to eliminate the weakness of MFO, its being trapped in local minimum is prevented, and it is observed to give more successful results particularly for unimodal and multimodal benchmark functions. Because of these features, the proposed algorithm has potential to provide superior performance compared to MFO. In following section, all kinds of benchmark functions are hired to verify the effectiveness of the proposed algorithm. The main steps of Lévy-flight moth-flame optimization can be simply presented in Algorithm 2.

4. Simulation Experiments

4.1. Simulation Platform. All the algorithms are tested in MATLAB R2012a (7.14) and numerical experiment is set up on Intel Core (TM) i5-4590 Processor, 3.30 GHz, 4 GB RAM, running on Windows 7.

4.2. Benchmark Functions. It is common in this field to benchmark the performance of algorithm on a set of mathematical functions with known global optimal. The same process is followed, in which nineteen standard benchmark functions are employed from the literature [27, 28] as test beds for comparison. Three groups of benchmark functions with different characteristics are selected to benchmark the performance of the LMFO algorithm from different perspectives. As shown in Tables 1–3, these benchmark functions are divided into three groups: unimodal functions, multimodal functions, and fixed-dimension multimodal functions. As their names imply, unimodal functions are suitable for benchmarking the exploitation and convergence of an algorithm since they have one global optimum and no local optima. In contrary, multimodal functions have more than one optimum, which makes them more challenging than unimodal functions. One of the optima is called global optimum, and the rest are called local optima. An algorithm should avoid all the local optima to approach and approximate the global

Table 1: Unimodal benchmark functions.

Name	Function	Range	Dim	$f_{\min}$
Sphere	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$	200	0
Schwefel's 2.22	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$	200	0
Schwefel's 1.2	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]$	200	0
Schwefel's 2.21	$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	$[-100, 100]$	200	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$	$[-30, 30]$	200	0
Step	$f_6(x) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$	$[-100, 100]$	200	0
Quartic	$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}(0, 1)$	$[-1.28, 1.28]$	200	0
X.S.Yang-7	$f_8(x) = \sum_{i=1}^{n} \varepsilon_j \left\lvert x_i - \frac{1}{i} \right\rvert$, $\varepsilon_j \in \bigcup [0, 1]$	$[-5, 5]$	200	0

optimum. Therefore, exploration and local optima avoidance of algorithms can be benchmarked by multimodal functions. The mathematical formulation of the employed benchmark functions is presented in Tables 1, 2, and 3, respectively. In these three tables, Range represent the boundary of the function's search space, Dim denotes the dimension of the function, and $f_{\min}$ is the theoretical minimum of the function.

Heuristic algorithms are stochastic optimization techniques, and therefore they have to be run more than 10 times for generating meaningful statistical results. The best obtained solution in the last iteration is calculated as the metrics of performance. The same method is selected to generate and report the results over 30 independent runs. However, average and standard deviation only compare the overall performance of algorithms.

To explore the performance of the proposed LMFO algorithm, some of the recent and well-known algorithms in the literature are chosen: ABC [9], BA [10], GGSA [29], DA [17], PSOGSA [30], and MFO [18]. Note that 30 number search agents and 1000 iterations are utilized for each of the algorithms. It should be noted that selection of the number of moths (or other candidate solutions in other algorithms) should be done experimentally.

In this paper, Best, Mean, Worst, and Std represent the optimal fitness value, mean fitness value, worst fitness value, and standard deviation, respectively. Experimental results are listed in Tables 4, 5, and 6. The best results are denoted in *bold type*.

Due to the stochastic nature of the algorithms, statistical tests should be conducted to confirm the significance of the results [31]. The averages and standard deviation only compare the overall performance of the algorithms, while a statistical test considers each run's results and proves that the results are statistically significant. In order to determine whether the results of LMFO differ from the best results of ABC, BA, GGSA, DA, PSOGSA, and MFO in a statistical method, a nonparametric test which is known as Wilcoxon's rank-sum test [32, 33] is performed at 5% significance level. Tables 7, 8, and 9 report the p values produced by Wilcoxon's test for the pairwise comparison of the best value of six groups. Such groups are formed by ABC versus LMFO, BA versus LMFO, GGSA versus LMFO, DA versus MFO, PSOGSA versus LMFO, and MFO versus LMFO. In general, p values < 0.05 can be considered as sufficient evidence against the null hypothesis. With the statistical test, we can make sure that the results are not generated by chance. The nonparametric Wilcoxon statistical test is conducted and the calculated p values are reported as metrics of significance as well. Experimental results of p values rank-sum test are listed in Tables 7, 8, and 9.

4.3. Unimodal Benchmark Functions. The unimodal benchmarks functions have only one global minimum and there are no local minima for them. Therefore, these kinds of functions are very suitable for benchmarking the convergence capability of algorithms. According to the results of Table 4, LMFO is able to provide very competitive results. As can be seen from this table, LMFO outperforms all other algorithms in $f_1 \sim f_8$. Therefore, the proposed algorithm has high performance to find the global minimum of unimodal benchmark functions. According to the p values of $f_1 \sim f_8$ in Table 7, LMFO achieves significant improvement in all the unimodal benchmark functions compared to other algorithms. Hence, this proves that LMFO has better performance than other algorithms in forging for global optimum solution of unimodal benchmark functions.

Figures 1–8 illustrate the averaged convergence curves of all algorithms disposing unimodal benchmark functions over 30 independent runs. It can be noted here that all the convergence curves in the following subsections are

Table 2: Multimodal benchmark functions.

Name	Function	Range	Dim	$f_{\min}$
Rastrigin	$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]$	200	0
Ackley	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]$	200	0
Griewank	$f_{11} = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$	200	0
Penalized 1	$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_1 - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$; $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]$	200	0
Penalized 2	$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]$	200	0
Alpine	$f_{14}(x) = \sum_{i=1}^{n} x_i \sin(x_i) + 0.1x_i$	$[-10, 10]$	200	0
Zakharov	$f_{15}(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$	$[-5, 10]$	200	0

Table 3: Fixed-dimension multimodal benchmark functions.

Name	Function	Range	Dim	$f_{\min}$
Goldstein-Price	$f_{16}(x) = \left[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$[-2, 2]$	2	3
Drop Wave	$f_{17}(x) = -\dfrac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{(1/2)(x_1^2 + x_2^2) + 2}$	$[-5.12, 5.12]$	2	−1
Schaffer's F6	$f_{18}(x) = \dfrac{\sin^2\sqrt{(x_1^2 + x_2^2)} - 0.5}{\left[1 + 0.001(x_1^2 + x_2^2)\right]^2}$	$[-100, 100]$	2	−1
Easom	$f_{19} = -\cos(x_1) \times \cos(x_2) \times e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	[−2∗pi, 2∗pi]	2	−1

Table 4: Results of unimodal benchmark functions.

Benchmark function	Result	Algorithm						
		ABC	BA	GGSA	DA	PSOGSA	MFO	LMFO
f_1 ($D = 200$)	Best	19896.78	365717.3	16278.78	1513.435	34835.45	140219.4	**1.2921E − 234**
	Worst	36049.23	456284	24262.84	52511.62	142227.4	240000.7	**3.3309E − 183**
	Mean	26677.04	414807.8	19224.04	24716.82	94735.12	185588.4	**1.1103E − 184**
	Std	4640.383	20515.24	1852.355	11046.55	25051.92	24177.67	**0**
f_2 ($D = 200$)	Best	108.0735	1.83E + 84	149.7263	26.27986	637.56	420.5512	**8.5E − 129**
	Worst	186.4672	3.9E + 99	189.8577	255.79923	8.06E + 40	722.4955	**7.3E − 99**
	Mean	140.2811	1.45E + 98	165.8829	137.88594	2.69E + 39	560.1173	**2.4E − 100**
	Std	14.5252	7.06E + 98	9.599018	56.674292	1.47E + 40	61.66749	**1.3E − 99**
f_3 ($D = 200$)	Best	582404	823783.5	160277.3	69110.62	344077.3	462517.7	**1.6E − 215**
	Worst	920006.2	4617625	1020727	906356.8	891332.5	1094366	**6.4E − 167**
	Mean	757278.3	1572330	434895.9	342508.6	541491.3	772538.3	**3.3E − 168**
	Std	72837.78	837152.1	190865.2	203490.9	142131.3	170946.5	**0**
f_4 ($D = 200$)	Best	94.265	87.48997	20.44273	27.0486	76.25401	95.26014	**6.1E − 117**
	Worst	97.40102	92.12282	31.30115	59.69617	98.74303	98.15999	**6.18E − 82**
	Mean	95.8655	90.09865	27.12328	41.97914	95.69831	97.03653	**2.06E − 83**
	Std	0.778229	1.330154	2.551274	7.80909	5.90858	0.687621	**1.13E − 82**
f_5 ($D = 200$)	Best	16111660	88328339	2399953	4717141	2565682	3.61E + 08	**196.9121**
	Worst	94041761	1.59E + 08	5103632	47976603	4.82E + 08	8.92E + 08	**198.6685**
	Mean	41344878	1.33E + 08	3740872	18257995	1.35E + 08	6.14E + 08	**198.2609**
	Std	17567952	18028179	643192.1	10933433	1.24E + 08	1.41E + 08	**0.459128**
f_6 ($D = 200$)	Best	17165.43	384155.3	14899	4422.437	33665.58	136678.5	**39.65709**
	Worst	30743.79	453905.5	24873	63473.4	109123.6	228511.5	**42.92694**
	Mean	23922.99	417903	19220.47	24167.16	76561.32	180669.8	**41.49763**
	Std	3074.406	17001.78	2314.721	12441.1	19018.55	24671.19	**0.632867**
f_7 ($D = 200$)	Best	51.88092	0.400252	6.492429	9.23801	11.89278	1181.991	**2.3E − 06**
	Worst	299.2207	0.744656	14.77883	181.7351	71.66514	2686.255	**0.000427**
	Mean	157.1676	0.551746	9.090199	52.06488	19.83666	1908.788	**8.56E − 05**
	Std	58.73575	0.072809	1.742479	39.34587	10.47516	360.7708	**0.000108**
f_8 ($D = 200$)	Best	181.8735	172.1645	28.45978	14.59629	130.6204	168.6843	**1.62807**
	Worst	198.3101	200.1905	40.21054	80.16251	169.9275	202.2427	**1.840634**
	Mean	190.2888	188.036	33.67195	43.38706	153.398	187.9161	**1.752264**
	Std	4.168661	6.867959	2.232835	13.6498	10.91374	7.601635	**0.06096**

also averaged curves. As may be seen from these curves, LMFO has the fastest convergence speed in all algorithms. From Table 4 and Figures 20–27, the LMFO's Std is much smaller than other algorithms. These show that LMFO has a strong sense of stability and robust comparing from other algorithms.

4.4. Multimodal Benchmark Functions. In contrast to the unimodal benchmark functions, multimodal benchmark functions have many local minima with the number increasing exponentially with dimension. This makes them suitable for benchmarking the exploration ability of an algorithm. So, the final results are more important because these benchmark functions can reflect the ability of the algorithm to escape from poor local optima and obtain the global optimum. The statistical results of the algorithms on multimodal benchmark functions are presented in Table 5. As the results of Best, Worst, Mean, and Std values show, LMFO is also able to provide very competitive results on the multimodal benchmark functions. These results show that the LMFO algorithm

Table 5: Results of multimodal benchmark functions.

Benchmark function	Result	Algorithm						
		ABC	BA	GGSA	DA	PSOGSA	MFO	LMFO
f_9 (D = 200)	Best	575.1608	1363.369	1563.422	730.92416	922.1832	1769.441	**0**
	Worst	709.019	1810.465	1867.616	1890.9545	1476.272	2125.69	**0**
	Mean	653.9316	1630.768	1752.838	1367.3899	1231.578	1951.426	**0**
	Std	36.46425	100.7203	81.02372	280.04565	117.1903	79.01827	**0**
f_{10} (D = 200)	Best	12.54258	19.20527	10.52907	8.666612	19.23309	19.92131	**8.88E − 16**
	Worst	14.8045	19.9564	11.89042	14.75436	19.96677	20.01897	**8.88E − 16**
	Mean	13.82113	19.73614	11.21024	11.88909	19.69719	19.95433	**8.88E − 16**
	Std	0.554162	0.260154	0.383654	1.453704	0.31107	0.019396	**0**
f_{11} (D = 200)	Best	142.2548	4290.404	137.6285	61.43179	654.9388	1229.64	**0**
	Worst	316.036	5283.435	221.9276	600.151	1631.547	2048.454	**0**
	Mean	225.3103	4997.492	171.5404	224.0744	1251.695	1543.04	**0**
	Std	51.75854	184.8135	16.55978	104.4042	204.6207	197.6001	**0**
f_{12} (D = 200)	Best	7393398	3.86E + 08	28.17774	1662.384	7043398	7.03E + 08	**8.88E − 16**
	Worst	1.67E + 08	6.85E + 08	77685.77	23172033	2.05E + 09	1.92E + 09	**8.88E − 16**
	Mean	58731093	5.59E + 08	11043.76	2516012	7.28E + 08	1.3E + 09	**8.88E − 16**
	Std	37177591	81914079	20355.03	4641204	5.15E + 08	3.01E + 08	**0**
f_{13} (D = 200)	Best	19464968	1.25E + 09	646700.5	2620391	26666517	1.68E + 09	**19.4713**
	Worst	3.4E + 08	2.09E + 09	3319003	1.83E + 08	2.47E + 09	4.04E + 09	**19.79025**
	Mean	1.53E + 08	1.72E + 09	1684844	27183053	9.73E + 08	2.62E + 09	**19.62771**
	Std	82296565	1.89E + 08	724732.9	34759974	7.07E + 08	5.46E + 08	**0.072549**
f_{14} (D = 200)	Best	47.28706	90.66079	103.3283	15.38694	58.10706	129.0595	**1.1E − 125**
	Worst	62.0743	168.7346	139.0044	197.7987	107.5327	216.9578	**1.1E − 103**
	Mean	53.55967	119.4603	118.7804	116.3619	80.96392	171.3074	**3.9E − 105**
	Std	3.438082	19.18661	8.52589	43.5411	12.93695	23.18654	**2E − 104**
f_{15} (D = 200)	Best	5373.354	4315.808	461.6698	1142.924	4411.244	5741.417	**5E − 168**
	Worst	6101.204	10167.76	3585.345	4914.844	9429.568	11241.25	**2.6E − 117**
	Mean	5810.789	5283	1327.234	3451.42	6873.222	8566.525	**8.6E − 119**
	Std	207.7704	1065.437	640.8766	1139.356	1451.294	1527.992	**4.7E − 118**

Table 6: Results of fixed-dimension multimodal benchmark functions.

Benchmark function	Result	Algorithm						
		ABC	BA	GGSA	DA	PSOGSA	MFO	LMFO
f_{16} (D = 2)	Best	3.000547	3	3	3	3	**3**	3
	Worst	3.052848	84.00001	30	3.179715	84	**3**	3.000357
	Mean	3.015989	15.6	7.144215	3.018584	8.4	**3**	3.000061
	Std	0.015909	25.30177	9.250619	0.049376	20.55036	**1.83E − 15**	7.19E − 05
f_{17} (D = 2)	Best	−1	−1	−1	−1	−1	−1	**−1**
	Worst	−0.98844	−0.36913	−0.93625	−0.78575	−0.93625	−0.93625	**−1**
	Mean	−0.99731	−0.71344	−0.949	−0.94932	−0.9915	−0.96175	**−1**
	Std	0.003196	0.175563	0.025938	0.053641	0.022043	0.031767	**0**
f_{18} (D = 2)	Best	−0.99909	−0.99028	−1	−1	−1	−1	**−1**
	Worst	−0.98981	−0.54822	−0.99028	−0.92181	−0.99028	−0.99028	**−1**
	Mean	−0.99095	−0.71504	−0.99286	−0.98153	−0.99093	−0.99061	**−1**
	Std	0.002023	0.126856	0.004349	0.02047	0.002465	0.001774	**0**
f_{19} (D = 2)	Best	−1	−1	−1	−1	**−1**	**−1**	−1
	Worst	−0.99991	−8.1E − 05	−8.1E − 05	−0.95227	**−1**	**−1**	−0.99918
	Mean	−0.99999	−0.80002	−0.92221	−0.99656	**−1**	**−1**	−0.99973
	Std	2.03E − 05	0.406805	0.257944	0.010494	**0**	**0**	0.000229

Table 7: Results of p-values rank-sum test on unimodal benchmark functions.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
ABC versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
BA versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
GGSA versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
DA versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
PSOGSA versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
MFO versus LMFO	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$

Table 8: Results of p-values rank-sum test on multimodal benchmark functions.

	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
ABC versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
BA versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
GGSA versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
DA versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
PSOGSA versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$
MFO versus LMFO	$1.21E-12$	$1.21E-12$	$1.21E-12$	$3.02E-11$	$3.02E-11$	$3.02E-11$	$3.02E-11$

Table 9: Results of p-values rank-sum test on fixed-dimension multimodal benchmark functions.

	f_{16}	f_{17}	f_{18}	f_{19}
ABC versus LMFO	$3.02E-11$	$1.21E-12$	$1.21E-12$	$5.07E-10$
BA versus LMFO	$7.29E-03$	$1.21E-12$	$1.21E-12$	$6.77E-05$
GGSA versus LMFO	$7.70E-02$	$8.81E-10$	$5.13E-11$	$2.48E-08$
DA versus LMFO	$5.19E-02$	$4.56E-10$	$1.45E-11$	$2.96E-06$
PSOGSA versus LMFO	$7.91E-09$	$4.18E-02$	$7.15E-13$	$1.21E-12$
MFO versus LMFO	$2.56E-11$	$6.89E-07$	$1.17E-13$	$1.21E-12$

has merit in terms of exploration. According to the p values of $f_9 \sim f_{15}$ reported in Table 8, LMFO achieves significant improvement on 200-D compared to other algorithms. When comparing LMFO and other algorithms, we can conclude that LMFO is significantly performing better with six groups of comparison algorithms. The p values of $f_9 \sim f_{15}$ reported in Table 8 are less than 0.05, which is strong evidence against null hypothesis. Hence, this evidence demonstrates that the results of LMFO are statistically significant not occurring by coincidence.

Seen from Table 5 and Figures 9–15, the convergence rate of LMFO on the multimodal benchmark functions in majority cases is better than other algorithms. On the basis of Table 5, and Figures 9–15, we can draw a conclusion that the LMFO is able to avoid local minima in multimodal benchmark functions with a good convergence speed. From Table 5 and Figures 28–37, the LMFO's Std is much smaller than other algorithms. These show that LMFO has a strong sense of stability and robust comparing from other algorithms.

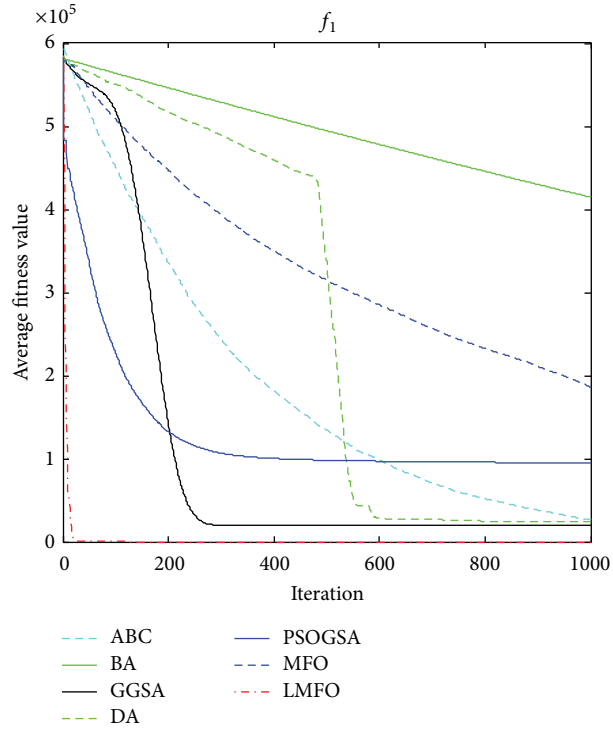

Figure 1: The convergence curves for f_1.

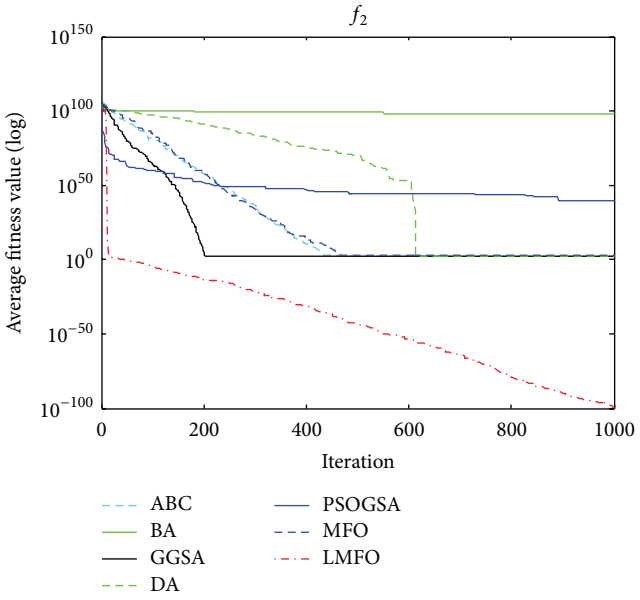

Figure 2: The convergence curves for f_2.

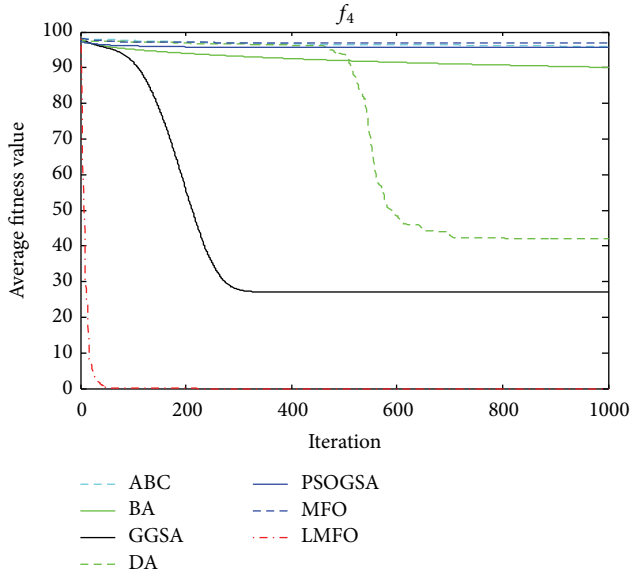

Figure 4: The convergence curves for f_4.

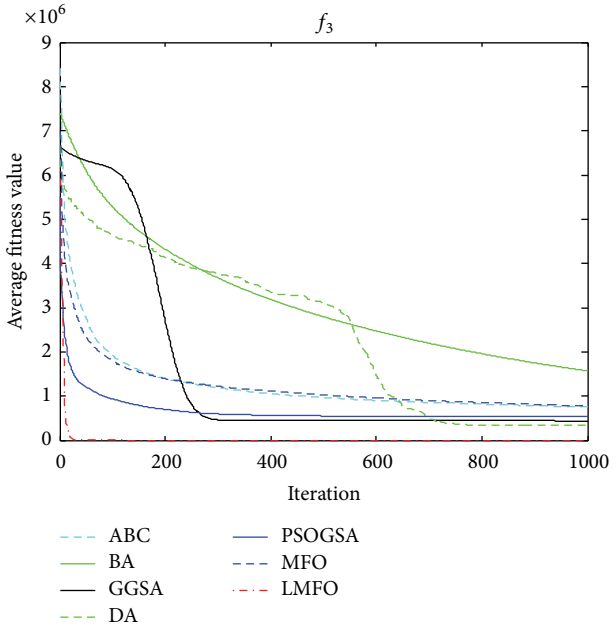

Figure 3: The convergence curves for f_3.

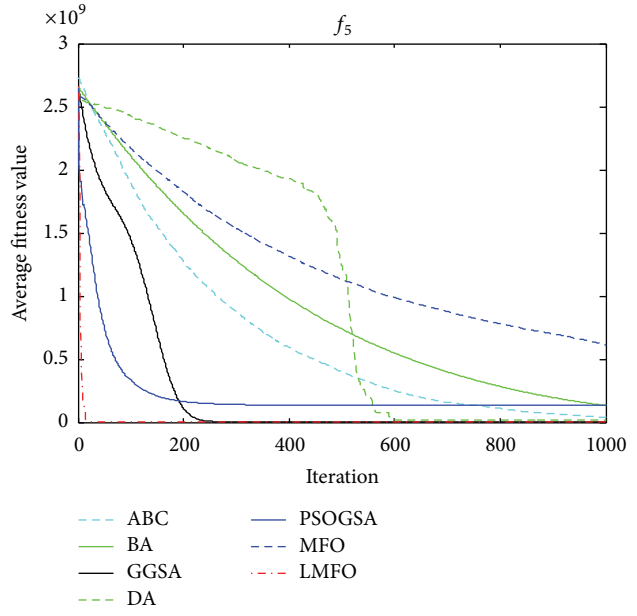

Figure 5: The convergence curves for f_5.

4.5. Fixed-Dimension Multimodal Benchmark Functions. For fixed-dimension multimodal benchmark functions with only a few local minima, the dimensions of the multimodal benchmark functions are also small. Under such circumstances, it is difficult to judge the performance of individual algorithm. The major difference compared with multimodal functions is that fixed-dimension multimodal functions appear to be simpler because of their low dimensions and a smaller number of local minima. In this experiment, the results of Best, Worst, Mean, and Std values of fixed-dimension multimodal benchmark functions are summarized in Table 6. For all fixed-dimension multimodal functions, LMFO can give the best solution in terms of Best. As Table 6 shows, the LMFO algorithm provides the best results on two of fixed-dimension multimodal benchmark functions. The results are followed by the MFO, PSOGSA, and ABC algorithms. In addition, the p values of Wilcoxon's rank-sum in Table 9 show that the result of LMFO in f_{16} is not significantly better than DA and GGSA algorithms (5% significance level), but it is significantly different compared with ABC, MFO, PSOGSA, and BA. In the remaining functions (f_{17}, f_{18}, and

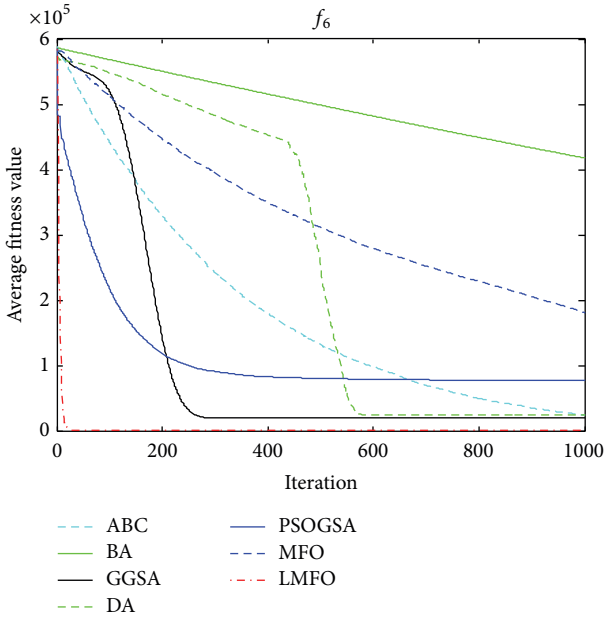

Figure 6: The convergence curves for f_6.

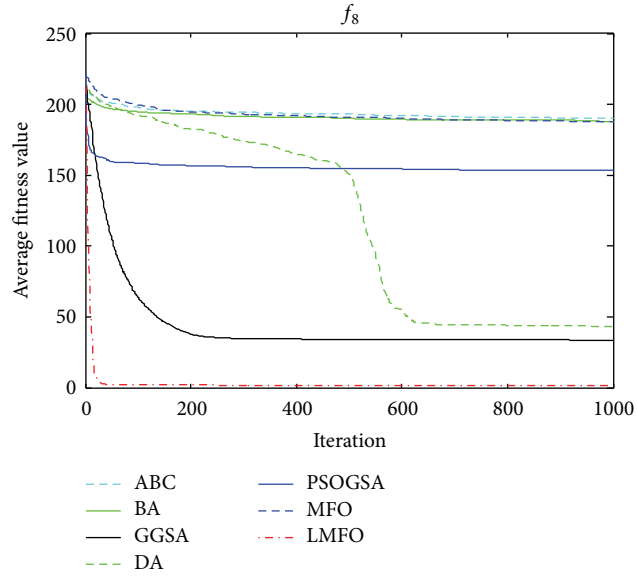

Figure 8: The convergence curves for f_8.

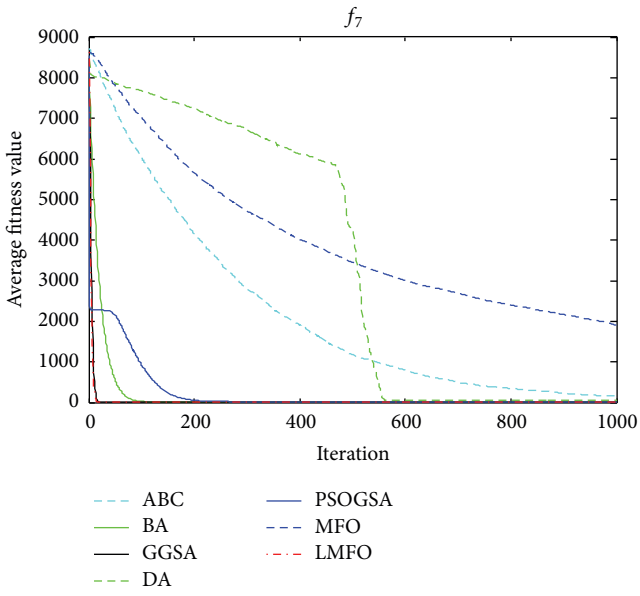

Figure 7: The convergence curves for f_7.

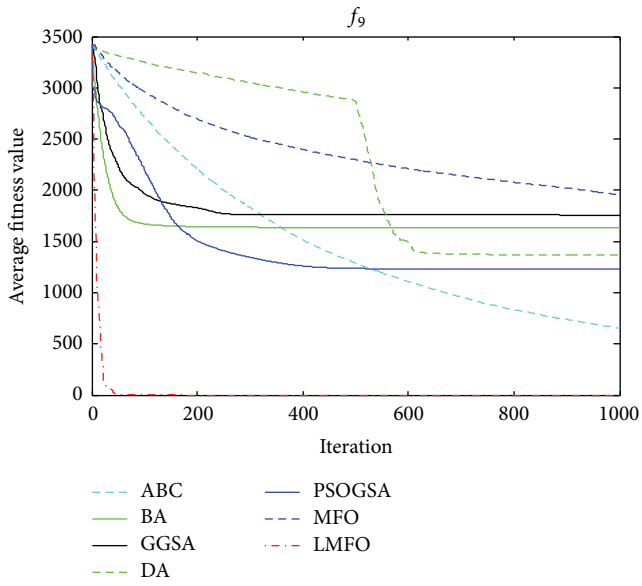

Figure 9: The convergence curves for f_9.

f_{19}), however, the results of LMFO are significantly better than other algorithms. So, it can be concluded that the results of LMFO in these benchmark functions are better than ABC, BA, GGSA, DA, PSOGSA, and MFO.

In addition, the convergence rate of LMFO on the fixed-dimension benchmark functions with 2-dim can be shown in Figures 16–19. As can be seen from these figures, it can be claimed that LMFO has the faster convergence rate on functions f_{17} and f_{18}. From Figures 35–38, we can find that all of the algorithms have a strong sense of stability except BA on fixed-dimension functions.

Overall, the results from Tables 4–6, Tables 7–9, Figures 1–19, and Figures 20–38 show that the proposed method is effective in not only optimizing unimodal and multimodal functions but also optimizing fixed-dimension multimodal functions.

Since constraints are one of the major challenges in solving real problems and the main objective of designing the LMFO algorithm is to solve real problems, two constrained real engineering problems are employed in the next section

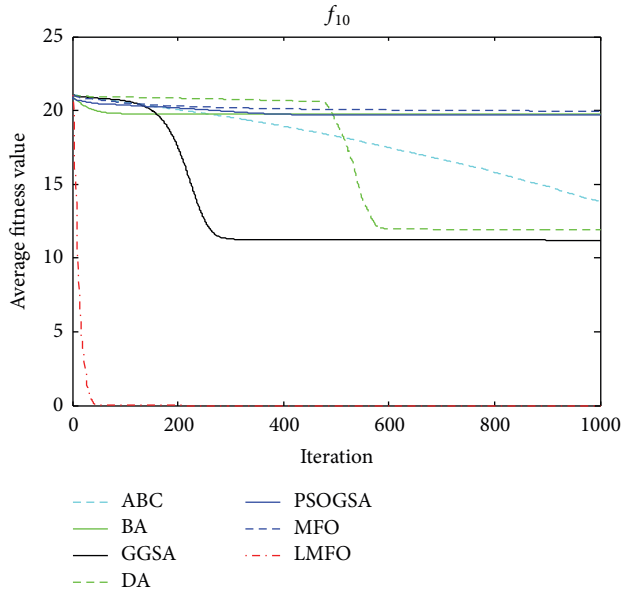

Figure 10: The convergence curves for f_{10}.

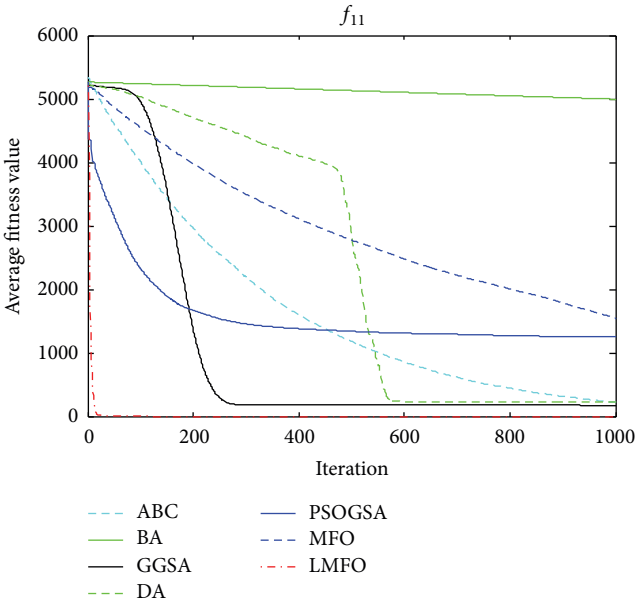

Figure 11: The convergence curves for f_{11}.

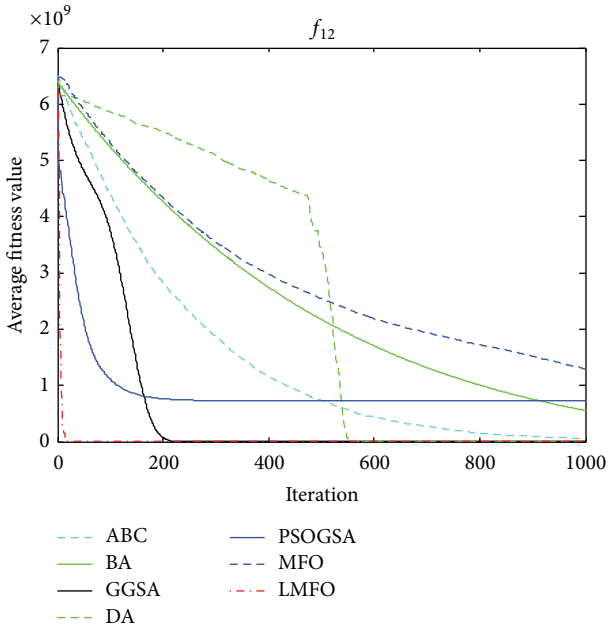

Figure 12: The convergence curves for f_{12}.

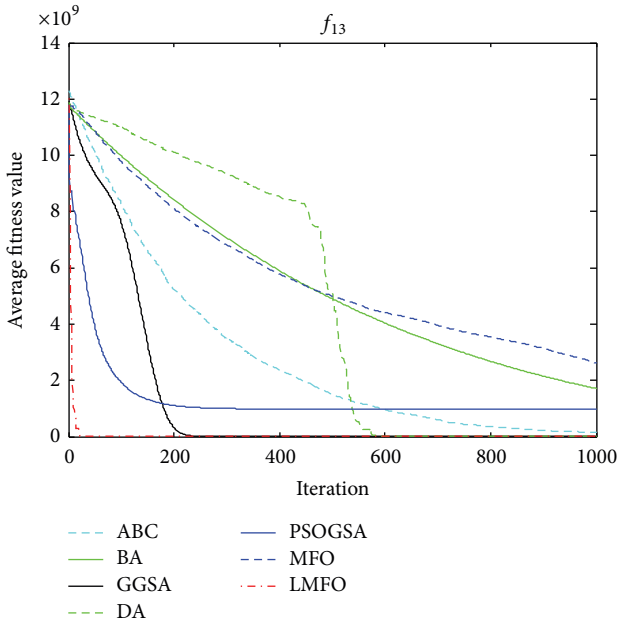

Figure 13: The convergence curves for f_{13}.

to further investigate the performance of the MFO algorithm and provide a comprehensive study.

5. LMFO for Engineering Optimization Problems

In this section, a set of two engineering problems (welded beam design and speed reducer design) is solved so as to further testify the performance of the proposed algorithm. There are some inequality constraints in real problems, so the LMFO algorithm should be capable of dealing with them during optimization. Several methods have been applied to handle constraints in the literature: penalty function, special operators, repaired algorithms, separation of objectives and constraints, and hybrid methods [34]. In this paper, penalty method is employed to handle the constraints of welded beam and speed reducer.

5.1. Welded Beam Design. The objective is to evaluate the optimal fabrication cost of a welded beam as shown in Figure 39 [35]. The constraints of the beam are shear stress

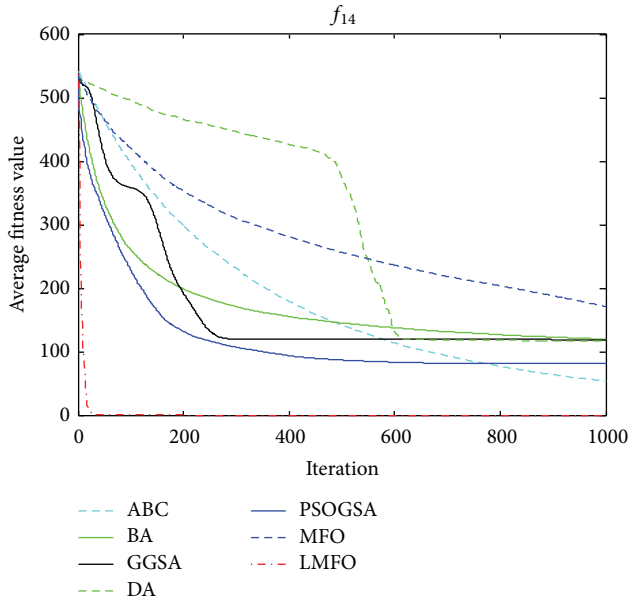

Figure 14: The convergence curves for f_{14}.

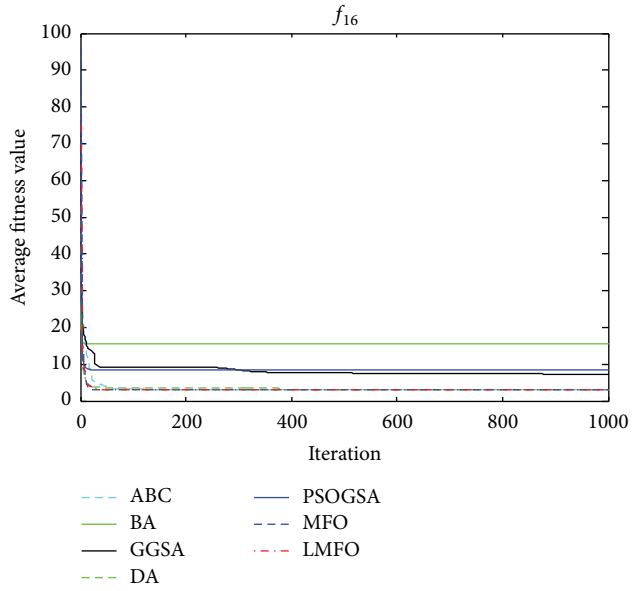

Figure 16: The convergence curves for f_{16}.

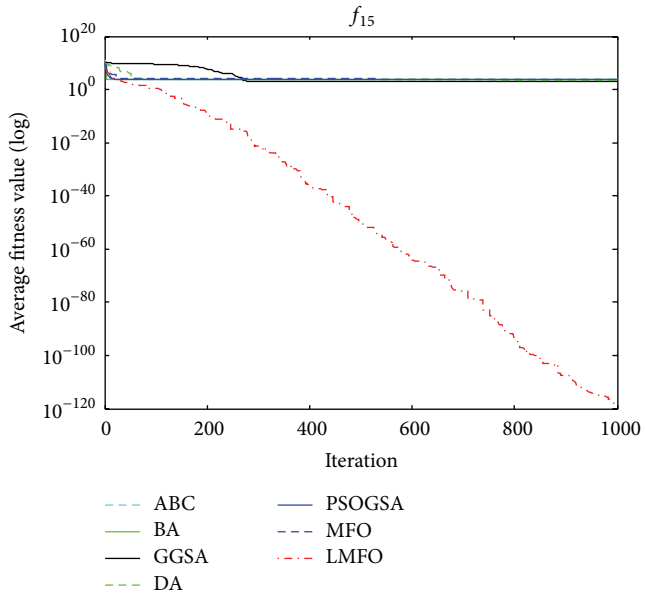

Figure 15: The convergence curves for f_{15}.

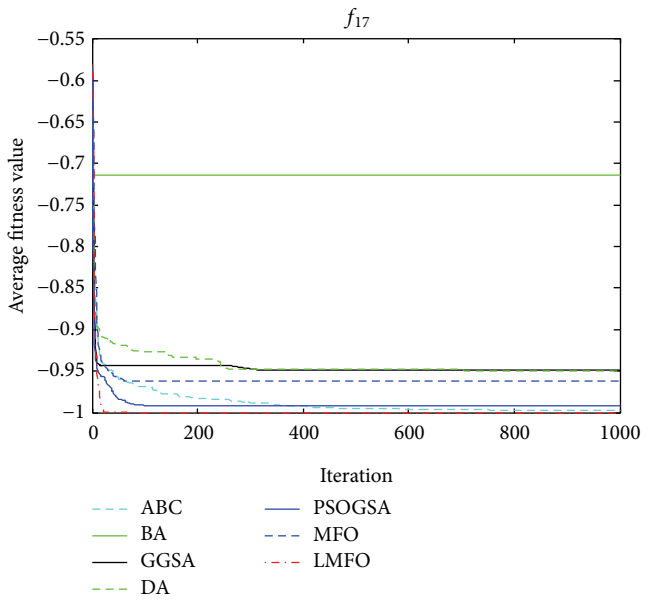

Figure 17: The convergence curves for f_{17}.

(τ), bending stress in the beam (θ), buckling load on the bar (P_c), end deflection of the beam (δ), and side constraints.

This problem has four variables that are thickness of weld (h), length of attached part of bar (l), the height of the bar (t), and thickness of the bar (b), respectively. This problem is formulated as follows:

$$\text{Consider} \quad \vec{x} = [x_1, x_2, x_3, x_4] = [h, l, t, b],$$

$$\text{Minimize} \quad f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2),$$

$$\text{Subject to} \quad g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0,$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0,$$

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0,$$

$$g_4(\vec{x}) = x_1 - x_4 \le 0,$$

$$g_5(\vec{x}) = P - P_c(\vec{x}) \le 0,$$

$$g_6(\vec{x}) = 0.125 - x_1 \le 0,$$

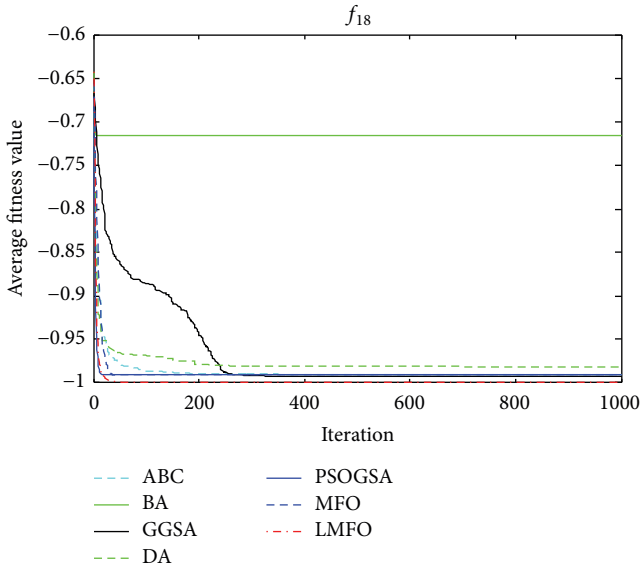

Figure 18: The convergence curves for f_{18}.

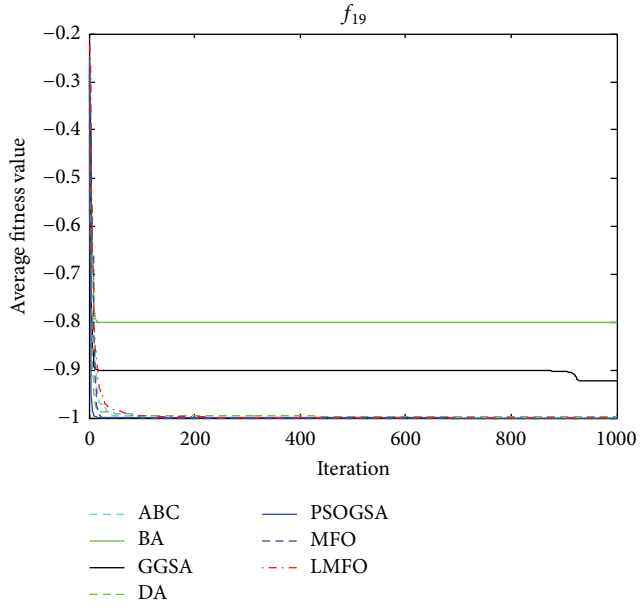

Figure 19: The convergence curves for f_{19}.

$$g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,$$

Variable range $0.1 \le x_1 \le 2$,

$0.1 \le x_2 \le 10$,

$0.1 \le x_3 \le 10$,

$0.1 \le x_4 \le 2$,

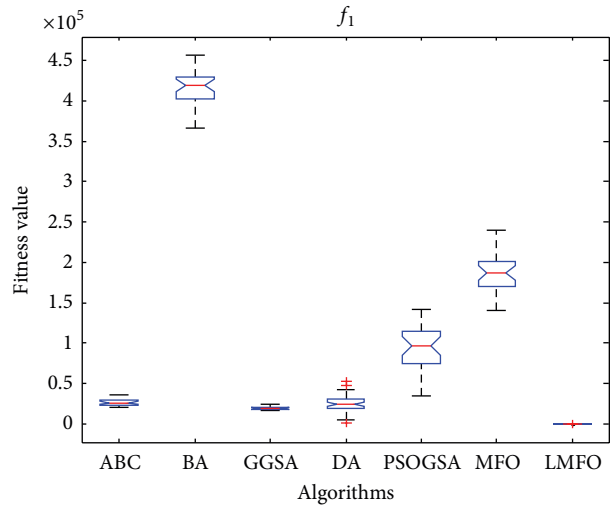

Figure 20: Standard deviation for f_1.

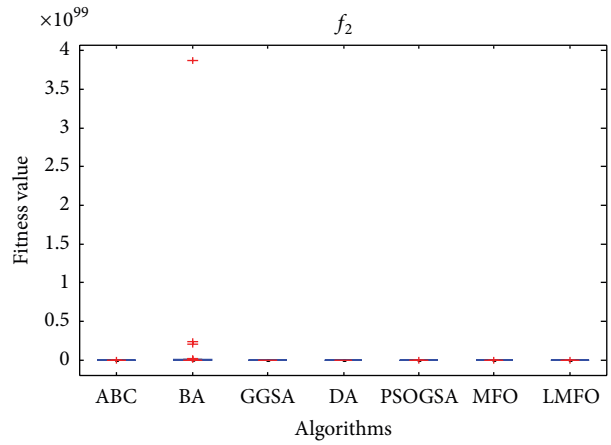

Figure 21: Standard deviation for f_2.

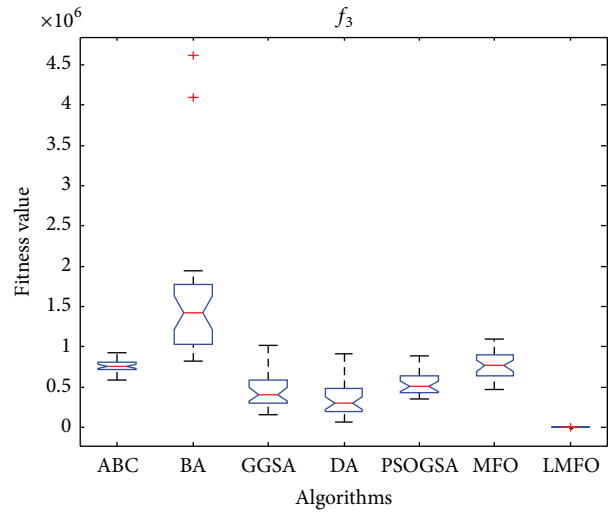

Figure 22: Standard deviation for f_3.

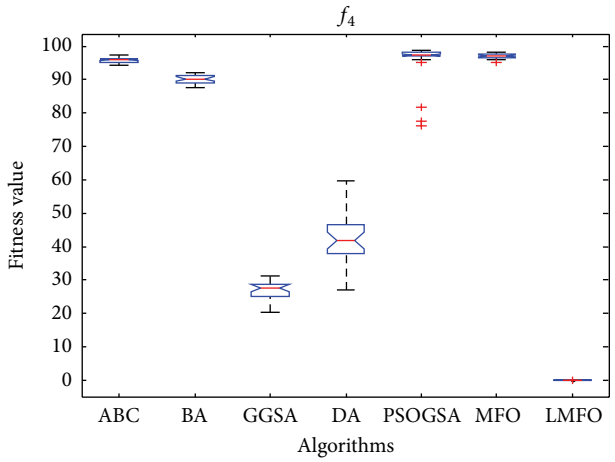

Figure 23: Standard deviation for f_4.

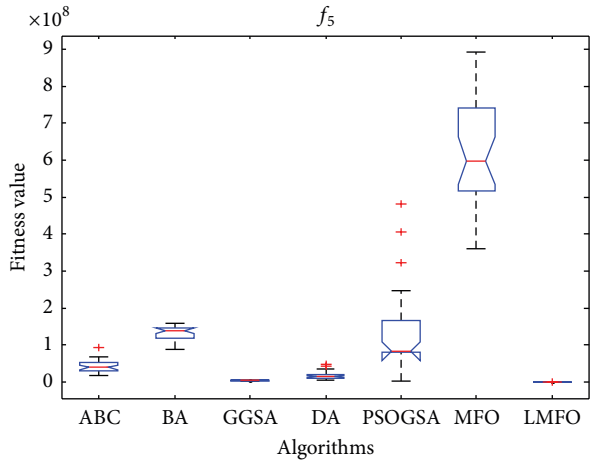

Figure 24: Standard deviation for f_5.

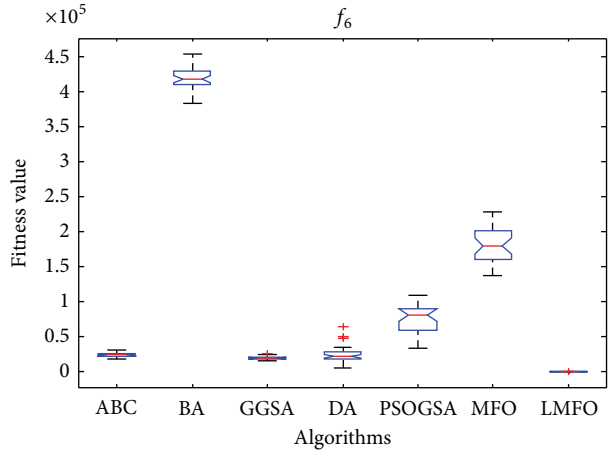

Figure 25: Standard deviation for f_6.

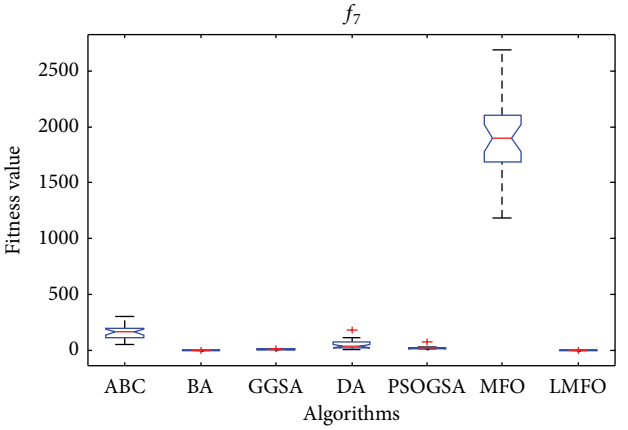

Figure 26: Standard deviation for f_7.

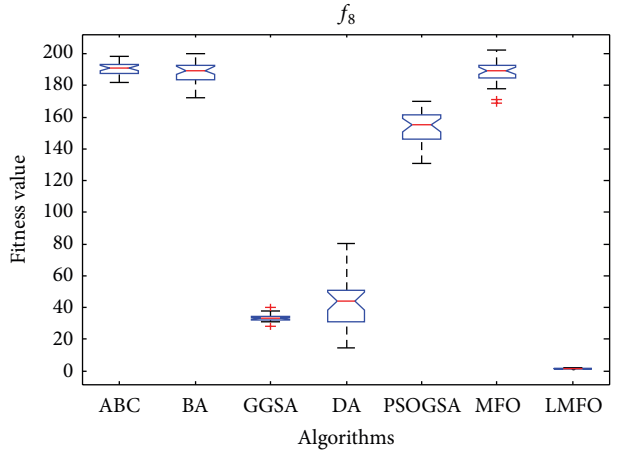

Figure 27: Standard deviation for f_8.

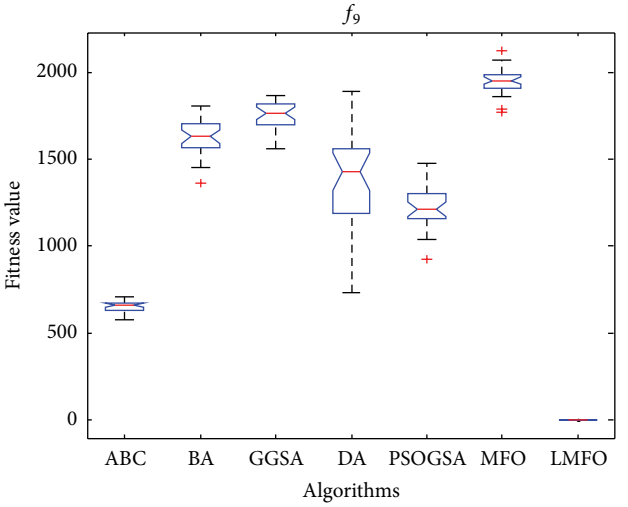

Figure 28: Standard deviation for f_9.

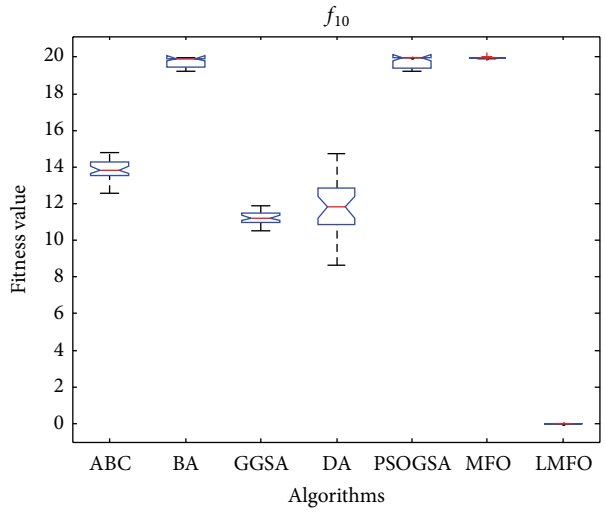

Figure 29: Standard deviation for f_{10}.

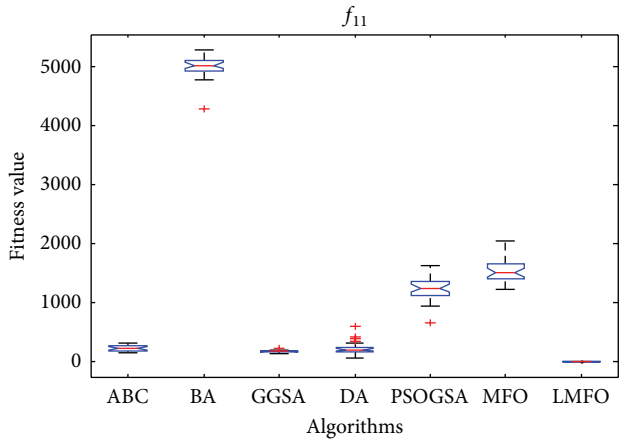

Figure 30: Standard deviation for f_{11}.

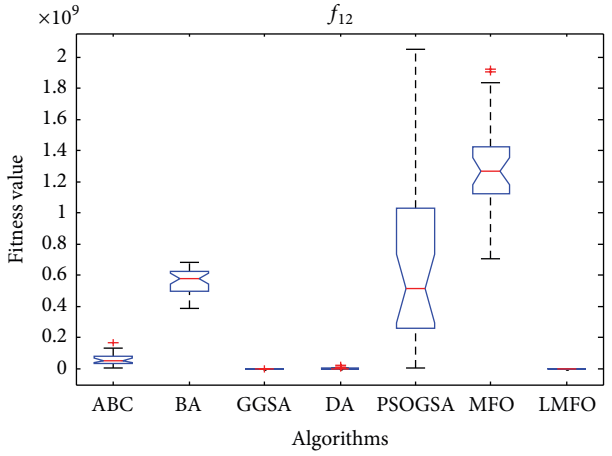

Figure 31: Standard deviation for f_{12}.

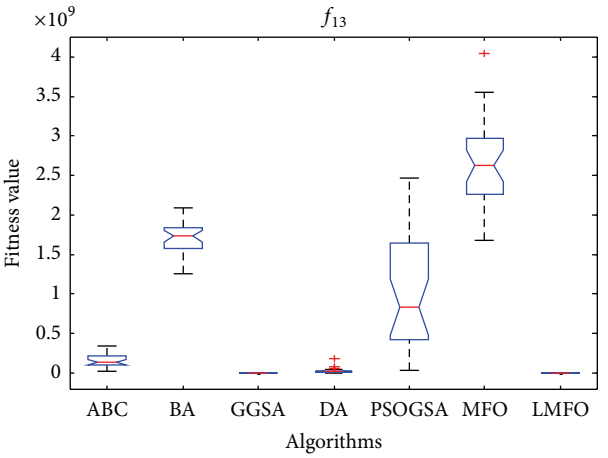

Figure 32: Standard deviation for f_{13}.

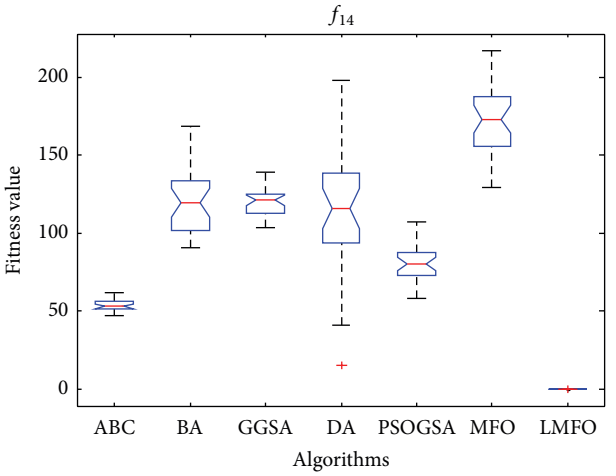

Figure 33: Standard deviation for f_{14}.

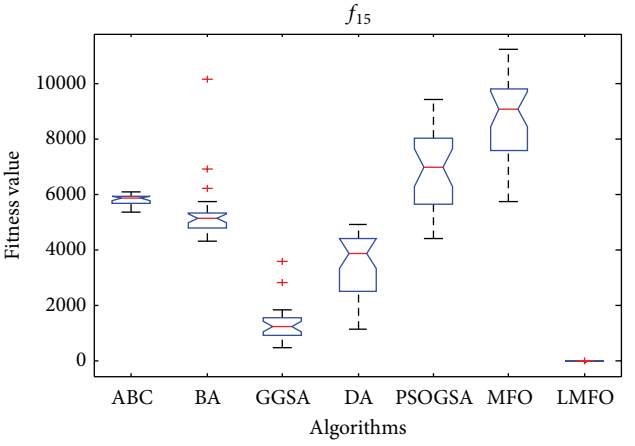

Figure 34: Standard deviation for f_{15}.

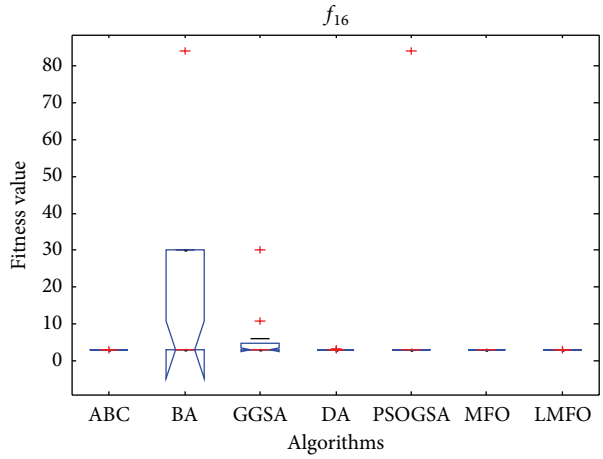

Figure 35: Standard deviation for f_{16}.

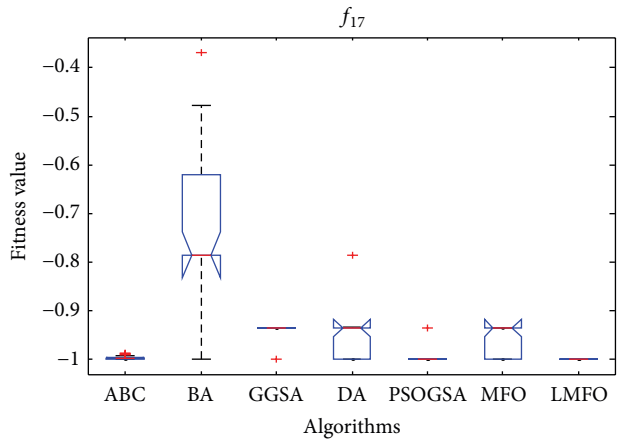

Figure 36: Standard deviation for f_{17}.

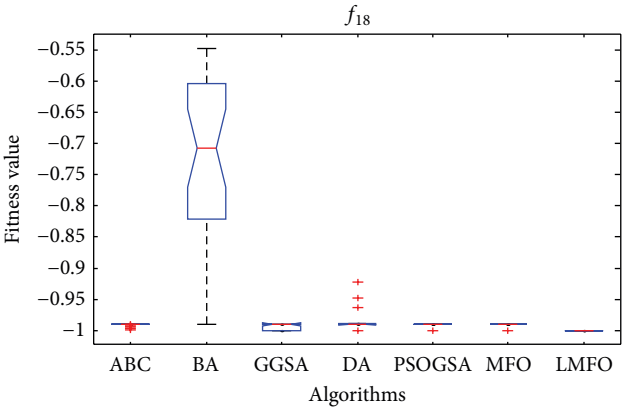

Figure 37: Standard deviation for f_{18}.

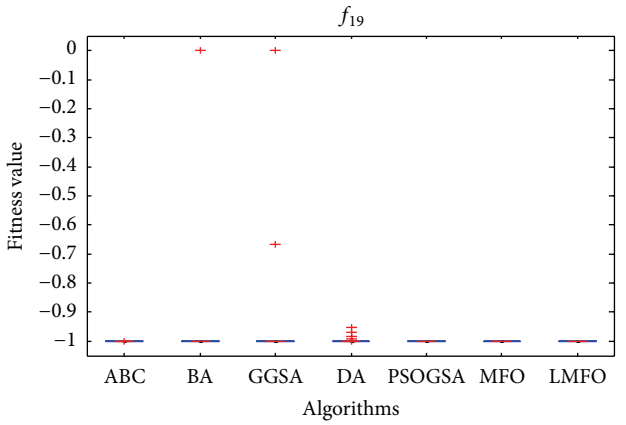

Figure 38: Standard deviation for f_{19}.

Where

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

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

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

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

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

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2},$$

$$\delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$$

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

$$P = 6000lb,$$

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

$$\delta_{\max} = 0.25\,\text{in.},$$

$$E = 30 \times 1^6\,\text{psi},$$

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

$$\tau_{\max} = 13600\,\text{psi},$$

$$\sigma_{\max} = 30000\,\text{psi}. \tag{13}$$

Mirjalili tried to solve this problem using MFO [18] and GGSA [29, 36]. Coello Coello [37] and Deb [38, 39] employed GA, whereas Lee and Geem [40] used HS to solve this problem. Richardson's random method, simplex method,

Table 10: Comparison results of the welded beam design problem.

Algorithm	Optimum variables				Optimal cost
	H	l	t	B	
LMFO	*0.2020*	*3.3575*	*9.0938*	*0.2061*	*1.7165*
MFO [18]	0.2057	3.4703	9.0364	0.2057	1.72452
GGSA [29, 36]	0.215917	3.314955	8.896195	0.215917	1.770829
GA (Coello Coello) [37]	N/A	N/A	N/A	N/A	1.8245
GA (Deb) [38]	N/A	N/A	N/A	N/A	2.3800
GA (Deb) [39]	0.2489	6.1730	8.1789	0.2533	2.4331
HS (Lee and Geem) [40]	0.2442	6.2231	8.2915	0.2443	2.3807
Random [41]	0.4575	4.7313	5.0853	0.6600	4.1185
Simplex [41]	0.2792	5.6256	7.7512	0.2796	2.5307
David [41]	0.2434	6.2552	8.2915	0.2444	2.3841
APPROX [41]	0.2444	6.2189	8.2915	0.2444	2.3815

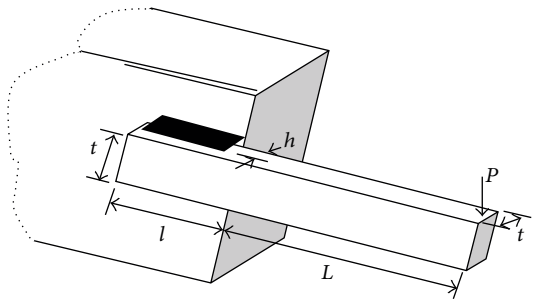

Figure 39: Structure of welded beam design.

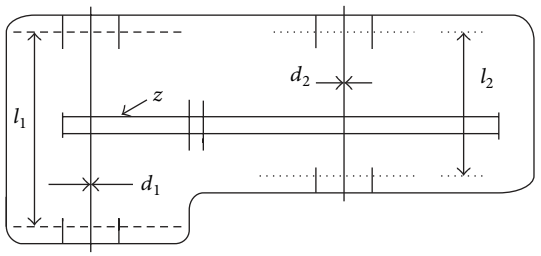

Figure 40: Structure of speed reducer design.

Davidon-Fletcher-Powell, and Griffith and Stewart's successive linear approximation are the mathematical approaches that have been adopted by Ragsdell and Philips [41] for this problem. The comparison results of the welded beam design problem are shown in Table 10.

The results of Table 10 show that the LMFO algorithm is able to find the best optimal design compared to other algorithms. The results of LMFO are closely followed by the MFO and GGSA algorithms.

5.2. Speed Reducer Design. The objective function of this problem is to minimize the total weight of the speed reducer as illustrated in Figure 40 [42]. The variables x_1~x_7 denote the face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), the diameter of first (d_1), and second shafts (d_2), respectively. The mathematical formulation of this problem can be summarized as follows:

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) \\
&= 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) \\
&\quad - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) \\
&\quad + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \\
\text{Subject to}\quad & g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0, \\
& g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0, \\
& g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0, \\
& g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_7^5x_3} - 1 \le 0, \\
& g_5(\vec{x}) = \frac{\left[\left(745x_4/x_2x_3\right)^2 + 16.9 \times 10^6\right]^{1/2}}{110x_6^3} - 1 \le 0, \\
& g_6(\vec{x}) = \frac{\left[\left(745x_5/x_2x_3\right)^2 + 157.5 \times 10^6\right]^{1/2}}{85x_7^3} - 1 \\
& \le 0, \\
& g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0, \\
& g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0, \\
& g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0, \\
& g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0, \\
& g_{11}(\vec{x}) = \frac{1.1x_6 + 1.7}{x_5} - 1 \le 0, \\
\text{where}\quad & 2.6 \le x_1 \le 3.6,
\end{aligned}$$

Table 11: Comparison results of the speed reducer design problem.

Algorithm	Optimal values for variables							Optimum weight
	$b(x_1)$	$m(x_2)$	$z(x_3)$	$l_1(x_4)$	$l_2(x_5)$	$d_1(x_6)$	$d_2(x_7)$	
LMFO	*3.50411*	*0.7*	*17*	*7.3*	*7.33342*	*3.37164*	*5.28916*	*2994.656*
Akhtar et al. [43]	3.506122	0.700006	17	7.549126	7.85933	3.365576	5.289773	3008.08
Mezura-Montes et al. [44]	3.506163	0.700831	17	7.460181	7.962143	3.3629	5.3090	3025.005
CS [11, 45]	3.5015	0.7	17	7.6050	7.8181	3.3520	5.2875	3000.981
HCPS [46]	3.5	0.7	17	7.3	7.71532	3.350215	5.286654	2994.47107
SCA [47]	3.5	0.7	17	7.327602	7.715321	3.350267	5.286655	2994.744241
$(\mu + \lambda)$ ES [5, 48]	3.499999	0.699999	17	7.3	7.8	3.350215	5.286683	2996.348094
ABC [9, 49]	3.499999	0.7	17	7.3	7.8	3.350215	5.287800	2997.058412

$$
\begin{aligned}
&0.7 \le x_2 \le 0.8,\\
&17 \le x_3 \le 28,\\
&7.3 \le x_4 \le 8.3,\\
&7.3 \le x_5 \le 8.3,\\
&2.9 \le x_6 \le 3.9,\\
&5.0 \le x_7 \le 5.5.
\end{aligned} \tag{14}
$$

This problem has also been popular among researchers and optimized in many studies. The heuristic algorithms that have been employed to optimize this problem are Akhtar et al. [43], Mezura-Montes et al. [44], CS [11, 45], HCPS [46], SCA [47], $(\mu + \lambda)$ ES [5, 48], and ABC [9, 49]. The results of this problem are provided in Table 11. According to this table, the LMFO and HCPS algorithms can find a design with the minimum weight for this problem.

6. Results and Discussion

In this paper, an improved version of MFO algorithm based on Lévy-flight strategy, which is named as LMFO, is proposed. In order to benchmark the performance of LMFO, nineteen unconstrained benchmark functions and two constrained engineering design problems were conducted.

According to the values of Best, Worst, Mean, and Std and p values in Section 4, the LMFO algorithm significantly outperforms others in terms of numerical optimization. There are several reasons of why LMFO algorithm did perform well on most of the test cases. First, Lévy-flight strategy: Lévy-flight can increase the diversity of the population and make the algorithm jump out of local optimum more effectively. This approach is helpful to make LMFO faster and more robust than MFO. Second, update mechanism of moths: in this mechanism, moths are required to update their positions with respect to the best recent feasible flames. Therefore, this approach promotes exploration of promising feasible regions and is the main reason of the superiority of the LMFO algorithm. Third, Quicksort method is utilized in LMFO algorithm. These are the reasons why LMFO performs better than other algorithms at the end of the results section. Another finding in the results is the poor performance of ABC, BA, and DA. These three algorithms belong to the class of swarm-based algorithms. In contrary to evolutionary algorithms, there is no mechanism for significant abrupt movements in the search space and this is likely to be the reason for the poor performance of ABC, BA, and DA.

As we can see in Section 4, the LMFO has been demonstrated to perform better than or highly competitive with the other algorithms. The advantages of LMFO involve performing simply and have few parameters to regulate. The work here proves the LMFO to be robust, powerful, and effective over all types of benchmark functions. Benchmark evaluating is a good way for testing the performance of the metaheuristic algorithms, but it also has some limitations. For example, different tuning parameter values in the optimization methods might lead to significant differences in their performance. Also, benchmark test may arrive at fully different conclusions if the termination criterion changes. If we change the population size or the number of iterations, we might draw a different conclusion.

In Section 5, the results show that MFO outperforms other algorithms in the majority of real case studies. Since the search space of these problems is unknown, these results are strong evidences for the applicability of LMFO in solving real problems. Due to the constrained nature of the case studies, in addition, it can be stated that the LMFO algorithm is able to optimize search spaces with infeasible regions as well. This is due to the update mechanism of moths, in which they are required to update their positions with respect to the best recent feasible flames. Therefore, this approach is the main reason of the superiority of the LMFO algorithm.

In our study, nineteen benchmark functions have been applied to evaluate the performance of LMFO. We also test our proposed method on the real-world engineering problems. Moreover, we will compare LMFO with other optimization algorithms.

7. Conclusion and Future Works

Due to the limited performance of MFO, Lévy-flight strategy has been introduced into the standard MFO to develop a novel Lévy-flight moth-flame optimization algorithm for optimization problems. As shown in Section 4, LMFO is very efficient with an almost exponential convergence rate and the results were compared to a wide range of algorithms for verification. This observation is based on the comparison of

LMFO with other algorithms to solve optimization problems. The proposed algorithm proved its superior performance on nineteen benchmark functions in terms of enhanced convergence speed and modified avoidance of local minima. This paper also identified and discussed the reasons for poor performances of other algorithms. It was observed that the swarm-based algorithms suffer from low exploration, whereas the LMFO does not.

Furthermore, this paper also considers solving two classical engineering problems by using the LMFO algorithm. The high level of exploration and exploitation of this algorithm were the motivations for this study. The comparative results in Section 5 show that the LMFO algorithm has high performance on challenging constrained problems with unknown spaces. In this work, the LMFO makes an attempt at taking merits of the MFO and Lévy-flight in order to avoid local optimal. With both techniques combined, LMFO can balance exploration and exploitation and effectively solve complex problems and real-world engineering problems.

For future works, two research directions can be recommended. Firstly, we are going to apply the LMFO to solve more real-world engineering problems. Secondly, it is recommended to develop binary and multiobjective versions of the MFO algorithm.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National Science Foundation of China under Grants no. 61463007 and 6153008.

References

[1] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[2] J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *ACM SIGART Bulletin*, no. 63, p. 49, 1977.

[3] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[4] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the European Conference on Artificial Life*, pp. 134–142, Paris, France, December 1991.

[5] I. Rechenberg, *Evolution Strategy: Nature's Way of Optimization. Optimization: Methods and Applications, Possibilities and Limitations*, Springer, Berlin, Germany, 1989.

[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] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[8] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

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

[10] X.-S. Yang, "A new metaheuristic bat-inspired Algorithm," *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010.

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

[12] R. Rajabioun, "Cuckoo optimization algorithm," *Applied Soft Computing*, vol. 11, no. 8, pp. 5508–5518, 2011.

[13] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[14] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3-4, pp. 267–289, 2010.

[15] X. S. Yang, "Firefly algorithm," in *Engineering Optimization*, pp. 221–230, John Wiley & Sons, 2010.

[16] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: ray optimization," *Computers and Structures*, vol. 112-113, no. 4, pp. 283–294, 2012.

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

[18] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[19] A. F. Kamaruzaman, A. M. Zain, S. M. Yusuf, and A. Udin, "Lévy flight algorithm for optimization problems—a literature review," *Applied Mechanics and Materials*, vol. 421, pp. 496–501, 2013.

[20] P. Barthelemy, J. Bertolotti, and D. S. Wiersma, "A Lévy flight for light," *Nature*, vol. 453, no. 7194, pp. 495–498, 2008.

[21] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Lévy flights in dobe Ju/'hoansi foraging patterns," *Human Ecology*, vol. 35, no. 1, pp. 129–138, 2007.

[22] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[23] I. Pavlyukevich, "Cooling down Lévy flights," *Journal of Physics A. Mathematical and Theoretical*, vol. 40, no. 41, pp. 12299–12313, 2007.

[24] A. M. Reynolds and M. A. Frye, "Free-flight odor tracking in *Drosophila* is consistent with an optimal intermittent scale-free search," *PLoS ONE*, vol. 2, no. 4, article e354, 2007.

[25] M. F. Shlesinger, "Mathematical physics: search research," *Nature*, vol. 443, no. 7109, pp. 281–282, 2006.

[26] J. Xie, Y. Zhou, and H. Chen, "A novel bat algorithm based on differential operator and lévy flights trajectory," *Computational Intelligence and Neuroscience*, vol. 2013, Article ID 453812, 13 pages, 2013.

[27] X. S. Yang, "Appendix a: test problems in optimization," *Engineering Optimization*, pp. 261–266, 2010.

[28] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, *Benchmark Functions for the cec'2008 Special Session and Competition on Large Scale Global Optimization*, Nature Inspired Computation and Applications Laboratory, 2009.

[29] M. B. Dowlatshahi and H. Nezamabadi-Pour, "GGSA: a grouping gravitational search algorithm for data clustering," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 114–121, 2014.

[30] S. Mirjalili and S. Z. M. Hashim, "A new hybrid PSOGSA algorithm for function optimization," in *Proceedings of the International Conference on Computer and Information Application (ICCIA '10)*, pp. 374–377, IEEE, Tianjin, China, November 2010.

[31] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[32] J. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, Springer, Berlin, Germany, 2011.

[33] D. A. Wolfe and M. Hollander, *Nonparametric Statistical Methods*, John Wiley & Sons, New York, NY, USA, 2013.

[34] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002.

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

[36] S. Mirjalili and A. Lewis, "Adaptive gbest-guided gravitational search algorithm," *Neural Computing and Applications*, vol. 25, no. 7-8, pp. 1569–1584, 2014.

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

[38] K. Deb, "Optimal design of a welded beam via genetic algorithms," *AIAA Journal*, vol. 29, no. 11, pp. 2013–2015, 1991.

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

[40] K. S. Lee and Z. W. Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 194, no. 36-38, pp. 3902–3933, 2005.

[41] K. M. Ragsdell and D. T. Phillips, "Optimal design of a class of welded structures using geometric programming," *Journal of Engineering for Industry*, vol. 98, no. 3, pp. 1021–1025, 1976.

[42] A. H. Gandomi and X. S. Yang, *Benchmark Problems in Structural Optimization. Computational Optimization, Methods and Algorithms*, Springer, Berlin, Germany, 2011.

[43] S. Akhtar, K. Tai, and T. Ray, "A socio-behavioral simulation model for engineering design optimization," *Engineering Optimization*, vol. 34, no. 4, pp. 341–354, 2002.

[44] E. Mezura-Montes, C. A. C. Coello, and R. Landa-Becerra, "Engineering optimization using simple evolutionary algorithm," in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pp. 149–156, IEEE Computer Society, November 2003.

[45] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[46] W. Long, W.-Z. Zhang, Y.-F. Huang, and Y.-X. Chen, "A hybrid cuckoo search algorithm with feasibility-based rule for constrained structural optimization," *Journal of Central South University*, vol. 21, no. 8, pp. 3197–3204, 2014.

[47] T. Ray and K. M. Liew, "Society and civilization: an optimization algorithm based on the simulation of social behavior," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 386–396, 2003.

[48] E. Mezura-Montes and C. A. C. Coello, "Useful infeasible solutions in engineering optimization with evolutionary algorithms," in *Proceedings of the 4th Mexican international conference on Advances in Artificial Intelligence (MICAI '05)*, vol. 3789, pp. 652–662, Springer, Monterrey, Mexico, 2005.

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