symmetry

MDPI

Article

An Improved Bat Algorithm Based on Lévy Flights and Adjustment Factors

Yu Li [1], Xiaoting Li [2,*], Jingsen Liu [3] and Ximing Ruan [4]

1 Institute of Management Science and Engineering, and School of Business, Henan University, Kaifeng 475004, China
2 School of Business, Henan University, Kaifeng 475004, China
3 Institute of Intelligent Network Systems, and Software School, Henan University, Kaifeng 475004, China
4 Bristol Business School, University of the West of England, Bristol BS16 1QY, UK
* Correspondence: 104753170952@vip.henu.edu.cn

Received: 29 May 2019; Accepted: 13 July 2019; Published: 15 July 2019

Abstract: This paper proposed an improved bat algorithm based on Lévy flights and adjustment factors (LAFBA). Dynamically decreasing inertia weight is added to the velocity update, which effectively balances the global and local search of the algorithm; the search strategy of Lévy flight is added to the position update, so that the algorithm maintains a good population diversity and the global search ability is improved; and the speed adjustment factor is added, which effectively improves the speed and accuracy of the algorithm. The proposed algorithm was then tested using 10 benchmark functions and 2 classical engineering design optimizations. The simulation results show that the LAFBA has stronger optimization performance and higher optimization efficiency than basic bat algorithm and other bio-inspired algorithms. Furthermore, the results of the real-world engineering problems demonstrate the superiority of LAFBA in solving challenging problems with constrained and unknown search spaces.

Keywords: bat algorithm; inertia weight; Lévy flights; engineering design problem

1. Introduction

Many problems in management can be treated as global optimization problems, and the need to efficiently solve large-scale optimization problems has prompted the development of bio-inspired intelligent optimization algorithms. For example, scholar Holland proposed a genetic algorithm (GA) based on the idea of evolution [1]; Kennedy and Eberhart proposed particle swarm optimization (PSO) [2,3] by referring to the swarm foraging behavior in nature; and Dorigo et al., inspired by ant foraging behavior, proposed the algorithm of ant colony optimization (ACO) [4]. In recent years, a variety of novel swarm intelligent optimization algorithms have been developed, such as bee colony algorithm [5], krill herd algorithm [6], cuckoo search algorithm [7,8] and moth flame optimization algorithm [9] etc. Although a meta-heuristic algorithm has the advantages of simple calculation steps and being easy to understand and implement, it also has the disadvantages of easily falling into local extreme value and low solution accuracy. Sergeyev et al. compared several widely used metaheuristic global optimization methods with Lipschitz deterministic methods by using operational zones, and simulation results show that, in some runs, the metaheuristic methods were getting stuck in the local solutions [10]. Therefore, it is of great significance to research and put forward swarm intelligence optimization algorithms with better performance to enrich the algorithm and expand the application field of the algorithm.

The bat algorithm (BA) is a metaheuristic optimization algorithm proposed by Yang [11]. The algorithm uses the bat's echolocation capability to design an optimization strategy that iterates through

frequency updates. The bat algorithm has the advantages of having less setting parameters, being easy to understand and implement, and having fast convergence. However, it also has drawbacks in balancing global and local search capabilities, it is easy to fall into a local optimum, and the solution accuracy is not high. To overcome these shortcomings, many scholars have made improvements to the bat algorithm. For instance, Ramli and other scholars, in order to improve the global search ability of the bat algorithm, put forward an enhanced bat algorithm (MBA) based on dimensional and inertia weight factor to enhance the convergence [12]. Banati and Chaudhary proposed a multimodal bat algorithm (MMBAIS) with improved search mechanism, which effectively alleviates the problem of early convergence and improves the convergence speed of the algorithm in later phase [13]. Al-Betar studied the alternative selection mechanism in the bat algorithm for global optimization [14]. Li added a mutation switch function to the standard bat algorithm and proposed a bat optimization algorithm (UGBA) that combines uniform variation and Gaussian variation [15]. Asma proposed a directional bat algorithm (dBA) to introduce directional echolocation in the standard bat algorithm to enhance the exploration and exploitation capabilities of the algorithm [16]. Al-Betar and Awadallah proposed an island bat algorithm (iBA) that uses the island model's strategy for bat algorithms to enhance algorithm population diversity to avoid premature convergence [17].

In addition, many scholars are committed to expanding the application fields of bat algorithms. For example, Laudis and other scholars have proposed a multi objective bat algorithm (MOBA) to solve the problem in Very Large-Scale Integration (VLSI) design [18]. Tawhid and Dsouza proposed a hybrid binary bat enhanced particle swarm optimization algorithm (HBBEPSO) to solve the feature selection problems [19]. Osaba proposed an improved discrete bat algorithm (IBA) for solving symmetric and asymmetric traveling salesman problems [20]. Mohamed and Moftah proposed a novel multiobjective binary bat algorithm for simultaneous ranking and selection of keystroke dynamics features [21]. Hamidzadeh used the ergodicity of chaotic algorithm and the automatic conversion of bat algorithm global search and local search to construct a weighted SVDD method based on chaotic bat algorithm (WSVDD–CBA) for effective data description [22]. Qi and others proposed a discretized bat algorithm for solving vehicle routing problems with time windows [23]. Bekdaş and others used the bat algorithm to modify the tuning quality and proposed an effective method to solve the damper optimization problem [24]. Ameur and Sakly proposed a new hardware implementation of the bat algorithm's field-programmable gate array (FPGA) [25]. Chaib and other scholars used the bat algorithm to optimize the design and adjust the novel fractional-order PID power system stabilizer [26]. Mohammad et al. used the bat algorithm to successfully solve the complex engineering problem of dam–reservoir operation [27]. In order to better solve the problem of mobile robot path planning, Liu et al. proposed a bat algorithm with reverse learning and tangent random exploration mechanism [28].

In order to improve the performance of the bat algorithm, this paper proposes an improved bat algorithm based on Lévy flights and adjustment factors (LAFBA). The search mechanism of Lévy flight is introduced to update the position of the bat algorithm, which can effectively help the algorithm maintain the diversity of the population and improve the global search ability. In addition, the introduction of the dynamic decreasing inertia weight and speed adjustment factor enables the algorithm to balance global exploration and local exploitation, improve the accuracy of the algorithm, and accelerate the later convergence speed.

The efficiency of the LAFBA is tested by solving 10 classical optimization functions and 2 structural optimization problems. The results obtained show that LAFBA is competitive in comparison with other state-of-the-art optimization methods.

2. Enhanced Bat Algorithm

2.1. Bat Algorithm

The bat algorithm is a swarm intelligent optimization algorithm that simulates the behavior of bats using the echolocation ability to prey. It realizes velocity and position update through the change of frequency f. The implementation of the algorithm is based on the following three idealized rules [11]:

Rule 1: Bats use echolocation to sense distance and can distinguish between prey and obstacles.

Rule 2: Bats fly randomly with the velocity v_i at the position x_i with a varying frequency f_i (from a minimum frequency f_{min} to a maximum frequency f_{max}) or a varying wavelength λ and sound loudness A_0 to search for prey. The wavelength (or frequency) of the emitted pulse can be automatically adjusted according to the proximity of the target, and the rate r of pulse emission, $r \in [0, 1]$, is also adjusted.

Rule 3: Assume that the loudness varies from the largest positive value A_0 to the minimum constant value A_{min}.

Based on the above rules, the position vector of the bat represents a solution in the search space. Since the global optimal position vector in the search space is not known a priori, the algorithm randomly initializes the bats. The initialize equation as follows:

$$x_{i,j} = x_{lb} + (x_{ub} - x_{lb})rand, \tag{1}$$

for i^{th} bat, where $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, d$; n represents the population size, d represents the dimension of the search space. x_{lb} and x_{ub} are lower and upper bounds for j^{th} dimension, and *rand* is a random number between [0,1]. Bat individuals, in the d-dimensional search space, update frequency, velocity vector, and position vector according to the following equations:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{2}$$

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right)f_i, \tag{3}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{4}$$

where β is a random number between [0, 1]. x_* represents the current global best solution, and the bat updates the velocity and position according to the change of frequency f.

When the bat performs a local search, the position update equation is as follows:

$$x_{new} = x_{old} + \varepsilon A^t, \tag{5}$$

where $\varepsilon \in [-1, 1]$ which is set as 0.001 in this paper. A^t represents the average loudness in current iteration.

As the bat approaches the prey, the loudness continues to decrease, while the rate of pulse emission increases. Loudness A_i which is a vector of values for all bats, which assists in updating the bat location. Rate r_i of pulse emission which is a vector for all bats controlling the diversification of bat algorithm. The update equation is as follows:

$$A_i^t = \alpha \cdot A_i^{t-1}, \tag{6}$$

$$r_i^t = r_i^0\left(1 - e^{-\gamma t}\right). \tag{7}$$

Here, α is the pulse emission loudness attenuation coefficient. When $t \to \infty$, for any value at $\alpha \in [0, 1]$ and $\gamma > 0$, it has $A_i^t \to 0$ and $r_i^t \to r_i^0$.

2.2. Dynamically Decreasing Inertia Weight

In the bat algorithm velocity update equation, right before the previous generation velocity v_i^{t-1} is a constant term coefficient 1. The fixed coefficient is not conducive to the algorithm's exploration of the

global search space, and also reduces the flexibility of the bat individual in the algorithm, thus making it easy to fall into local optimum. The particle swarm optimization algorithm balances the global and local search by adjusting the inertia weight [29]. A larger inertia weight makes the individual's change range larger, which is conducive to global exploration and local exploitation. A smaller inertia weight makes the individual change range smaller, which is advantageous for the algorithm to perform a local search on the optimization function. Inspired by this, this paper introduces the dynamic decreasing inertia weight, and changes the velocity update equation to

$$v_i^t = w_i(t)v_i^{t-1} + \left(x_i^t - x_*\right)f_i, \tag{8}$$

$$w_i(t) = w_{max} - (w_{max} - w_{min})arctan\left(\frac{4t}{N\_gen}\right) \tag{9}$$

where w_{max} and w_{min} respectively represent the maximum and minimum weight, t represents the current number of iterations, and N_gen represents the maximum number of iterations. In this paper, w_{max} takes a value of 0.9 and w_{min} takes a value of 0.42. The inverse tangent function *arctan* is a monotonically increasing function [30], so $w_i(t)$ is a monotonically decreasing function. The inertia of the early period is great, while the inertia of the late period is small, which can balance the global and local search of the algorithm well. In this paper, the arctangent function domain is [0,4]. The change of the arctangent function in the domain is gradually slowed down, making the decrease of $w_i(t)$ rapid in the early stage but slow in the late stage. The algorithm is converted from a fast global search to a slow local search, which can effectively improve the speed and accuracy of the algorithm.

2.3. Lévy Flights

In 1926, the French mathematician Paul Lévy proposed Lévy flights. Lévy flights phenomenon is very common in nature. The foraging activities of creatures, such as wasps, jackals, monkeys, and human hunting behaviors, are all consistent with the random motion model of Lévy's flights [31]. For example, some herbivores randomly move around in a given area to find a source of grass, but if they cannot find it, they will quickly go to another area and then resume the previous way of walking. This can effectively avoid wasting time in a place with insufficient resource. Along the iterative process, an agglomeration phenomenon occurs in the individual bat algorithm, the population diversity is reduced, the global search ability is undermined, and the algorithm easily falls into premature convergence of local optimum. In order to solve this problem, this paper proposes to add the Lévy flight search mechanism to the bat algorithm, and modify the bat position vector update equation as below:

$$x_i^t = \text{Lévy}(\text{d})x_i^{t-1} + v_i^t. \tag{10}$$

Here, t is the current number of iterations, and d is the dimension of the search space. The Lévy flight is calculated as follows [32]:

$$\text{Lévy}(\text{x}) = 0.01\frac{r_1\delta}{|r_2|^{\frac{1}{\psi}}}, \tag{11}$$

where r_1, r_2 are two random numbers in [0,1], ψ is a constant 1.5, and δ is calculated as follows:

$$\delta = \left(\frac{\Gamma(1+\psi)\sin\left(\frac{\pi\psi}{2}\right)}{\Gamma\left(\frac{1+\psi}{2}\right)\psi 2^{\left(\frac{\psi-1}{2}\right)}}\right)^{\frac{1}{\psi}}, \tag{12}$$

where $\Gamma(\text{x}+1) = \text{x}!$, 50 step sizes have been drawn to form a consecutive 50 steps of Lévy flights as shown in Figure 1 [32].

Figure 1. A series of 50 consecutive steps of Lévy flights.

Figure 1 intuitively shows Lévy's ability to suddenly move a long distance after several short distance movements. In this paper, the Lévy flight mechanism is introduced to the bat position update. On the one hand, it can effectively avoid the overreliance of the bat position change on the previous generation position information and ensure the diversity of the population. On the other hand, the random walk mode of Lévy flight that changes large steps suddenly after a series of small steps gives the bat individual the ability to jump suddenly, which helps the algorithm to jump out of the local optimum, avoid the premature convergence of the algorithm, and improve the global search ability.

2.4. Speed Adjustment Factor

In order to ensure the efficiency of the algorithm, this paper designs a speed adjustment factor $c_i(t)$ based on the dimension of the optimization function to be solved and the number of iterations of the algorithm, which is applied to the bat position update. Along the iteration progress, the speed adjustment factor changes from large to small, and the position movement step size changes from large to small, which satisfies the requirement of the algorithm for global search in the early iteration stage and local search in the later iteration stage, enabling the bat algorithm to search for the optimum when solving each generation of the movement in space. The position vector update equation in this article is as below:

$$x_i^t = \text{Lévy}(\text{d}) x_i^{t-1} + v_i^t c_i(t) \tag{13}$$

$$c_i(t) = \theta^d \cdot e^{-\frac{t}{N\_gen}}. \tag{14}$$

Here, θ can be any value between [0,1], which is set as 0.01 in this paper. d represents the dimension of the search space. At the same dimensional level, $c_i(t)$ gradually decreases as the number of iterations increases, making the position update of the algorithm change from large-scale movement in the early stage to small-scale movement in the late stage, and accelerates the convergence speed of the algorithm. When the algorithm solves the function, the search difficulty will increase with the increase of dimension, and the problems of low solution accuracy and reduced solution speed may occur. With the increase of the solution dimension, the speed adjustment factor proposed in this paper will be maintained at a relatively low level, which can speed up the algorithm and improve the accuracy of the solution when solving high-dimensional functions.

2.5. The Pseudocode of the LAFBA

The dynamic declining inertia weight added in the velocity update can effectively balance the global and local search of the algorithm; the Lévy flight search mechanism introduced to the location update can ensure the diversity of the population and improve the global search ability of the algorithm, and the construction of the speed adjustment factor can effectively improve the solution accuracy and speed of the algorithm. The pseudocode of the LAFBA is shown as follows:

1. Define objective function $f(x)$, $x = (x_1, \ldots, x_d)^T$
2. Set the initial value of population size n, α, γ, and N_gen
3. Initialize pulse rates r_i and loudness A_i
4. Initialize the bat population (Equation (1))
5. Evaluate and find x_* where $x_* \in \{1, 2, \ldots, n\}$
6. **while** $t \leq N\_gen$
7. **for** i = 1 to n
8. Adjust frequency (Equation (2))
9. Update inertia weight (Equation (9)) and Lévy(d) (Equation (11))
10. Update the velocity (Equation (8)) and position vector (Equation (13)) of the bat
11. **if** (rand > r_i)
12. Select a solution among the best solutions
13. Generate a local solution around selected best (Equation (5))
14. **end if**
15. Evaluate objective function
16. **if** (rand < A_i & $f(x_i) < f(x_*)$)
17. $x_* = x_i$
18. $f(x_*) = f(x_i)$
19. Increase r_i (Equation (7))
20. Reduce A_i (Equation (6))
21. **end if**
22. **if** ($f(x_i^{t+1}) < f(x_*)$)
23. Update the best solution x_*
24. **end if**
25. **end for**
26. Rank the bats and find the current best x_*
27. $t = t + 1$
28. **end while**
29. Return x_*, postprocess results and visualization

3. Numerical Simulation and Analysis

In order to test the performance of the improved algorithm proposed in this paper, ten standard optimization functions are selected [33], and the algorithm is compared with the standard bat algorithm (BA) [11], particle swarm optimization (PSO) algorithm [2], moth flame optimization (MFO) algorithm [9], sine cosine algorithm (SCA) [34], and butterfly optimization algorithm (BOA) [35].

3.1. Parameters Setting

We have tried to use different population sizes from n = 10 to 100, and we found that for most problems, n = 20 is sufficient. Therefore, we use a fixed population n = 20 for all simulations. In order to guarantee the comparability and fairness of the simulation experiment, the same parameters are set for the four algorithms: the population size n = 20, and the number of iterations N_gen = 500. For BA and LAFBA, $\alpha = \gamma = 0.9$, the sound loudness A_i = 0.25 and the rate of pulse emission r_i = 0.5. For PSO, we have used the standard version with learning parameters c1 = c2 = 2. For BOA, modular modality c is 0.01, power exponent a is increased from 0.1 to 0.3 and the probability switch p = 0.8.

The simulation environment is MATLAB 2014a, the operating system is Windows10 Home Chinese version, 4.00GB running memory, the processor is Intel(R) Core (TM) i5-6200U CPU @ 2.30 GHz 2.40 GHz.

3.2. Standard Optimization Functions

Optimization functions are as below:

(1) Griewank Function

$$f_1(x) = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{15}$$

The function definition field is [−600, 600] and the theoretical optimal value is 0. This function is a continuous, differentiable, non-separable, scalable, multimodal multi-extreme function, with many local optima. The higher the function dimension, the more the number of local optima. It is extremely difficult to optimize and is thus often used to test the exploration and exploitation capabilities of the algorithm.

(2) Quartic Function

$$f_2(x) = \sum_{i=1}^{d} x_i^4 \tag{16}$$

The function definition field is [−1.28, 1.28], and the theoretical optimal value is 0. This function is a continuous, differentiable, separable, scalable, and high-dimensional unimodal function. Unimodal functions are often used to test the convergence speed of an algorithm.

(3) Ackley Function

$$f_3(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} \tag{17}$$

The function definition field is [−30, 30] and the theoretical optimal value is 0. This function is a continuous, differentiable, non-separable, scalable, and complex nonlinear multimodal function which is formed by the superposition of a moderately amplified cosine wave to an exponential function. The undulation of the function surface makes the search of this function more complicated, and there are a large number of local optima.

(4) Rastrigin Function

$$f_4(x) = \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i) + 10\right] \tag{18}$$

The function definition field is [−5.12, 5.12], and the function theory optimal value is 0. This function is a high-dimensional multimodal function. There are about 10d local minimum values in the solution space. The peak shape of the function fluctuates volatility, making the global search rather difficult.

(5) Schaffer Function

$$f_5(x) = 0.5 + \frac{sin^2\sqrt{\sum_{i=1}^{d} x_i^2} - 0.5}{\left[1 + 0.001\left(\sum_{i=1}^{d} x_i^2\right)\right]^2} \tag{19}$$

The function definition field is [−10, 10] and the theoretical optimal value is 0. The fluctuation is fierce, and it is difficult to find the global optimal value. The function graph presents a "four-corner hat" shape, which is a continuous, differentiable, non-separable, scalable, and typical multimodal function. The global optimal position is in the center of the brim, and the relative search area is very small.

(6) Sphere Function

$$f_6(x) = \sum_{i=1}^{d} x_i^2 \tag{20}$$

The function definition field is [−5.12, 5.12], and the theoretical optimal value is 0. This function is a continuous, differentiable, separable, scalable, and classic high-dimensional unimodal function.

(7) Axis Parallel Function

$$f_7(x) = \sum_{i=1}^{d} i x_i^2 \tag{21}$$

The function definition field is [−5.12, 5.12], and the theoretical optimal value is 0. This function is a unimodal function.

(8) Zakharov Function

$$f_8(x) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^4 \quad (22)$$

The function definition field is [−10,10], and the theoretical optimal value is 0. This function is a continuous, differentiable, non-separable, scalable, multimodal function.

(9) Schwefel 2.21 Function

$$f_9(x) = max_i\{|x_i|, 1 \le i \le d\} \quad (23)$$

The function definition field is [−10, 10] and the theoretical optimal value is 0. This function is a continuous, non-differentiable, separable, scalable and unimodal function.

(10) Schwefel 1.2 Function

$$f_{10}(x) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2 \quad (24)$$

The function definition field is [−5.12, 5.12], and the theoretical optimal value is 0. This function is a continuous, differentiable, non-separable, scalable, unimodal function.

3.3. Simulation Result Comparison and Analysis

The six algorithms are run on the above 10 optimization functions 30 times independently in dimensions 10, 30, and 100, and the function value obtained each time was recorded. According to the results of the algorithm in different dimensions, four criteria were collected and analyzed: best, worst, average, and standard deviation (SD). The data are shown in Tables 1–3.

With the statistical test, we can make sure that the results are not generated by chance [36]. The Wilcoxon rank-sum test [37,38] was conducted in this experiment and p_value < 0.05 and h = 1 indicate that the difference between the two data is significant. The statistical comparison results between LAFBA and other 5 algorithms are shown in Table 4.

With the increase of the solution dimension, the difficulty faced by the algorithm to achieve the best result increases, and the problems of low solution accuracy, slowing down of the solving speed, or even failure to achieve the best result may occur. Therefore, this paper sets the three dimensions of 10D, 30D, and 100D to observe the change in the performance of the six algorithms in different dimensions.

It should be noted that the best optimal solution obtained is highlighted in bold font. From the results presented in Tables 1–3, under different dimensions, LAFBA is able to obtain the best values among 10 test functions in comparison with the BA, PSO, MFO, and SCA. When d = 10, compared to the BOA, the LAFBA obtained better results on 9 benchmark functions except for F3. When d = 30/100, BOA obtained the better mean value of F9 was better than the LAFBA. As the dimension increases, the difficulty in solving the function increases. The improved LAFBA proposed in this paper can still solve each optimization function effectively with the smallest standard deviation, indicating a good stability of the function. That is, the robustness is good.

As shown in Table 4, both the p-value and h = 1 indicate the rejection of the null hypothesis of equal medians at the default 5% significance level. This means that the superiority of LAFBA is statistically significant. p_value $\geq$ 0.05 and h = 0 have been underlined, and LAFBA is significantly different from BOA with the exception of F3.

In summary, the comparison with the other algorithms shows the superiority of LAFBA in several benchmarks. Among the six algorithms, LAFBA had the best performance.

Table 1. Comparison results of Lévy flights and adjustment factors (LAFBA) and other five algorithms for benchmark functions with D = 10. BA, bat algorithm; PSO, particle swarm optimization; MFO, moth flame optimization; SCA, sine cosine algorithm; BOA, butterfly optimization algorithm.

Algorithm	Function	Best	Worst	Average	SD	Function	Best	Worst	Average	SD
LAFBA	F1	**0**	**0**	**0**	**0**	F6	**0**	$\mathbf{7.89 \times 10^{-16}}$	$\mathbf{1.05 \times 10^{-16}}$	$\mathbf{1.98 \times 10^{-16}}$
BA		1.27×10^{1}	1.46×10^{2}	7.99×10^{1}	3.52×10^{1}		7.48×10^{-2}	8.76×10^{1}	3.03×10^{1}	2.63×10^{1}
PSO		2.17×10^{-1}	2.45	9.51×10^{-1}	4.52×10^{-1}		5.19×10^{-6}	6.80×10^{-3}	8.89×10^{-4}	1.54×10^{-3}
MFO		7.92×10^{-11}	7.58×10^{-1}	1.51×10^{-1}	1.44×10^{-1}		9.89×10^{-17}	1.48×10^{-13}	1.10×10^{-14}	2.82×10^{-14}
SCA		1.34×10^{-14}	8.98×10^{-1}	1.56×10^{-1}	2.16×10^{-1}		4.64×10^{-18}	1.69×10^{-10}	7.33×10^{-12}	3.16×10^{-11}
BOA		3.11×10^{-14}	1.45×10^{-12}	2.95×10^{-13}	3.26×10^{-13}		5.84×10^{-12}	1.03×10^{-11}	8.24×10^{-12}	1.16×10^{-12}
LAFBA	F2	**0**	$\mathbf{3.08 \times 10^{-31}}$	$\mathbf{1.63 \times 10^{-32}}$	$\mathbf{5.77 \times 10^{-32}}$	F7	**0**	$\mathbf{1.53 \times 10^{-15}}$	$\mathbf{1.67 \times 10^{-16}}$	$\mathbf{3.98 \times 10^{-16}}$
BA		6.74×10^{-14}	7.29×10^{-13}	2.85×10^{-13}	1.51×10^{-13}		4.63×10^{-6}	4.76×10^{1}	7.93	9.83
PSO		2.62×10^{-11}	8.69×10^{-7}	7.48×10^{-8}	1.91×10^{-7}		5.19×10^{-6}	6.80×10^{-3}	8.89×10^{-4}	1.54×10^{-3}
MFO		2.11×10^{-29}	3.17×10^{-21}	1.11×10^{-22}	5.78×10^{-22}		1.17×10^{-16}	5.02×10^{-13}	5.50×10^{-14}	1.16×10^{-13}
SCA		1.15×10^{-30}	7.54×10^{-20}	3.46×10^{-21}	1.42×10^{-20}		9.03×10^{-18}	4.87×10^{-12}	3.70×10^{-13}	9.31×10^{-13}
BOA		4.12×10^{-15}	1.17×10^{-14}	7.28×10^{-15}	1.84×10^{-15}		6.40×10^{-12}	1.26×10^{-11}	9.46×10^{-12}	1.46×10^{-12}
LAFBA	F3	**0**	2.74×10^{-8}	7.01×10^{-9}	8.65×10^{-9}	F8	**0**	$\mathbf{3.26 \times 10^{-15}}$	$\mathbf{4.89 \times 10^{-16}}$	$\mathbf{9.14 \times 10^{-16}}$
BA		1.21×10^{1}	1.94×10^{1}	1.74×10^{1}	1.51		1.31	8.01×10^{1}	2.58×10^{1}	1.92×10^{1}
PSO		2.94×10^{-3}	1.17	3.73×10^{-1}	5.25×10^{-1}		1.05×10^{-3}	2.96×10^{-1}	5.96×10^{-2}	7.38×10^{-2}
MFO		2.99×10^{-8}	5.09	4.07×10^{-1}	1.05		2.45×10^{-6}	2.51×10^{2}	2.57×10^{1}	5.80×10^{1}
SCA		7.17×10^{-10}	1.69×10^{-4}	7.00×10^{-6}	3.07×10^{-5}		1.81×10^{-9}	9.82×10^{-3}	9.40×10^{-4}	2.48×10^{-3}
BOA		1.65×10^{-9}	$\mathbf{6.14 \times 10^{-9}}$	$\mathbf{3.49 \times 10^{-9}}$	$\mathbf{1.20 \times 10^{-9}}$		7.47×10^{-12}	1.14×10^{-11}	9.61×10^{-12}	1.09×10^{-12}
LAFBA	F4	**0**	$\mathbf{9.59 \times 10^{-14}}$	$\mathbf{1.50 \times 10^{-14}}$	$\mathbf{2.48 \times 10^{-14}}$	F9	**0**	1.23×10^{-8}	$\mathbf{2.51 \times 10^{-9}}$	4.09×10^{-9}
BA		1.79×10^{1}	8.76×10^{1}	4.78×10^{1}	1.95×10^{1}		1.65	5.08	3.46	9.33
PSO		1.31	3.02×10^{1}	9.41	6.93		2.14×10^{-2}	2.94×10^{-1}	8.88×10^{-2}	5.66×10^{-2}
MFO		5.97	6.28×10^{1}	2.70×10^{1}	1.39×10^{1}		9.20×10^{-2}	4.80	1.37	1.16
SCA		2.56×10^{-12}	2.41×10^{1}	2.44	6.64		1.09×10^{-7}	3.34×10^{-3}	3.59×10^{-4}	6.97×10^{-4}
BOA		4.26×10^{-14}	5.67×10^{1}	3.10×10^{1}	2.18×10^{1}		3.77×10^{-9}	$\mathbf{5.34 \times 10^{-9}}$	4.50×10^{-9}	$\mathbf{4.30 \times 10^{-10}}$
LAFBA	F5	**0**	$\mathbf{6.11 \times 10^{-16}}$	$\mathbf{8.23 \times 10^{-17}}$	$\mathbf{1.52 \times 10^{-16}}$	F10	**0**	$\mathbf{9.09 \times 10^{-16}}$	$\mathbf{9.00 \times 10^{-17}}$	$\mathbf{2.36 \times 10^{-16}}$
BA		9.72×10^{-3}	2.28×10^{-1}	1.31×10^{-1}	5.67×10^{-2}		6.91	1.58×10^{2}	6.14×10^{1}	3.83×10^{1}
PSO		9.72×10^{-3}	7.82×10^{-2}	2.67×10^{-2}	1.68×10^{-2}		7.82×10^{-4}	9.71×10^{-2}	2.89×10^{-2}	3.03×10^{-2}
MFO		3.72×10^{-2}	2.28×10^{-1}	1.28×10^{-1}	4.60×10^{-2}		1.59×10^{-5}	1.75×10^{1}	3.35	6.25
SCA		9.72×10^{-3}	3.72×10^{-2}	1.06×10^{-2}	5.02×10^{-3}		8.55×10^{-10}	0.02047	9.25×10^{-4}	0.003736
BOA		3.72×10^{-2}	8.08×10^{-2}	7.18×10^{-2}	1.47×10^{-2}		5.97×10^{-12}	1.11×10^{-11}	9.02×10^{-12}	1.38×10^{-12}

Table 2. comparison results of LAFBA and other five algorithms for benchmark functions with D = 30.

Algorithm	Function	Best	Worst	Average	SD	Function	Best	Worst	Average	SD
LAFBA	F1	**0**	$\mathbf{1.33 \times 10^{-15}}$	$\mathbf{1.42 \times 10^{-16}}$	$\mathbf{3.01 \times 10^{-16}}$	F6	**0**	$\mathbf{1.50 \times 10^{-14}}$	$\mathbf{3.34 \times 10^{-15}}$	$\mathbf{4.40 \times 10^{-15}}$
BA		9.85×10^{1}	5.36×10^{2}	3.23×10^{2}	1.08×10^{2}		1.85	3.35×10^{2}	1.86×10^{2}	8.73×10^{1}
PSO		5.80×10^{-2}	3.46×10^{-1}	1.66×10^{-1}	7.21×10^{-2}		1.49×10^{-1}	9.03×10^{-1}	3.74×10^{-1}	1.56×10^{-1}
MFO		9.48×10^{-1}	2.71×10^{2}	2.22×10^{1}	6.11×10^{1}		6.42×10^{-3}	2.62×10^{1}	3.55	9.05
SCA		5.39×10^{-1}	7.04	1.49	1.21		8.99×10^{-5}	1.55	9.02×10^{-2}	2.81×10^{-1}
BOA		7.17×10^{-13}	1.73×10^{-11}	6.82×10^{-12}	4.58×10^{-12}		9.88×10^{-12}	1.20×10^{-11}	1.10×10^{-11}	5.75×10^{-13}
LAFBA	F2	**0**	$\mathbf{1.14 \times 10^{-28}}$	$\mathbf{6.72 \times 10^{-30}}$	$\mathbf{2.13 \times 10^{-29}}$	F7	**0**	$\mathbf{1.75 \times 10^{-13}}$	$\mathbf{3.06 \times 10^{-14}}$	$\mathbf{5.03 \times 10^{-14}}$
BA		2.18×10^{-11}	6.16×10^{-8}	2.18×10^{-9}	1.14×10^{-8}		4.00×10^{1}	1.31×10^{3}	5.37×10^{2}	2.94×10^{2}
PSO		4.78×10^{-4}	2.69	1.16×10^{-1}	4.96×10^{-1}		2.22	3.37×10^{1}	8.28	8.07
MFO		3.59×10^{-6}	2.86×10^{-3}	2.43×10^{-4}	5.34×10^{-4}		3.87×10^{-2}	7.87×10^{2}	2.01×10^{2}	2.24×10^{2}
SCA		5.29×10^{-7}	2.01×10^{-1}	1.03×10^{-2}	3.66×10^{-2}		1.44×10^{-3}	6.09	4.90×10^{-1}	1.12
BOA		8.92×10^{-15}	1.56×10^{-14}	1.15×10^{-14}	1.35×10^{-15}		1.10×10^{-11}	1.37×10^{-11}	1.23×10^{-11}	7.91×10^{-13}
LAFBA	F3	**0**	1.04×10^{-7}	2.42×10^{-8}	3.32×10^{-8}	F8	**0**	$\mathbf{3.70 \times 10^{-14}}$	$\mathbf{7.39 \times 10^{-15}}$	$\mathbf{1.15 \times 10^{-14}}$
BA		1.36×10^{1}	1.90×10^{1}	1.75×10^{1}	1.15		1.21×10^{1}	2.27×10^{3}	2.42×10^{2}	4.05×10^{2}
PSO		1.52	4.28	2.90	5.77×10^{-1}		5.01×10^{1}	4.20×10^{2}	1.65×10^{2}	8.14×10^{1}
MFO		1.25	1.98×10^{1}	1.51×10^{1}	5.34		2.06×10^{2}	9.81×10^{2}	5.09×10^{2}	1.97×10^{2}
SCA		3.78×10^{-2}	2.03×10^{1}	7.69	8.97		4.31×10^{1}	2.05×10^{2}	1.26×10^{2}	4.20×10^{1}
BOA		5.53×10^{-9}	$\mathbf{7.04 \times 10^{-9}}$	$\mathbf{6.24 \times 10^{-9}}$	$\mathbf{3.84 \times 10^{-10}}$		8.71×10^{-12}	1.18×10^{-11}	1.05×10^{-11}	7.86×10^{-13}
LAFBA	F4	**0**	$\mathbf{2.49 \times 10^{-12}}$	$\mathbf{2.57 \times 10^{-13}}$	$\mathbf{6.51 \times 10^{-13}}$	F9	**0**	6.42×10^{-8}	1.62×10^{-8}	2.38×10^{-8}
BA		5.97×10^{1}	2.77×10^{2}	1.43×10^{2}	5.73×10^{1}		3.80	8.41	6.17	1.05
PSO		$6.26\text{E} \times 10^{1}$	1.39×10^{2}	9.11×10^{1}	2.09×10^{1}		4.02×10^{-1}	1.49	7.33×10^{-1}	2.34×10^{-1}
MFO		1.24×10^{2}	2.84×10^{2}	1.75×10^{2}	3.39×10^{1}		5.80	8.48	7.31	6.34×10^{-1}
SCA		1.476745263	1.48×10^{2}	4.68×10^{1}	3.24×10^{1}		1.11	6.68	3.98	1.26
BOA		**0**	2.19×10^{2}	3.93×10^{1}	8.01×10^{1}		4.30×10^{-9}	$\mathbf{5.59 \times 10^{-9}}$	$\mathbf{5.13 \times 10^{-9}}$	$\mathbf{2.75 \times 10^{-10}}$
LAFBA	F5	**0**	$\mathbf{1.64 \times 10^{-14}}$	$\mathbf{2.62 \times 10^{-15}}$	$\mathbf{4.82 \times 10^{-15}}$	F10	**0**	$\mathbf{6.76 \times 10^{-14}}$	$\mathbf{1.10 \times 10^{-14}}$	$\mathbf{1.82 \times 10^{-14}}$
BA		1.78×10^{-1}	3.73×10^{-1}	3.06×10^{-1}	6.35×10^{-2}		1.57×10^{2}	2.68×10^{3}	7.54×10^{2}	4.79×10^{2}
PSO		3.72×10^{-2}	2.28×10^{-1}	9.21×10^{-2}	3.74×10^{-2}		3.53	3.39×10^{1}	1.32×10^{1}	7.06
MFO		3.12×10^{-1}	3.73×10^{-1}	3.42×10^{-1}	1.72×10^{-2}		7.39	1.65×10^{2}	6.46×10^{1}	3.86×10^{1}
SCA		3.72×10^{-2}	1.27×10^{-1}	4.87×10^{-2}	2.21×10^{-2}		3.02	7.53×10^{1}	3.54×10^{1}	1.91×10^{1}
BOA		7.85×10^{-2}	1.27×10^{-1}	1.17×10^{-1}	1.87×10^{-2}		9.36×10^{-12}	1.23×10^{-11}	1.11×10^{-11}	6.22×10^{-14}

Table 3. Comparison results of LAFBA and other five algorithms for benchmark functions with D = 100.

Algorithm	Function	Best	Worst	Average	SD	Function	Best	Worst	Average	SD
LAFBA	F1	**0**	$\mathbf{1.67 \times 10^{-15}}$	$\mathbf{2.87 \times 10^{-16}}$	$\mathbf{5.31 \times 10^{-16}}$	F6	**0**	$\mathbf{1.81 \times 10^{-13}}$	$\mathbf{5.31 \times 10^{-14}}$	$\mathbf{6.00 \times 10^{-14}}$
BA		5.33×10^{2}	2.05×10^{3}	1.31×10^{3}	3.86×10^{2}		4.33×10^{2}	2.06×10^{3}	1.15×10^{3}	4.26×10^{2}
PSO		1.11	1.03×10^{1}	4.16	2.19		1.56	6.14×10^{1}	1.35×10^{1}	1.38×10^{1}
MFO		4.57×10^{2}	1.03×10^{3}	6.68×10^{2}	1.29×10^{2}		1.27×10^{2}	2.41×10^{2}	1.90×10^{2}	3.25×10^{1}
SCA		1.40×10^{1}	2.31×10^{2}	1.01×10^{2}	6.37×10^{1}		3.02	9.46×10^{1}	3.31×10^{1}	2.13×10^{1}
BOA		4.79×10^{-12}	1.99×10^{-11}	1.29×10^{-11}	4.35×10^{-12}		1.09×10^{-11}	1.34×10^{-11}	1.19×10^{-11}	5.62×10^{-13}
LAFBA	F2	**0**	$\mathbf{5.18 \times 10^{-27}}$	$\mathbf{5.64 \times 10^{-28}}$	$\mathbf{1.11 \times 10^{-27}}$	F7	**0**	$\mathbf{5.14 \times 10^{-12}}$	$\mathbf{1.14 \times 10^{-12}}$	1.90×10^{-12}
BA		1.58×10^{-7}	1.51	2.02×10^{-1}	4.22×10^{-1}		2.37×10^{3}	2.11×10^{4}	9.63×10^{3}	4.51×10^{3}
PSO		1.28	1.31×10^{1}	4.69	2.88		2.12×10^{2}	4.43×10^{3}	6.72×10^{2}	9.23×10^{2}
MFO		2.76	1.33×10^{1}	7.31	2.53		5.75×10^{3}	1.40×10^{4}	9.27×10^{3}	2.35×10^{3}
SCA		1.61	9.52	4.29	1.89		2.39×10^{2}	3.80×10^{3}	1.17×10^{3}	7.31×10^{2}
BOA		1.10×10^{-14}	1.58×10^{-14}	1.29×10^{-14}	1.02×10^{-15}		1.19×10^{-11}	1.53×10^{-11}	1.35×10^{-11}	$\mathbf{8.61 \times 10^{-13}}$
LAFBA	F3	**0**	1.53×10^{-7}	3.86×10^{-8}	5.92×10^{-8}	F8	**0**	$\mathbf{1.57 \times 10^{-12}}$	$\mathbf{1.68 \times 10^{-13}}$	$\mathbf{3.74 \times 10^{-13}}$
BA		1.51×10^{1}	1.92×10^{1}	1.78×10^{1}	8.52×10^{-1}		3.44×10^{2}	1.74×10^{3}	8.28×10^{2}	2.86×10^{2}
PSO		4.56	8.12	6.12	9.17×10^{-1}		7.99×10^{2}	3.06×10^{3}	1.52×10^{3}	5.23×10^{2}
MFO		1.93×10^{1}	1.99×10^{1}	1.97×10^{1}	1.62×10^{-1}		2.39×10^{3}	5.00×10^{3}	3.99×10^{3}	6.68×10^{2}
SCA		8.28	2.06×10^{1}	1.68×10^{1}	4.74		9.96×10^{2}	2.00×10^{3}	1.44×10^{3}	2.26×10^{2}
BOA		5.14×10^{-9}	$\mathbf{6.81 \times 10^{-9}}$	$\mathbf{5.85 \times 10^{-9}}$	$\mathbf{3.48 \times 10^{-10}}$		8.18×10^{-12}	1.19×10^{-11}	1.04×10^{-11}	8.34×10^{-13}
LAFBA	F4	**0**	$\mathbf{3.41 \times 10^{-11}}$	$\mathbf{1.03 \times 10^{-11}}$	$\mathbf{1.11 \times 10^{-11}}$	F9	**0**	1.80×10^{-7}	4.13×10^{-8}	6.89×10^{-8}
BA		2.02×10^{2}	8.14×10^{2}	4.60×10^{2}	1.49×10^{2}		5.36	9.19	7.04	1.06
PSO		4.28×10^{2}	7.24×10^{2}	5.65×10^{2}	6.46×10^{1}		1.19	2.84	1.77	4.13×10^{-1}
MFO		$8.15E \times 10^{2}$	1.10×10^{3}	9.19×10^{2}	7.34×10^{-3}		8.95	9.69	9.37	2.02×10^{-1}
SCA		3.24×10^{1}	6.68×10^{2}	2.59×10^{2}	1.43×10^{2}		8.50	9.47	9.15	2.18×10^{-1}
BOA		**0**	3.51×10^{-1}	1.17×10^{-2}	6.40×10^{-2}		4.66×10^{-9}	$\mathbf{5.89 \times 10^{-9}}$	$\mathbf{5.28 \times 10^{-9}}$	$\mathbf{2.71 \times 10^{-10}}$
LAFBA	F5	**0**	$\mathbf{2.66 \times 10^{-13}}$	$\mathbf{5.69 \times 10^{-14}}$	$\mathbf{7.18 \times 10^{-14}}$	F10	**0**	$\mathbf{1.96 \times 10^{-12}}$	$\mathbf{5.81 \times 10^{-13}}$	$\mathbf{7.43 \times 10^{-13}}$
BA		3.73×10^{-1}	4.72×10^{-1}	4.44×10^{-1}	2.52×10^{-2}		2.47×10^{3}	1.37×10^{4}	6.31×10^{3}	2.72×10^{3}
PSO		7.82×10^{-2}	3.12×10^{-1}	1.92×10^{-1}	5.56×10^{-2}		1.29×10^{2}	4.26×10^{2}	2.55×10^{2}	7.16×10^{1}
MFO		4.60×10^{-1}	4.76×10^{-1}	4.70×10^{-1}	3.45×10^{-3}		4.13×10^{2}	9.28×10^{2}	6.59×10^{2}	1.39×10^{2}
SCA		1.78×10^{-1}	3.47×10^{-1}	2.83×10^{-1}	4.28×10^{-2}		4.48×10^{2}	1.38×10^{3}	7.26×10^{2}	1.94×10^{2}
BOA		1.27×10^{-1}	1.54×10^{-1}	1.30×10^{-1}	5.35×10^{-3}		9.76×10^{-12}	1.43×10^{-11}	1.21×10^{-11}	1.06×10^{-12}

Table 4. Results of Wilcoxon rank-sum test for LAFBA and other algorithms on 10 test functions with D = 30.

F	LAFBA vs. BA		LAFBA vs. PSO		LAFBA vs. MFO		LAFBA vs. SCA		LAFBA vs. BOA	
	p_Value	h	p_Value	h	p_Value	h	p_Value	h	p_Value	h
F1	9.78×10^{-12}	1	9.78×10^{-12}	1	9.78×10^{-12}	1	9.78×10^{-12}	1	9.78×10^{-12}	1
F2	6.51×10^{-11}	1	6.50×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.48×10^{-11}	1
F3	3.71×10^{-11}	1	3.71×10^{-11}	1	3.71×10^{-11}	1	3.71×10^{-11}	1	0.111655	0
F4	1.24×10^{-11}	1	1.24×10^{-11}	1	1.24×10^{-11}	1	1.24×10^{-11}	1	1.14×10^{-06}	1
F5	2.23×10^{-11}	1	1.68×10^{-11}	1	9.12×10^{-12}	1	2.52×10^{-11}	1	2.55×10^{-11}	1
F6	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.45×10^{-11}	1
F7	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.46×10^{-11}	1
F8	6.50×10^{-11}	1	6.50×10^{-11}	1	6.50×10^{-11}	1	6.50×10^{-11}	1	6.46×10^{-11}	1
F9	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	0.043201	1
F10	6.50×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.51×10^{-11}	1	6.46×10^{-11}	1

3.4. Convergence Curve Analysis

The convergence curve is an important indicator for the performance of the algorithm, through which we can see the convergence speed and the ability of the algorithm to jump out of the local optimum. For further illustration, the convergence curves of the LAFBA and other 5 algorithms with D = 30 on 10 benchmark functions are plotted in Figure 2.

The different trends of the six curves in Figure 2 show the difference in the performance of the six algorithms. Functions 2, 6, 7, 9, and 10 are unimodal functions, and are often used to compare the convergence speed and execution ability of the algorithm. Observing the contrast curve of the corresponding function convergence curve, the LAFBA proposed in this paper is effective in obtaining the optimal solutions with a faster convergence rate. The quality of the solution is much higher than BA, PSO, MFO, SCA, and BOA. Functions 1, 3, 4, 5, and 8 are multimodal functions, with a large number of local optima, which are extremely difficult to optimize. They are commonly used to test the global optimization ability of the algorithm. Observing the convergence curves of the corresponding functions, the inflection point in the curve shows that the LAFBA algorithm successfully jumps out of the local optimum and continues to optimize, while the other algorithms converge to the local optimum too early, resulting in a higher curve than the LAFBA. In summary, the LAFBA shows stronger optimization performance and higher optimization efficiency.

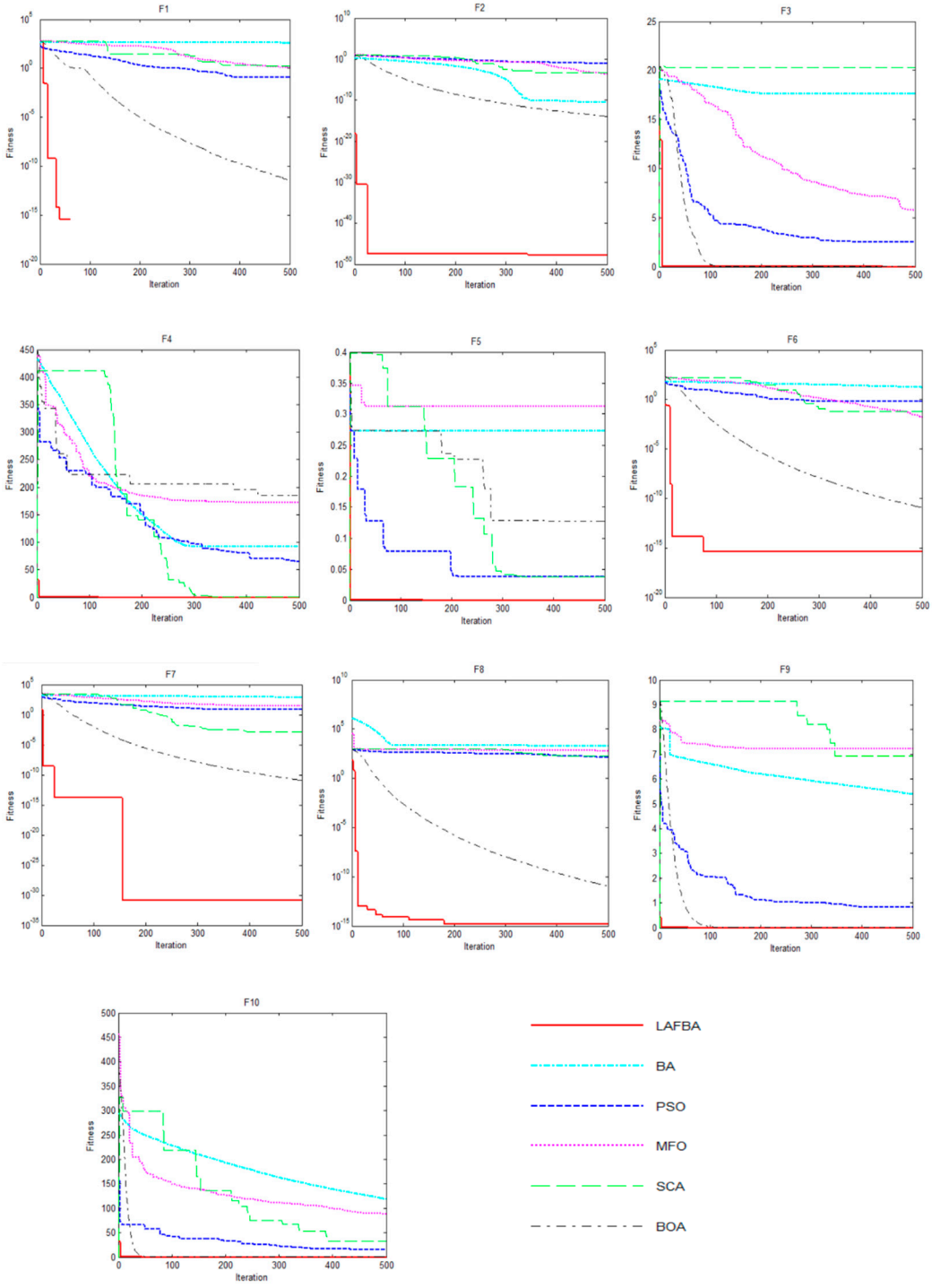

Figure 2. Convergence curve of F1–F10 function.

4. LAFBA for Classical Engineering Problems

This section further verifies the performance and efficiency of the LAFBA by solving two constrained real engineering design problems: tension/compression spring design, and welded beam design. These problems were widely discussed in the literature and have been solved to better clarify the effectiveness of the algorithms. In the LAFBA, the population size n = 20, and the number of iterations N_gen = 500.

4.1. Tension/Compression Spring Design

The objective of this test problem is to minimize the weight of the tension/compression spring. Figure 3 shows the spring and its parameters [39,40]. The optimum design must satisfy constraints on shear stress, surge frequency, and deflection. This problem contains three constraint variables: the mean coil diameter (D), number of active coils (N), and wire diameter (d). The mathematical expression of tension/compression spring design problem is as follows:

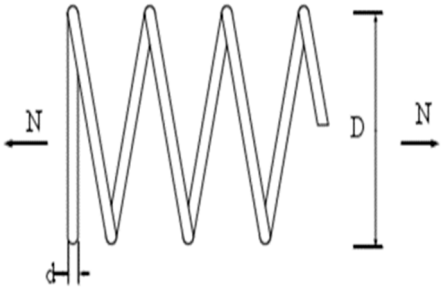

Figure 3. Tension/compression spring design problem.

Consider $\vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N]$,

Minimize $f(\vec{x}) = (x_3 + 2)x_2x_1^2$,

Subject to $g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0$,

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

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

$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$,

Variable range $0.05 \leq x_1 \leq 2.00$, $0.25 \leq x_2 \leq 1.30$, $2.00 \leq x_3 \leq 15.0$.

There are several solutions for this problem found in the literature. This test case was solved using either mathematical techniques (constraints correction at constant cost [41] and penalty functions [40]) or meta-heuristics, such as GSA [42], PSO [43], evolution strategy (ES) [44], GA [45], and improved harmony search (HS) [46]. The best results of LAFBA are compared with 10 other optimization algorithms that were previously reported, as shown in Table 5.

From Table 5, compared with the GSA, PSO, ES, GA, and WOA, the LAFBA yielded better results for the tension/compression spring design problem. It can be seen that LAFBA outperforms all other algorithms except MFO, and the LAFBA algorithm is also the third-lowest-costing design.

Table 5. Comparison results for tension/compression spring design problem.

Algorithms	Optimal Values for Variables			Optimal Cost
	d	D	N	
GSA [42]	0.050276	0.323680	13.525410	0.0127022
PSO (Ha and Wang) [43]	0.051728	0.357644	11.244543	0.0126747
ES (Coello and Montes) [44]	0.051989	0.363965	10.890522	0.0126810
GA(Coello) [45]	0.051480	0.351661	11.632201	0.0127048
Improved HS (Mmahdavi et al.) [46]	0.051154	0.349871	12.076432	0.0126706
MFO [9]	0.051994	0.364109	10.868422	0.0126669
WOA [47]	0.051207	0.345215	12.004032	0.0126763
Montes and Coello [48]	0.051643	0.355360	11.397926	0.0126980
Constraint correction (Arora) [41]	0.050000	0.315900	14.250000	0.0128334
Mathematical optimization (Belegundu) [40]	0.053396	0.399180	9.1854000	0.0127303
LAFBA	0.051663	0.356074	11.333400	0.0126720

4.2. Welded Beam Design

The objective of this test problem is to minimize the fabrication costs of the welded beam design, and Figure 4 shows the welded beam and parameters involved in the design [45]. The optimum design must satisfy constraints on shear stress (τ) and bending stress in the beam (θ), buckling load (P_C), and end deflection of the beam (δ). This problem contains four constraint variables: thickness of weld (h), length of the clamped bar (l), height of the bar (t), and thickness of the bar (b). The mathematical expression of welded beam design problem is as follows:

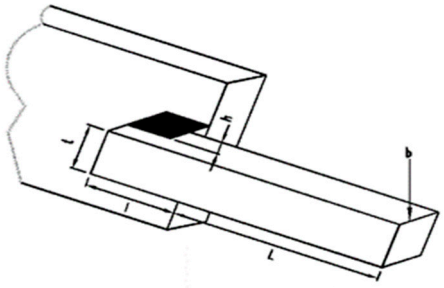

Figure 4. Schematic view of welded beam design problem.

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

Minimize $f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$,

Subject to
$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \le 0,$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \le 0,$$
$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \le 0,$$
$$g_4(\vec{x}) = x_1 - x_4 \le 0,$$
$$g_5(\vec{x}) = p - p_c(\vec{x}) \le 0,$$
$$g_6(\vec{x}) = 0.125 - x_1 \le 0,$$
$$g_7(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,$$

Variable range $0.1 \le x_1 \le 2, 0.1 \le x_2 \le 10, 0.1 \le x_3 \le 10, 0.1 \le x_4 \le 2,$

where
$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \quad \tau' = \frac{p}{\sqrt{2}x_1x_2}, \ \tau'' = \frac{MR}{J},$$
$$M = p\left(L + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2},$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, \quad \sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \ \delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4}$$
$$P_C(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$
$$p = 6000 \text{ lb}, \ L = 14 \text{ in.}, \delta_{max} = 0.25 \text{ in.},$$
$$E = 30 \times 10^6 \text{ psi}, G = 12 \times 10^6 \text{ psi},$$
$$\tau_{max} = 13,600 \text{ psi}, \sigma_{max} = 30,000 \text{ psi}.$$

This problem was solved by GWO [49], GSA [42], Richardson's random method, simplex method, Davidon–Fletcher–Powell, and Griffith and Stewart's successive linear approximation [45]. The optimization results obtained by the proposed LAFBA for this problem were evaluated by comparing it with 15 other optimization algorithms that were previously reported, as shown in Table 6. Table 6 shows the best obtained results.

Table 6 shows that the LAFBA algorithm is able to find a similar optimal design compared to those of GWO, MVO, and CPSO. This shows that this algorithm is also able to provide very competitive results in solving this problem.

Table 6. Comparison results for welded beam design problem.

Algorithms	Optimal Values for Variables				Optimal Cost
	h	l	t	b	
GWO [49]	0.205676	3.478377	9.03681	0.205778	1.72624
GSA [42]	0.182129	3.856979	10.0000	0.202376	1.87995
CPSO [50]	0.202369	3.544214	9.048210	0.205723	1.72802
GA(Coello) [51]	N/A	N/A	N/A	N/A	1.8245
GA(Deb) [52]	N/A	N/A	N/A	N/A	2.3800
GA(Deb) [53]	0.2489	6.1730	8.1789	0.2533	2.4331
HS (Lee and Geem) [54]	0.2442	6.2331	8.2915	0.2443	2.3807
MVO [55]	0.2054	3.47319	9.044502	0.20569	1.72645
GSA [56]	0.2057	3.4704	9.0366	0.2057	1.7248
MFO [9]	0.2057	3.4703	9.0364	0.2057	1.72452
WOA [47]	0.205396	3.484293	9.037426	0.206276	1.730499
Random [57]	0.4575	4.7313	5.0853	0.6600	4.1185
Simplex [57]	0.2792	5.6256	7.7512	0.2796	2.5307
David [57]	0.2434	6.2552	8.2915	0.2444	2.3841
Approx [57]	0.2444	6.2189	8.2915	0.2444	2.3815
LAFBA	0.184706185	3.642655691	9.134897358	0.205254053	1.7287

5. Conclusions

In this study, an improved bat algorithm based on Lévy flights and adjustment factors (LAFBA) is proposed. Three modifications have been embedded into the BA to increase its global and local search abilities and, consequently, have significantly enhanced the BA performance. In order to evaluate the effectiveness of the LAFBA, 10 benchmark functions and 2 real-world engineering problems are used. The results of the simulation experiment of 10 benchmark functions and Wilcoxon rank-sum test show that the proposed LAFBA was a great improvement in terms of exploration and exploitation abilities, solution accuracy, and convergence speed compared with the bat algorithm, and four other bio-inspired algorithms. In addition, the results of the LAFBA in solving classical engineering design problems were compared with several state-of-the-art algorithms and produced comparable results. As the LAFBA algorithm shows a stable performance, it can also be applied to other more challenging real-world optimization problems.

Author Contributions: Methodology, J.L.; Resources, X.R.; Writing—original draft, X.L.; Writing—review& editing, Y.L.

Acknowledgments: This study is supported by the National Natural Science Foundation of China (No. 71601071), the Science & Technology Program of Henan Province, China (No. 182102310886 and 162102110109), and an MOE Youth Foundation Project of Humanities and Social Sciences (No. 15YJC630079). We are particularly grateful to the suggestions of the editor and the anonymous reviewers which is greatly improved the quality of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Holland, J.H. Erratum: Genetic Algorithms and the Optimal Allocation of Trials. *Siam J. Comput.* **1974**, *3*, 326. [CrossRef]
2. Kennedy, J.; Eberhart, R. Particle swarm optimization. *IEEE Int. Conf. Neural Netw.* **1995**, *2002*, 1942–1948.
3. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*; IEEE Press: Piscataway, NJ, USA, 1995; pp. 39–43.

4. Dorigo, M.; Colorni, V.A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cyber. Part B Cyber.* **1996**, *26*, 29–41. [CrossRef] [PubMed]
5. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization. TR-06*; Erciyes University: Kayseri, Turkey, 2005.
6. Gandomi, A.H.; Alavi, A.H. Krill herd: A new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2012**, *17*, 4831–4845. [CrossRef]
7. Yang, X.S.; Deb, S. Cuckoo Search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
8. Yang, X.S.; Deb, S. Engineering Optimisation by Cuckoo Search. *Int. J. Math. Modell. Numer. Optim.* **2010**, *1*, 330–343. [CrossRef]
9. Mirjalili, S. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowl.-Based Syst.* **2015**, *89*, 228–249. [CrossRef]
10. Sergeyev, Y.D.; Kvasov, D.E.; Mukhametzhanov, M.S. Operational zones for comparing metaheuristic and deterministic one-dimensional global optimization algorithms. *Math. Comput. Simul.* **2017**, *141*, 96–109. [CrossRef]
11. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. *Comput. Knowl. Technol.* **2010**, *284*, 65–74.
12. Ramli, M.R.; Abas, Z.A.; Desa, M.I.; Abidin, Z.Z.; Alazzam, M.B. Enhanced Convergence of Bat Algorithm Based on Dimensional and Inertia Weight Factor. *J. King Saud Univ.-Comput. Inf. Sci.* **2018**. [CrossRef]
13. Banati, H.; Chaudhary, R. Multi-Modal Bat Algorithm with Improved Search (MMBAIS). *J. Comput. Sci.* **2017**, *23*, 130–144. [CrossRef]
14. Al-Betar, M.A.; Awadallah, M.A.; Faris, H.; Yang, X.S.; Khader, A.T.; Alomari, O.A. Bat-inspired Algorithms with Natural Selection mechanisms for Global optimization. *Neurocomputing* **2018**, *273*, 448–465. [CrossRef]
15. Li, Y.; Pei, Y.H.; Liu, J.S. Bat optimization algorithm combining uniform variation and Gaussian variation. *Control Decis.* **2017**, *32*, 1775–1781.
16. Chakri, A.; Khelif, R.; Benouaret, M.; Yang, X.S. New directional bat algorithm for continuous optimization problems. *Expert Syst. Appl.* **2017**, *69*, 159–175. [CrossRef]
17. Al-Betar, M.A.; Awadallah, M.A. Island Bat Algorithm for Optimization. *Expert Syst. Appl.* **2018**, *107*, 126–145. [CrossRef]
18. Laudis, L.L.; Shyam, S.; Jemila, C.; Suresh, V. MOBA: Multi Objective Bat Algorithm for Combinatorial Optimization in VLSI. *Proc. Comput. Sci.* **2018**, *125*, 840–846. [CrossRef]
19. Tawhid, M.A.; Dsouza, K.B. Hybrid Binary Bat Enhanced Particle Swarm Optimization Algorithm for solving feature selection problems. *Appl. Comput. Inf.* **2018**. [CrossRef]
20. Osaba, E.; Yang, X.S.; Diaz, F.; Lopez-Garcia, P.; Carballedo, R. An improved discrete bat algorithm for symmetric and asymmetric Traveling Salesman Problems. *Eng. Appl. Artif. Intell.* **2016**, *48*, 59–71. [CrossRef]
21. Mohamed, T.M.; Moftah, H.M. Simultaneous Ranking and Selection of Keystroke Dynamics Features Through A Novel Multi-Objective Binary Bat Algorithm. *Future Comput. Inf. J.* **2018**, *3*, 29–40. [CrossRef]
22. Hamidzadeh, J.; Sadeghi, R.; Namaei, N. Weighted Support Vector Data Description based on Chaotic Bat Algorithm. *Appl. Soft Comput.* **2017**, *60*, 540–551. [CrossRef]
23. Qi, Y.H.; Cai, Y.G.; Cai, H. Discrete Bat Algorithm for Vehicle Routing Problem with Time Window. *Chin. J. Electron.* **2018**, *46*, 672–679.
24. Bekdaş, G.; Nigdeli, S.M.; Yang, X.S. A novel bat algorithm based optimum tuning of mass dampers for improving the seismic safety of structures. *Eng. Struct.* **2018**, *159*, 89–98. [CrossRef]
25. Ameur, M.S.B.; Sakly, A. FPGA based hardware implementation of Bat Algorithm. *Appl. Soft Comput.* **2017**, *58*, 378–387. [CrossRef]
26. Chaib, L.; Choucha, A.; Arif, S. Optimal design and tuning of novel fractional order PID power system stabilizer using a new metaheuristic Bat algorithm. *Ain Shams Eng. J.* **2017**, *8*, 113–125. [CrossRef]
27. Mohammad, E.; Sayed-Farhad, M.; Hojat, K. Bat algorithm for dam–reservoir operation. *Environ. Earth Sci.* **2018**, *77*, 510.
28. Liu, J.S.; Ji, H.Y.; Li, Y. Robot Path Planning Based on Improved Bat Algorithm and Cubic Spline Interpolation. *Acta Autom. Sin.* **2019**.
29. Shi, Y.; Eberhart, R. Modified particle swarm optimizer. *Proc. IEEE ICEC Conf. Anchorage* **1999**, 69–73.
30. Du, Y.H. *Advanced Mathematics*; Beijing Jiaotong University Press: Beijing, China, 2014.
31. Ball, F.; Bao, Y.N. *Predict Society*; Contemporary China Publishing House: Beijing, China, 2007.

32. Yang, X.S.; Karamanoglu, M.; He, X. Flower pollination algorithm: A novel approach for multiobjective optimization. *Eng. Optim.* **2014**, *46*, 1222–1237. [CrossRef]
33. Jamil, M.; Yang, X.S. A Literature Survey of Benchmark Functions for Global Optimization Problems. *Mathematics* **2013**, *4*, 150–194.
34. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133. [CrossRef]
35. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
36. Derrac, J.; García, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evolut. Comput.* **2011**, *1*, 3–18. [CrossRef]
37. Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biom. Bull.* **1945**, *1*, 80–83. [CrossRef]
38. García, S.; Molina, D.; Lozano, M.; Herrera, F. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 Special Session on Real Parameter Optimization. *J. Heuristics* **2009**, *15*, 617–644.
39. Zhao, Z.Y. Introduction to optimum design. *Probabilistic Eng. Mech.* **1990**, *5*, 100.
40. Belegundu, A.D.; Arora, J.S. A study of mathematical programming methods for structural optimization. Part I: Theory. *Int. J. Numer. Methods Eng.* **2010**, *21*, 1601–1623. [CrossRef]
41. Kaveh, A.; Talatahari, S. An improved ant colony optimization for constrained engineering design problems. *Eng. Comput.* **2010**, *27*, 155–182. [CrossRef]
42. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
43. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [CrossRef]
44. Mezura-Montes, E.; Coello, C.A.C. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int. J. Gen. Syst.* **2008**, *37*, 443–473.
45. Coello Coello, C.A. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Comput. Ind.* **2000**, *41*, 113–127. [CrossRef]
46. Mahdavi, M.; Fesanghary, M.; Damangir, E. An improved harmony search algorithm for solving optimization problems. *Appl. Math. Comput.* **2007**, *188*, 1567–1579. [CrossRef]
47. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
48. Li, L.J.; Huang, Z.B.; Liu, F.; Wu, Q.H. A heuristic particle swarm optimizer for optimization of pin connected structures. *Comput. Struct.* **2007**, *85*, 340–349. [CrossRef]
49. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
50. Krohling, R.A.; Coelho, L.D.S. Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems. *IEEE Trans. Cyber.* **2007**, *36*, 1407–1416. [CrossRef]
51. Coello, C.; Carlos, A. constraint-handling using an evolutionary multi objective optimization technique. *Civ. Eng. Environ. Syst.* **2000**, *17*, 319–346. [CrossRef]
52. Deb, K. Optimal design of a welded beam via genetic algorithms. *AIAA J.* **1991**, *29*, 2013–2015.
53. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
54. Lee, K.S.; Geem, Z.W. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Comput. Methods Appl. Mech. Eng.* **2005**, *194*, 3902–3933. [CrossRef]
55. MartÍ, V.; Robledo, L.M. Multi-Verse Optimizer: A nature-inspired algorithm for global optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513.
56. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]
57. Ragsdell, K.M.; Phillips, D.T. Optimal Design of a Class of Welded Structures Using Geometric Programming. *J. Eng. Ind.* **1976**, *98*, 1021–1025. [CrossRef]