

Article

A Novel Bat Algorithm with Multiple Strategies Coupling for Numerical Optimization

Yechuang Wang ¹ , Penghong Wang ¹, Jiangjiang Zhang ¹, Zhihua Cui ^{1,*}, Xingjuan Cai ¹, Wensheng Zhang ² and Jinjun Chen ³

¹ Complex System and Computational Intelligent Laboratory, Taiyuan University of Science and Technology, Taiyuan 030024, China; yechuangwang@sina.com (Y.W.); penghongwang@sina.cn (P.W.); jiangjofyouth@163.com (J.Z.); xingjuancai@gmail.com (X.C.)

² State Key Laboratory of Intelligent Control and Management of Complex Systems, Institute of Automation Chinese Academy of Sciences, Beijing 100190, China; wensheng.zhang@ia.ac.cn

³ Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne 3000, Australia; jinjun.chen@gmail.com

* Correspondence: zhihua.cui@hotmail.com; Tel.: +86-138-3459-9274

Received: 10 December 2018; Accepted: 21 January 2019; Published: 1 February 2019



Abstract: A bat algorithm (BA) is a heuristic algorithm that operates by imitating the echolocation behavior of bats to perform global optimization. The BA is widely used in various optimization problems because of its excellent performance. In the bat algorithm, the global search capability is determined by the parameter loudness and frequency. However, experiments show that each operator in the algorithm can only improve the performance of the algorithm at a certain time. In this paper, a novel bat algorithm with multiple strategies coupling (mixBA) is proposed to solve this problem. To prove the effectiveness of the algorithm, we compared it with CEC2013 benchmarks test suits. Furthermore, the Wilcoxon and Friedman tests were conducted to distinguish the differences between it and other algorithms. The results prove that the proposed algorithm is significantly superior to others on the majority of benchmark functions.

Keywords: bat algorithm (BA); bat algorithm with multiple strategy coupling (mixBA); CEC2013 benchmarks; Wilcoxon test; Friedman test

1. Introduction

In the past ten years, many heuristic optimization algorithms, such as particle swarm optimization (PSO) [1–3], ant colony optimization (ACO) [4,5], bat algorithm (BA) with triangle-flipping strategy [6], fly algorithm (FA) [7–9], cuckoo search [10–13], pigeon-inspired optimization algorithm, and genetic algorithm (GA) [14], have been developed to solve complex computational problems. It became popular because of its superior ability, which deals with a variety of complex issues. Moreover, it has been proven that there is no heuristic algorithm that can perform generally enough to solve all optimization problems [15]. Therefore, scholars have tried to solve these problems with different bionic algorithms.

BA [16–19] is a novel heuristic optimization algorithm, inspired by the echolocation behavior of bats. This algorithm carries out the search process using artificial bats as search agents mimicking the natural pulse loudness and emission rate of real bats. To improve the performance of BA, different strategies have been proposed. We will elaborate in the following three research situations.

(I) Parameter adjustment

For the standard BA algorithm, four main parameters are required: frequency, emission, constants, and emission rate. The frequency is used to balance the impact of the historical optimal position on the

current position. The bat individual will search far from the group historical position when the search range of frequency is large, and vice versa. In general, the choice of frequency range is determined by different issues. Hasançebi [20] set the pulse frequency range to [0–1]. Gandomi and Yang [21] sets the frequency range to [0–2] in the chaotic bat algorithm. Fister et al. [22] sets the frequency to [0–5] in their algorithm. Ali [23] sets the frequency to [0–100] in the power system. Xie et al. [24] proposed an adaptive adjustment strategy for frequency. Pérez et al. [25] designed a fuzzy controller to dynamically adjust the range of pulse frequencies, while Liu [26] replaced the frequency with a Lévy distribution. To improve the local search capability, Yilmaz and Kucuksille [27] added a random item with two randomly selected bats to explore more search space. Cai [28] introduced a linear decreasing function into the bat algorithm to enhance the global search capability.

(II) Formula adjustment

In terms of global search, the step size of the standard BA algorithm decreases with the increase of iterations, which causes the algorithm to be sensitive to local optimum. Focusing on this problem, Bahmani-Firouzi and Azizipناه-Abarghoee [29] proposed four different velocity updating strategies to keep a balance between exploitation and exploration. Inspired by PSO, Yilmaz and Kucuksille [30] put the inertia weight into the velocity update equation. Xie et al [24] use random parts associated with Lévy distributions instead of avoidance. To improve the local search capability, four differential evolutionary strategies were employed to replace the original local search pattern in the standard BA [31]. Xie et al. [32] also incorporated the Lévy flight in the velocity update equation, but four randomly selected bats were used to guide the search pattern. Zhu et al. [33] replace the swarm historical best position with the mean best position to enhance the convergence speed.

(III) Application

BA has been widely applied to various areas, including classification, wireless sensor [34], and data mining. Yang and Gandomi [35] proposed a bat algorithm to solve multi-objective problems; Bora et al. [36] proposed a bat-inspired optimization approach to solve the brushless direct current (DC) wheel motor problem; Sambariya and Prasad [37] proposed a metaheuristic bat algorithm for solving robust turning of power system stabilizer for small signal stability enhancement; Sathya and Ansari [38] highlighted the load frequency control using dual mode bat algorithm based scheduling of PI controllers for interconnected power systems; Sun and Xu [39] proposed node localization of wireless sensor networks based on a hybrid bat-quasi-newton algorithm; Cao et al. [40] improved low energy adaptive clustering hierarchy protocol based on a local centroid bat algorithm.

Furthermore, there are many applications in big data and machine learning [41,42], such as Hamidzadeh et al. [43], who proposed a novel method called chaotic bat algorithm for support vector data description (SVDD) (CBA-SVDD) to design effective descriptions of data. Alslibi [44] proposed a novel membrane-inspired binary bat algorithm for facial feature selection. Furthermore, it outperforms recent state-of-the-art face recognition methods on three benchmark databases. Therefore, the bat algorithm has a wide range of applications. In addition, the bat algorithm has been proposed to optimize support vector machine (SVM) parameters that reduce the classification error [45]. Notably, increasing SVM prediction accuracy and avoiding local optimal trap using the bat algorithm has been very helpful in biomedical research [46,47].

The rest of this paper is organized as follows. Section 2 provides a brief description of the standard BA. In Section 3, we listed eight improvement strategies and proposed a novel bat algorithm with multiple strategy coupling. Numerical experiments on the CEC2013 benchmark set are conducted in Section 4. Finally, the discussion and future work are given in Section 5.

2. Bat Algorithm

The bat algorithm [48] was proposed by Xin-She Yang, based on the echolocation of microbats. Bats usually use echolocation to find food. During removal, bats usually send out short pulses,

however, when they encounter food, their pulse send out rates increase and the frequency goes up. The increase in frequency means frequency-tuning, which shortens the echolocations' time and increases the location accuracy. In the standard bat algorithm, each individual i has a defined position $x_i(t)$ and velocity $v_i(t)$ in the search space, which will be updated as the number of iterations increases. The new positions $x_i(t)$ and velocities $v_i(t)$ can be calculated as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (1)$$

$$v_i(t+1) = v_i(t) + (x_i(t) - p(t)) \cdot f_i \quad (2)$$

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \cdot \beta \quad (3)$$

where β is a random vector with uniform distribution, the range of which is $[0, 1]$. $p(t)$ is the current global optimal solution and $f_{\min} = 0$, $f_{\max} = 1$.

As we also know, whether BA has global and local search capabilities depends on its parameters; therefore, it is necessary to achieve a balance between global search and local search capabilities by adopting adaptive parameters. The formula for the local search strategy is as follows:

$$x_i(t+1) = \vec{p}(t) + \varepsilon \bar{A}(t) \quad (4)$$

where ε is a random number from $[-1, 1]$, $\bar{A}(t)$ is the average loudness of population.

In addition, it achieves global search by controlling loudness $A_i(t+1)$ and pulse rate $r_i(t+1)$.

$$A_i(t+1) = \alpha A_i(t) \quad (5)$$

$$r_i(t+1) = r_i(0)[1 - \exp(-\gamma t)] \quad (6)$$

where α and γ are constants and $\alpha > 0$, $\gamma > 0$. $A_i(0)$ and $r_i(0)$ are initial values of loudness and pulse rate, respectively.

The following describes the execution steps of the standard bat algorithm.

- Step 1:** For each bat, initialize the position, velocity, and parameters and randomly generate the frequency with Equation (3).
- Step 2:** Update the position and velocity of each bat with Equations (1) and (2).
- Step 3:** For each bat, generate a random number ($0 < rand1 < 1$). Update the temp position and calculate the fitness value for corresponding bat with Equation (4) if $rand1 < r_i(t)$.
- Step 4:** For each bat, generate a random number ($0 < rand2 < 1$). Update $A_i(t)$ and $r_i(t)$ with Equations (5) and (6), respectively, if $rand2 < A_i(t)$ and $f(x_i(t)) < f(p(t))$.
- Step 5:** Sort each individual based on fitness values and save the best position.
- Step 6:** The algorithm is finished if the condition is met, otherwise, move on to Step 2.

Detailed steps about the standard bat algorithm are presented in Figure 1.

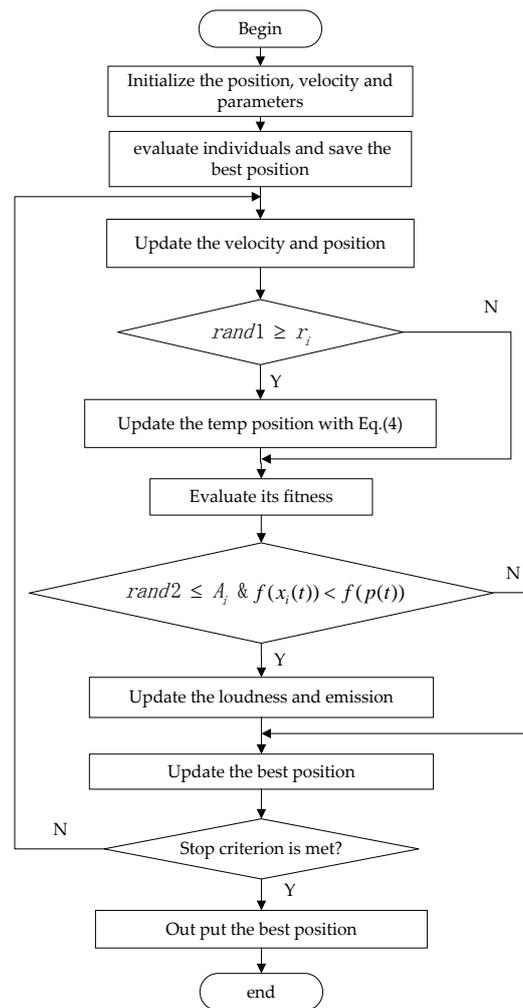


Figure 1. The flowchart of the standard bat algorithm (BA).

3. Bat Algorithm with Multiple Strategy Coupling

Through a large number of experimental studies, we found that different operators play an important role in the convergence ability of the algorithm. When the development operator increases, the global convergence ability of the algorithm becomes weaker; when the exploration operator increases, the convergence accuracy will be insufficient. Therefore, in this paper, we propose a multiple strategy autonomous selection strategy. The main idea is that different individuals choose which strategy to update the position according to the quality of fitness. In this paper, the bat algorithm with multiple strategy coupling (mixBA) formed will adopt the following eight strategies.

- (1) The velocity and position formula of the original algorithm are adopted [42]:

$$x_{ik}(t + 1) = x_{ik}(t) + v_{ik}(t + 1) \tag{7}$$

$$v_{ik}(t + 1) = v_{ik}(t) + (x_{ik}(t) - p_k(t)) \cdot f_i \tag{8}$$

- (2) The velocity and position formula of the improved algorithm are adopted [29]:

$$x_{ik}(t + 1) = x_{ik}(t) + v_{ik}(t + 1) \tag{9}$$

$$v_{ik}(t + 1) = v_{ik}(t) + (x_{ik}(t) - w_k(t)) \cdot f_i \tag{10}$$

where w_k is the position of the worst individual.

- (3) The position and velocity formula of Levy flight was adopted by Xie et al. [24]:

$$f_i^t = ((f_{\max} - f_{\min}) \frac{t}{n_t} + f_{\min}) \beta \tag{11}$$

$$v_i^t = (\hat{x}_i - x^*) f_i^t \tag{12}$$

$$x_i^t = \hat{x}_i + \text{usign}[\text{rand}(1) - \frac{1}{2}] \oplus \text{Levy}(\lambda) \tag{13}$$

where β is a random constant, n_t is a constant, \hat{x}_i is the best position of the i th bat, and x^* is the best position of those found so far. $\text{Levy}(\lambda)$ ($1 < \lambda \leq 3$) is step length of the Levy flight and \oplus stands for dot product.

- (4) The position and velocity formula of Levy flight was adopted by Liu [26]:

$$x_i^{t+1} = x_i^t + \text{Levy}(\lambda) \otimes (x_i^t - x^*) \tag{14}$$

x_i^t and x_i^{t+1} are i th the position of t bat in generation and $t + 1$ generation, respectively.

- (5) The position and velocity formula with the idea of genetic algorithms was adopted:

$$x_i(t + 1) = Dx_i(t) + (1 - D)x_i^*(t) \tag{15}$$

The formula is a two-point crossover operator in simulation genetic algorithm.

- (6) The position and velocity formula with the idea of PSO was adopted:

$$v_{ik}(t + 1) = v_{ik}(t) + r_1(x_{ik}(t) - p_{gk}(t))_i + r_2(x_{ik}(t) - p_{ik}(t)) \tag{16}$$

$$x_{ik}(t + 1) = x_{ik}(t) + v_{ik}(t + 1) \tag{17}$$

where r_1 and r_2 are random constants and p_{gk} is the best position by the entire swarm.

- (7) A local disturbance strategy based on inertial parameters is adopted:

$$x_i(t + 1) = x_i^*(t) + wr \tag{18}$$

$$w = w_{\max} - (w_{\max} - w_{\min}) \cdot t / T_{\max} \tag{19}$$

where w_{\max} and w_{\min} are the maximum and minimum values, respectively, of inertia weight and T_{\max} is the maximum number of iterations.

- (8) The local search strategy of flight to optimal position is adopted.

$$x_i(t + 1) = x_i^*(t) + r \cdot (p_{gk}(t) - x_{ik}(t)) \tag{20}$$

where p_{gk} is the best position by the entire swarm and r is a random constant.

The above strategies are chosen by the form of probability. Therefore, the number of bat individuals of choosing different strategies varies from generation to generation. Each strategy adjusts the probability of it being selected according to evaluation results. When the fitness value is better, the probability of the strategy will be adjusted by Equation (21).

$$P(n + 1) = P(n) + (1 - \lambda) \cdot P(n) \tag{21}$$

Otherwise, it is calculated as follows:

$$P(n + 1) = \lambda \cdot P(n) \tag{22}$$

where $\lambda = 0.75$, which is a parameter used to adjust the rate of probability change. The larger λ is, the slower the rate of probability reduction will be. To ensure the diversity of the population, we define the probability lower bound for each strategy as 0.01. The procedure of the bat algorithm with multiple strategy coupling is shown in Table 1.

Table 1. The procedure of the bat algorithm (BA) with multiple strategy coupling.

Algorithm 1: Bat algorithm with multiple strategy coupling

```

Begin
  For each bat, initialize the position, velocity, parameters and probability table;
  While (stop criterion is met)
    Randomly generate the frequency for each bat with Equation 3;
    Evaluate its fitness;
  Switch num = 8
    Case 1 ( $rand < p1$ )
      Update the velocity and position with strategy1.
    Case 2 ( $p1 < rand < p2$ )
      Update the velocity and position with strategy2
    Case 3 ( $p2 < rand < p3$ )
      Update the velocity and position with strategy3.
    Case 4 ( $p3 < rand < p4$ )
      Update the velocity and position with strategy4.
    Case 5 ( $p4 < rand < p5$ )
      Update the velocity and position with strategy5.
    Case 6 ( $p5 < rand < p6$ )
      Update the velocity and position with strategy6.
    Case 7 ( $p6 < rand < p7$ )
      Update the velocity and position with strategy7.
    Case 8 ( $p7 < rand < p8$ )
      Update the velocity and position with strategy8.
  Evaluate its fitness;
  If the position is update
    Update the loudness and emission rate;
    Update the probability table;
    If  $pi < 0$ 
       $Pi = 0.001$ ;
    End
  End
  Rank the bats and save the best position;
End
Output the best position;
End

```

4. Experimental Result

4.1. Text Functions and Parameter

The algorithm is tested on the CEC2013 benchmark set [43]. The test set can be divided into three groups, as shown in Table 2.

Table 2. The CEC2013 benchmark set.

F1–F5 belongs to uni-modal functions
F6–F20 belongs to multi-modal functions
F21–F28 belongs to composition functions

The experiment is tested on Matlab 2016a environment (2016a, MathWorks, Natick, MA, USA). For details of parameter settings for the bat algorithm with multiple strategies coupling (mixBA), please

refer to Table 3. It is worth emphasizing that the parameters of the adopted strategy are not optimized in this paper. In our algorithm, we used the following indicators to evaluate the experimental results.

$$meanerror = \left| \frac{\sum_{j=1}^{51} f_j}{51} - f_{true} \right| \tag{23}$$

where f_{true} is the actual solution set of the test set.

Table 3. The CEC2013 benchmark set.

Pop size	100
Run	51
Frequency	[0, 5]
$A(0)$	0.95
$r(0)$	0.9
α	0.99
γ	0.9
Search Domain	$[-100, 100]^D$

4.2. Comparison of MixBA with State-of-the-Art Algorithms

In this section, we will compare mixBA with six other algorithms. The algorithms involved are presented in Table 4.

Table 4. The involved algorithm.

Bat algorithm with multiple strategy coupling (mixBA); Standard bat algorithm (SBA); Self-adaptive heterogeneous particle swarm optimization (PSO) [49]; Bat algorithm with Lévy distribution (LBA1) [26]; Bat algorithm with Lévy distribution (LBA2) [32]; Bat algorithm with arithmetic centroid strategy (ACBA) [40] Oriented cuckoo search (OCS) [34]

The experimental results will be compared with the standard bat algorithm (SBA), PSO, bat algorithm with Lévy distribution LBA1, LBA2, bat algorithm with arithmetic centroid strategy (ACBA), and oriented cuckoo search (OCS) algorithms. Table A1 (in Appendix A) shows the average error obtained by different algorithms in different test functions. In the last line of the table, w refers to the number of mixBA algorithms superior to other algorithms in the test function, t indicates the number of performances similar to other algorithms, and L refers to the number of mixBA algorithms inferior to other algorithms. The dynamic comparison can be viewed in Figure 2.

From Table A1, mixBA won 27 functions, 28 functions, and 26 functions compared with SBA, LBA1, and LBA2, respectively. Compared with the mixBA algorithm, the PSO and OCS algorithm has seven functions and six functions, respectively, that are good.

For most of the test functions, the SBA does not find the global optimal solution. For ACBA and PSO, it only finds optimal solutions on function F1 and F5. LBA1 and LBA2 showed excellent searching ability on functions F11, F14, and F17. OCS obtains good solutions on functions F1, F4, and F5. MixBA can find reasonable solutions on the most of the functions.

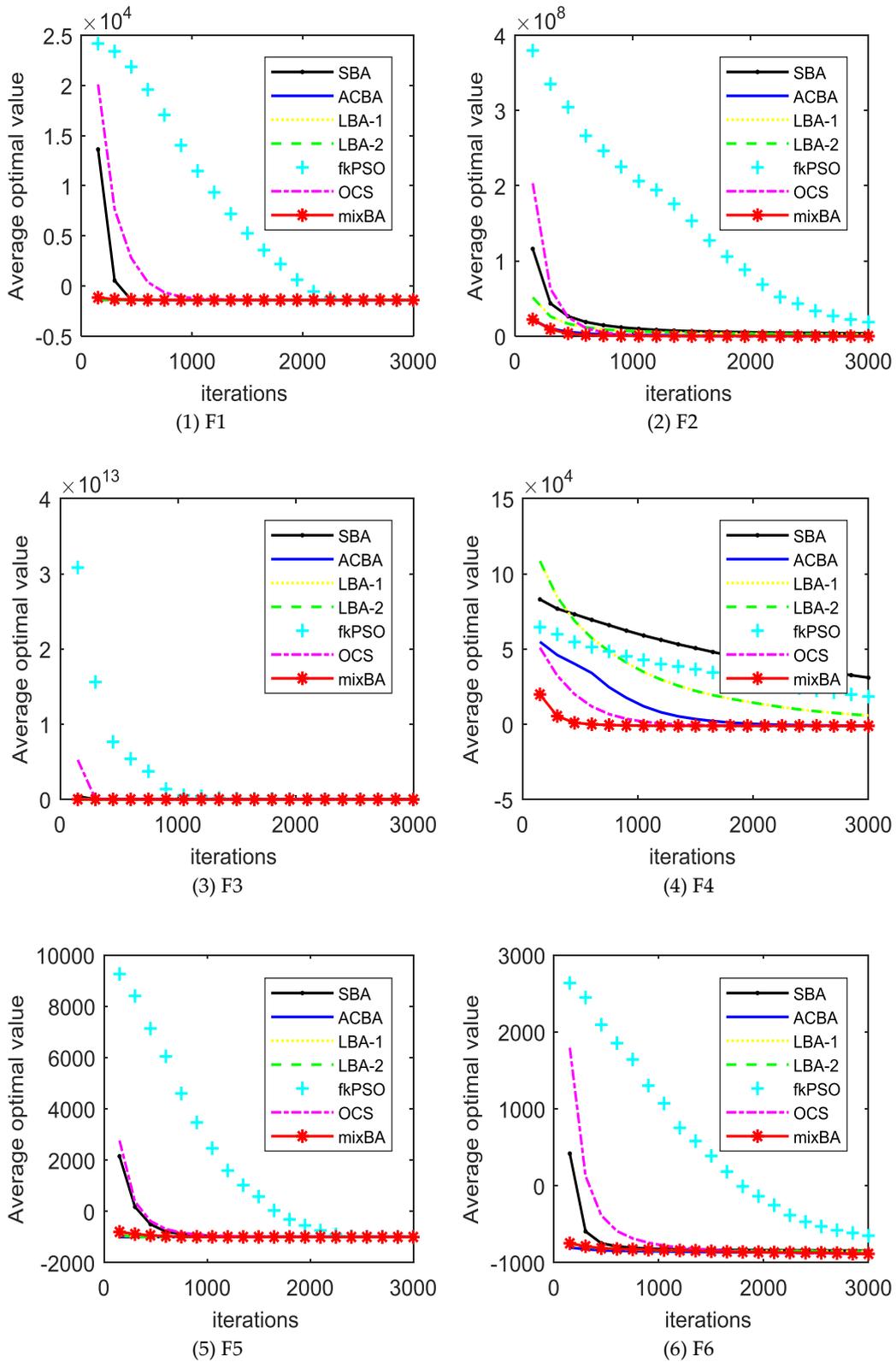


Figure 2. Cont.

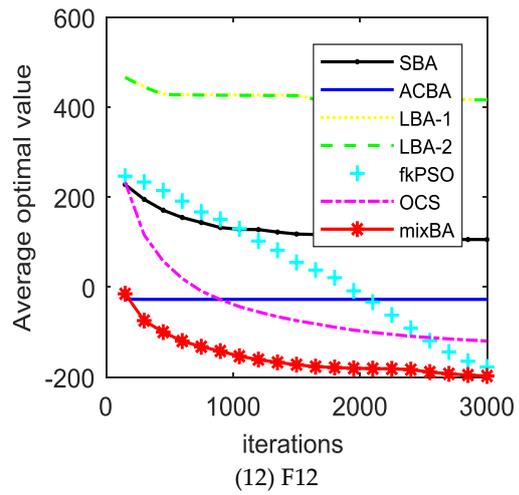
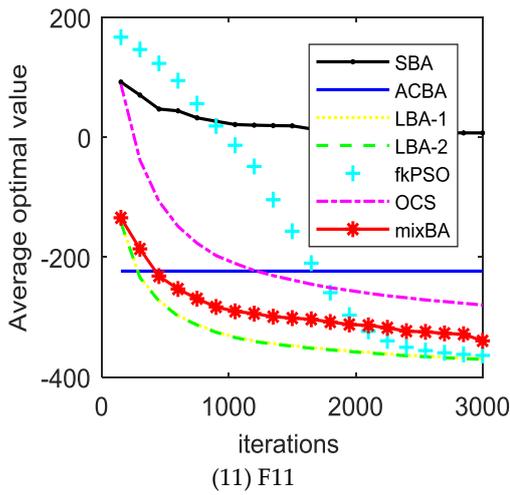
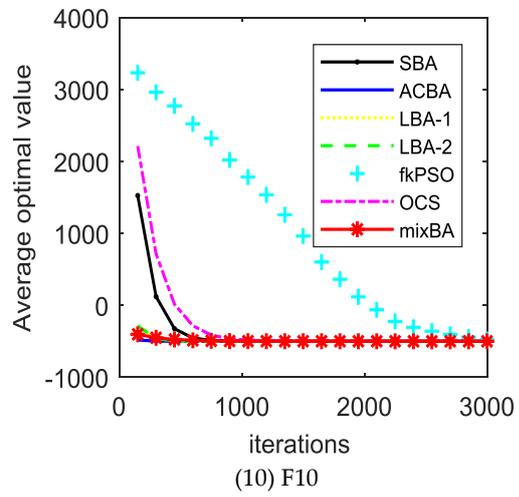
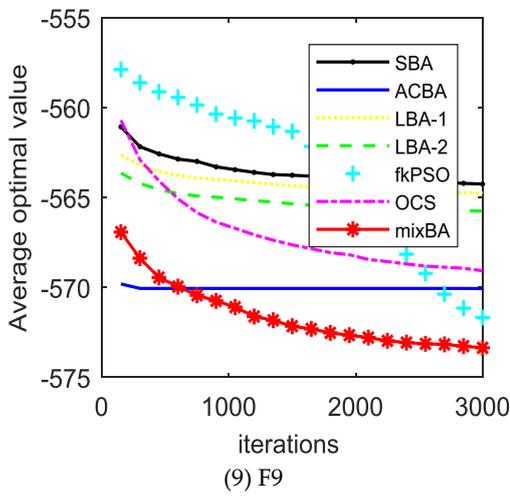
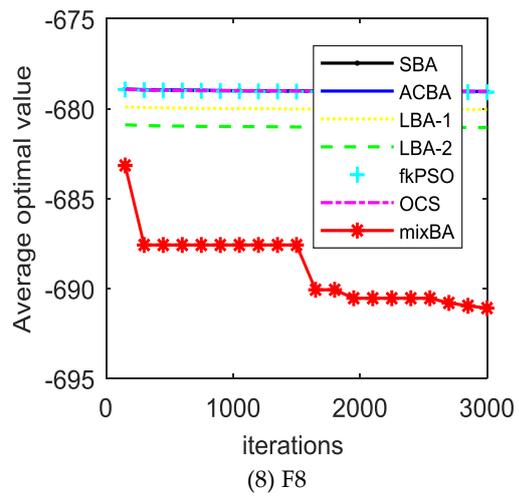
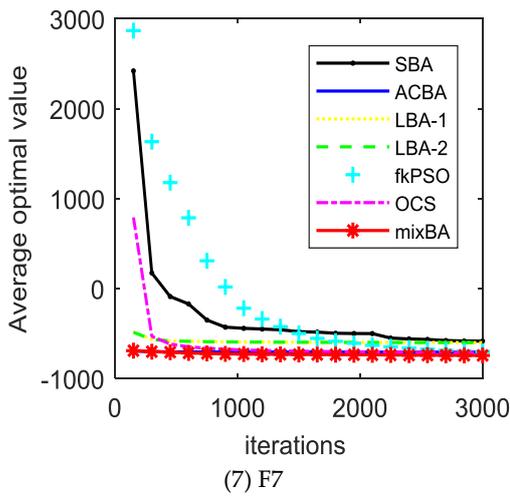


Figure 2. Cont.

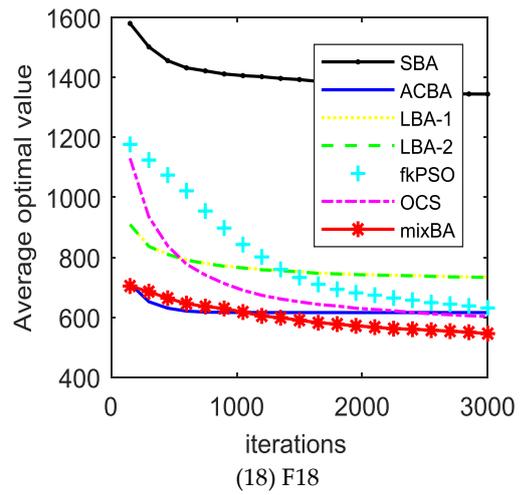
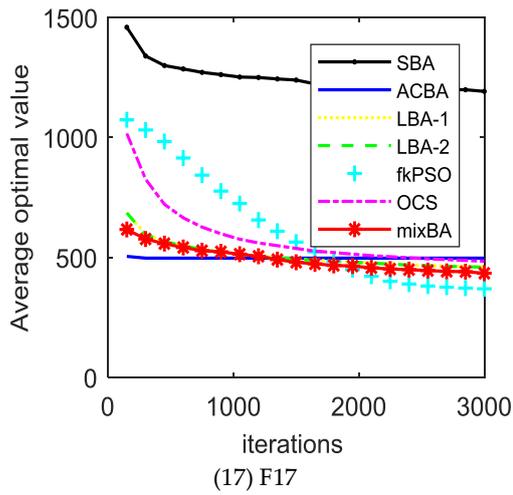
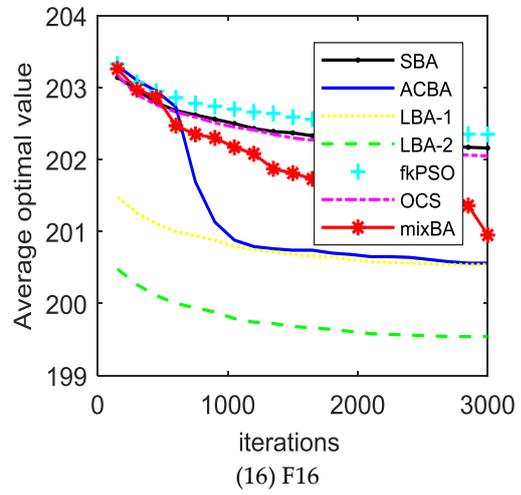
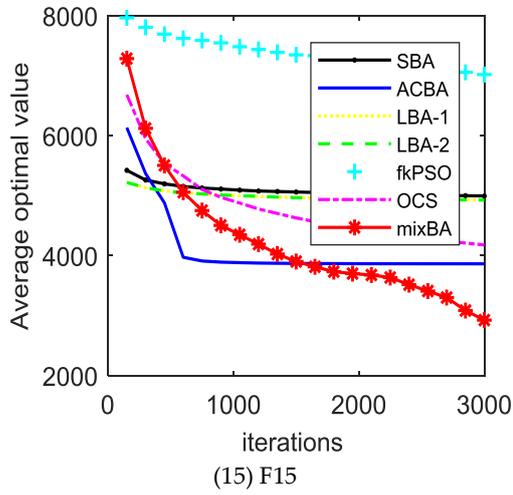
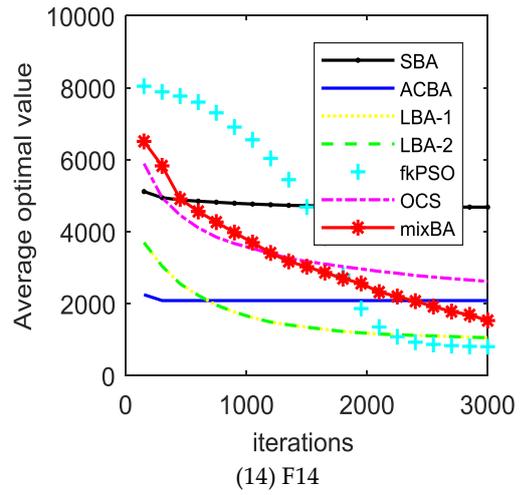
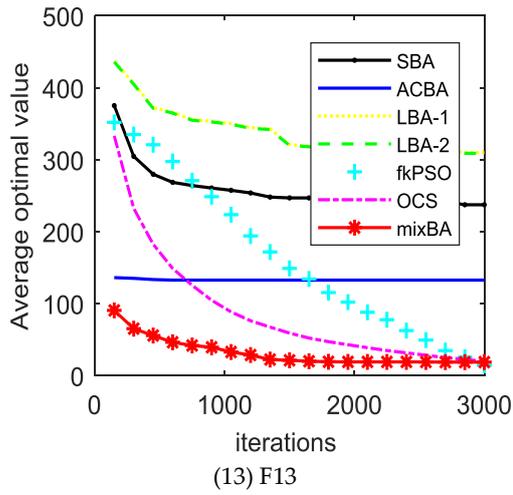


Figure 2. Cont.

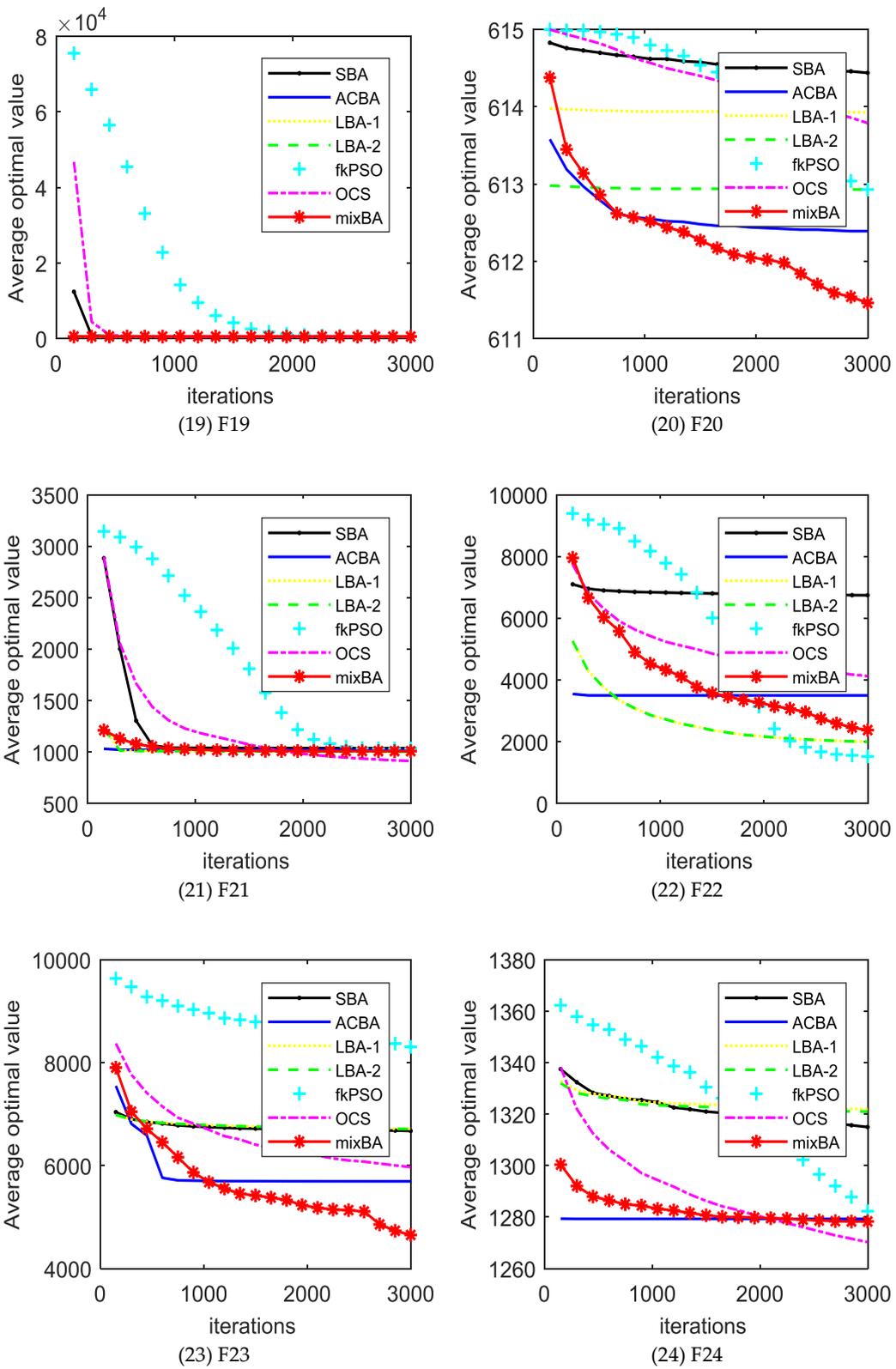


Figure 2. Cont.

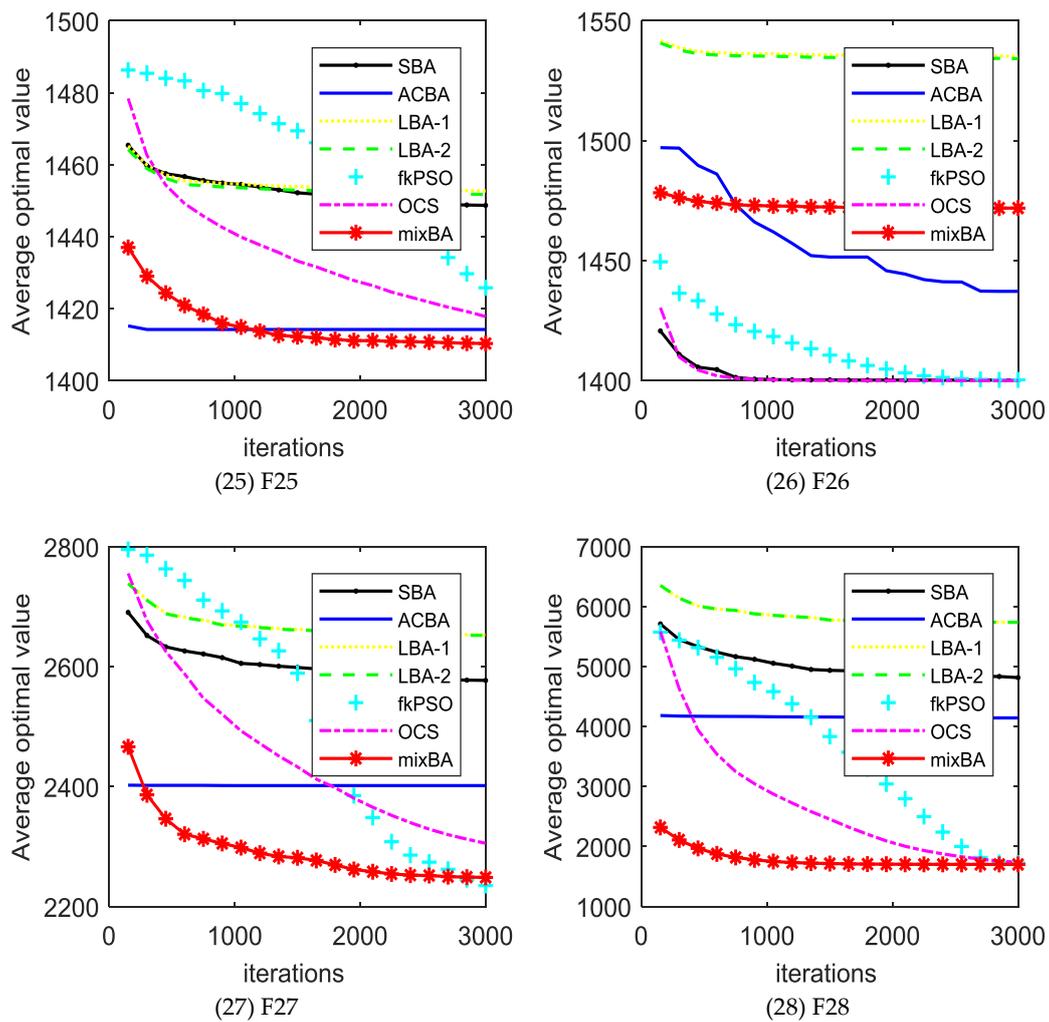


Figure 2. The convergence curves of different algorithms on the benchmark set. SBA—standard bat algorithm; ACBA—bat algorithm with arithmetic centroid strategy; LBA—bat algorithm with Lévy distribution; PSO—particle swarm optimization; OCS—oriented cuckoo search.

Table 5 presents the statistical results obtained by Friedman tests [30,50]. The smaller the ranking value, the better the performance of the algorithm. From the results, we can get the ranks of seven algorithms as follows: SBA, LBA1, LBA2, ACBA, PSO, OCS, mixBA. The highest ranking shows that mixBA is the best algorithm among the seven algorithms.

Table 5. Friedman test for the seven algorithms. SBA—standard bat algorithm; ACBA—bat algorithm with arithmetic centroid strategy; LBA—bat algorithm with Lévy distribution; PSO—particle swarm optimization; OCS—oriented cuckoo search.

Algorithm	Rankings
SBA	5.86
ACBA	3.59
LBA1	5.61
LBA2	5.05
FK-PSO	3.09
OCS	2.86
mixBA	1.95

Table 6 shows the results of the Wilcoxon test [51]. For SBA, ACBA, LBA1, LBA2 and PSO algorithm, $p < 0.05$. The results of this experiment show that the performance of mixBA is far superior to that of other algorithms. For OCS, the p -value is approximately equal to the 0.05 significance level.

Table 6. Wilcoxon test for the seven algorithms.

mixBA vs	p -Value
SBA	0
ACBA	0
LBA1	0
LBA2	0
FK-PSO	0.04
OCS	0.068

Figure 2 shows the convergence of different algorithms in different test functions. In most cases, our proposed algorithm has better results. However, it is undeniable that in a few cases, our proposed algorithm shows poor performance compared with other algorithms, such as in F11, F14, F16, F17, F22, F24, and F26. This is because the magnitude of the probability adjustment of the strategy is large, which leads to the algorithm prematurely selecting a certain strategy to make the algorithm fall into the local optimal. In addition, Table A2 shows the runtime of each algorithm on the CEC2013 benchmark set. It is obvious that the running time of the mixBA algorithm is slightly higher than that of SBA on F1 and F2, but significantly smaller than other algorithms in other test functions.

5. Conclusions

Bio-inspired computation is a collection for stochastic optimization algorithms inspired by biological phenomenon. BA is novel bio-inspired algorithm inspired by bat behaviors, and has been used to solve engineering optimization problems. However, with a single optimization strategy, it shows weakness in solving various complex optimization problems. To tackle this issue, this paper proposes a bat algorithm with multiple strategy coupling (mixBA) to improve the performance of BA. The simulation results show that the performance of the mixBA is superior to that of other algorithms. This is because of the adoption of adaptive multi-strategies coupling rules. The algorithm can adjust the probabilities of different strategies based on the evaluation index. In most cases, this manner guarantees the global convergence and local exploration ability of the algorithm. However, in some cases, this treatment causes other strategies to be ignored early and fall into the local optimum. This is the reason that the proposed algorithm performs worse than the other ones in some test functions, such as F11, F14, F16, F17, F22, F24, and F26. Therefore, we will continue to explore the impact of the probability change of the strategies on optimization performance in subsequent studies and seek better solutions.

Author Contributions: Writing—original draft preparation, Y.W.; writing—review and editing, P.W.; visualization, J.Z.; supervision, Z.C., X.C., W.Z., and J.C.

Acknowledgments: This work is supported by the National Natural Science Foundation of China under Grant No. 61806138, No. U1636220, and No. 61663028; Natural Science Foundation of Shanxi Province under Grant No. 201801D121127; Scientific and Technological innovation Team of Shanxi Province under Grant No. 201805D131007; PhD Research Startup Foundation of Taiyuan University of Science and Technology under Grant No. 20182002; and Zhejiang Provincial Natural Science Foundation of China under Grant No. Y18F030036.

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

Appendix A

Table A1. Comparison results for mixBA and the other six algorithms.

Function	SBA	ACBA	LBA1	LBA2	FK-PSO	OCS	mixBA
F1	1.96×10^0	0.00×10^0	8.24×10^{-1}	3.59×10^{-1}	0.00×10^0	4.51×10^{-5}	1.98×10^{-5}
F2	3.69×10^6	3.04×10^5	3.54×10^6	2.23×10^6	1.59×10^6	3.18×10^0	1.27×10^3
F3	3.44×10^8	6.47×10^7	4.78×10^8	3.58×10^8	2.40×10^8	9.96×10^6	3.38×10^7
F4	3.20×10^4	1.22×10^2	1.45×10^4	6.85×10^3	4.78×10^2	7.53×10^{-3}	6.80×10^1
F5	5.86×10^{-1}	0.00×10^0	4.74×10^{-1}	2.76×10^{-1}	0.00×10^0	4.26×10^{-3}	5.82×10^{-3}
F6	5.63×10^1	2.87×10^1	5.07×10^1	4.85×10^1	2.29×10^1	8.79×10^0	1.45×10^1
F7	2.16×10^2	9.30×10^1	1.77×10^2	2.00×10^2	6.39×10^1	6.07×10^1	5.61×10^1
F8	2.09×10^1	2.10×10^1	2.09×10^1	2.10×10^1	2.09×10^1	2.10×10^1	2.09×10^1
F9	3.57×10^1	2.99×10^1	3.40×10^1	3.62×10^1	1.85×10^1	2.79×10^1	2.55×10^1
F10	1.32×10^0	1.92×10^{-1}	1.23×10^0	1.07×10^0	2.29×10^{-1}	1.40×10^{-2}	1.77×10^{-2}
F11	4.07×10^2	1.77×10^2	1.49×10^2	3.16×10^1	2.36×10^1	6.83×10^1	6.05×10^1
F12	4.06×10^2	2.73×10^2	7.42×10^2	7.18×10^2	5.64×10^1	1.07×10^2	1.03×10^2
F13	4.37×10^2	3.33×10^2	5.59×10^2	5.11×10^2	1.23×10^2	1.84×10^2	1.14×10^2
F14	4.78×10^3	2.18×10^3	3.17×10^3	1.15×10^3	7.04×10^3	2.39×10^3	1.63×10^2
F15	4.89×10^3	3.76×10^3	4.76×10^3	4.83×10^3	3.42×10^3	3.55×10^3	2.82×10^3
F16	2.16×10^0	5.61×10^{-1}	1.33×10^0	1.54×10^0	8.48×10^{-1}	1.65×10^0	9.50×10^{-1}
F17	8.92×10^2	1.96×10^2	3.36×10^2	1.61×10^2	5.26×10^2	1.61×10^2	1.33×10^2
F18	9.44×10^2	2.16×10^2	3.28×10^2	3.35×10^2	6.81×10^2	1.86×10^2	1.45×10^2
F19	6.07×10^1	1.34×10^1	1.89×10^1	1.28×10^1	3.12×10^1	7.38×10^1	6.07×10^0
F20	1.44×10^1	1.24×10^1	1.47×10^1	1.49×10^1	1.20×10^1	1.19×10^1	1.14×10^1
F21	3.38×10^2	3.22×10^2	3.22×10^2	3.05×10^2	3.11×10^2	2.88×10^2	3.01×10^2
F22	5.94×10^3	2.70×10^3	3.32×10^3	1.20×10^3	8.59×10^3	2.81×10^3	1.58×10^3
F23	5.77×10^3	4.79×10^3	6.03×10^3	5.82×10^3	3.57×10^3	4.10×10^3	3.75×10^3
F24	3.15×10^2	2.79×10^2	3.22×10^2	3.23×10^2	2.48×10^2	2.67×10^2	2.57×10^2
F25	3.49×10^2	3.14×10^2	3.53×10^2	3.54×10^2	2.49×10^2	3.01×10^2	2.94×10^2
F26	2.00×10^2	2.37×10^2	3.54×10^2	3.36×10^2	2.95×10^2	2.00×10^2	2.72×10^2
F27	1.28×10^3	1.10×10^3	1.33×10^3	1.35×10^3	7.76×10^2	9.84×10^2	8.91×10^2
F28	3.42×10^3	2.74×10^3	4.68×10^3	4.34×10^3	4.01×10^2	3.48×10^2	3.00×10^2
w\ t\	27\ 1\ 0	27\ 1\ 0	28\ 0\ 0	26\ 1\ 1	21\ 7\ 0	22\ 6\ 0	-

Table A2. The computation time of each algorithm.

Function	SBA	ACBA	LBA1	LBA2	FK-PSO	OCS	mixBA
F1	3.23×10^1	4.82×10^1	9.22×10^1	9.07×10^1	4.01×10^1	1.10×10^2	3.29×10^1
F2	5.81×10^1	7.88×10^1	1.28×10^2	1.26×10^2	7.14×10^1	1.43×10^2	5.17×10^1
F3	6.29×10^1	8.53×10^1	1.31×10^2	1.30×10^2	7.77×10^1	1.44×10^2	5.05×10^1
F4	4.26×10^1	6.09×10^1	1.06×10^2	1.06×10^2	5.16×10^1	1.24×10^2	4.13×10^1
F5	3.32×10^1	5.05×10^1	9.42×10^1	9.40×10^1	4.23×10^1	1.12×10^2	3.40×10^1
F6	4.02×10^1	6.14×10^1	1.04×10^2	1.05×10^2	5.20×10^1	1.21×10^2	3.85×10^1
F7	1.40×10^1	1.65×10^2	2.21×10^2	2.21×10^2	1.70×10^2	2.38×10^2	8.21×10^1
F8	1.26×10^1	1.39×10^2	2.02×10^2	1.90×10^2	1.34×10^2	2.10×10^2	8.55×10^1
F9	8.54×10^1	8.99×10^2	1.09×10^3	1.10×10^3	1.03×10^3	1.13×10^3	3.59×10^2
F10	7.14×10^1	9.06×10^1	1.39×10^2	1.38×10^2	8.77×10^1	1.56×10^2	5.02×10^1
F11	8.00×10^1	1.03×10^2	1.47×10^2	1.47×10^2	9.95×10^2	1.69×10^2	5.58×10^1
F12	1.03×10^1	1.27×10^2	1.76×10^2	1.75×10^2	1.31×10^2	1.97×10^2	6.68×10^1
F13	1.05×10^1	1.20×10^2	1.75×10^2	1.76×10^2	1.33×10^2	1.99×10^2	6.67×10^1
F14	8.75×10^1	1.09×10^2	1.56×10^2	1.57×10^2	1.06×10^2	1.77×10^2	6.67×10^1
F15	9.49×10^1	1.20×10^2	1.67×10^2	1.68×10^2	1.18×10^2	1.87×10^2	6.87×10^1
F16	2.13×10^2	2.47×10^2	3.15×10^2	3.11×10^2	2.66×10^2	3.26×10^2	1.23×10^2
F17	5.90×10^1	8.09×10^1	1.29×10^2	1.27×10^2	7.07×10^2	1.41×10^2	4.89×10^1
F18	7.50×10^1	9.68×10^1	1.52×10^2	1.47×10^2	9.27×10^2	1.62×10^2	5.59×10^1
F19	5.01×10^1	6.72×10^1	1.18×10^2	1.14×10^2	6.43×10^1	1.30×10^2	4.25×10^2
F20	9.18×10^1	9.34×10^1	1.62×10^2	1.66×10^2	1.10×10^2	1.83×10^2	5.71×10^2

Table A2. Cont.

Function	SBA	ACBA	LBA1	LBA2	FK-PSO	OCS	mixBA
F21	2.07×10^2	2.47×10^2	3.08×10^2	2.97×10^2	2.55×10^2	3.15×10^2	1.04×10^2
F22	2.61×10^1	2.96×10^2	3.72×10^2	3.58×10^2	3.17×10^2	3.81×10^2	1.32×10^2
F23	2.85×10^1	3.22×10^2	4.02×10^2	3.90×10^2	3.50×10^2	4.09×10^2	1.42×10^2
F24	1.05×10^3	1.11×10^3	1.37×10^3	1.31×10^3	1.31×10^3	1.33×10^3	4.27×10^2
F25	1.05×10^3	1.12×10^3	1.35×10^3	1.33×10^3	1.29×10^3	1.32×10^3	4.35×10^2
F26	1.17×10^3	1.25×10^3	1.49×10^3	1.46×10^3	1.4×10^3	1.47×10^3	4.85×10^2
F27	1.14×10^3	1.17×10^3	1.44×10^3	1.42×10^3	1.40×10^3	1.42×10^3	4.62×10^2
F28	3.98×10^2	4.35×10^2	5.25×10^2	5.21×10^2	4.83×10^2	5.14×10^2	1.70×10^2
w\ t\	26\ 0\ 2	28\ 0\ 0	28\ 0\ 0	28\ 0\ 0	28\ 0\ 0	28\ 0\ 0	-

References

1. Yang, X.S. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*; Elsevier Science Publishers B. V.: New York, NY, USA, 2013. [\[CrossRef\]](#)
2. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the International Symposium on MICRO Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43. [\[CrossRef\]](#)
3. Pan, J. Diversity enhanced particle swarm optimization with neighborhood search. *Inf. Sci.* **2013**, *223*, 119–135. [\[CrossRef\]](#)
4. Dorigo, M.; Stützle, T. Ant Colony Optimization: Overview and Recent Advances. In *Handbook of Metaheuristics*; Springer: Cham, Switzerland, 2010. [\[CrossRef\]](#)
5. Stodola, P.; Mazal, J. *Applying the Ant Colony Optimization Algorithm to the Capacitated Multi-Depot Vehicle Routing Problem*; Inderscience Publishers: Geneva, Switzerland, 2016. [\[CrossRef\]](#)
6. Cai, X.; Wang, H.; Cui, Z.; Cai, J.; Xue, Y.; Wang, L. Bat algorithm with triangle-flipping strategy for numerical optimization. *Int. J. Mach. Learn. Cybern.* **2018**, *9*, 199–215. [\[CrossRef\]](#)
7. Yang, X.S.; Deb, S. Cuckoo Search via Levy Flights. *Mathematics* **2010**, 210–214. [\[CrossRef\]](#)
8. Cui, Z.; Li, F.; Zhang, W. Bat algorithm with principal component analysis. *Int. J. Mach. Learn. Cybern.* **2018**, 1–20. [\[CrossRef\]](#)
9. Zhang, M.; Wang, H.; Cui, Z.; Chen, J. Hybrid multi-objective cuckoo search with dynamical local search. *Memet. Comput.* **2018**, *10*, 199–208. [\[CrossRef\]](#)
10. Wang, H.; Wang, W.; Sun, H.; Rahnamayan, S. Firefly algorithm with random attraction. *Int. J. Bio-Inspired Comput.* **2016**, *8*, 33–41. [\[CrossRef\]](#)
11. Wang, H.; Wang, W.; Zhou, X.; Sun, H.; Zhao, J.; Yu, X.; Cui, Z. Firefly algorithm with neighborhood attraction. *Inf. Sci.* **2017**, *382*, 374–387. [\[CrossRef\]](#)
12. Iglesias, A.; Gálvez, A.; Collantes, M. Global-Support Rational Curve Method for Data Approximation with Bat Algorithm. In Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations, Bayonne, France, 14–17 September 2015. [\[CrossRef\]](#)
13. Iglesias, A.; Gálvez, A. Memetic electromagnetism algorithm for surface reconstruction with rational bivariate Bernstein basis functions. *Natural Comput.* **2017**, *16*, 1–15. [\[CrossRef\]](#)
14. Holland, J.H. *Adaptation in Natural and Artificial System*; MIT Press: Cambridge, MA, USA, 1992. [\[CrossRef\]](#)
15. Zhao, S.Z.; Suganthan, P.N.; Zhang, Q. Decomposition-Based Multiobjective Evolutionary Algorithm with an Ensemble of Neighborhood Sizes. *IEEE Trans. Evol. Comput.* **2012**, *16*, 442–446. [\[CrossRef\]](#)
16. Yang, X.S. A New Metaheuristic Bat-Inspired Algorithm. *Comput. Knowl. Technol.* **2010**, *28*, 65–74.
17. Yang, X. Bat algorithm for multi-objective optimization. *Int. J. Bio-Inspired Comput.* **2012**, *3*, 267–274. [\[CrossRef\]](#)
18. Tharakeshwar, T.K.; Seetharamu, K.N.; Prasad, B.D. Multi-objective optimization using bat algorithm for shell and tube heat exchangers. *Appl. Therm. Eng.* **2017**, *110*, 1029–1038. [\[CrossRef\]](#)
19. Damasceno, N.C.; Filho, O.G. PI controller optimization for a heat exchanger through metaheuristic Bat Algorithm, Particle Swarm Optimization, Flower Pollination Algorithm and Cuckoo Search Algorithm. *IEEE Lat. Am. Trans.* **2017**, *15*, 1801–1807. [\[CrossRef\]](#)
20. Hasançebi, O.; Teke, T.; Pekcan, O. A bat-inspired algorithm for structural optimization. *Comput. Struct.* **2013**, *128*, 77–90. [\[CrossRef\]](#)
21. Gandomi, A.H.; Yang, X.S. Chaotic bat algorithm. *J. Comput. Sci.* **2014**, *5*, 224–232. [\[CrossRef\]](#)

22. Fister, I.; Fong, S.; Brest, J. A novel hybrid self-adaptive bat algorithm. *Sci. World J.* **2014**, *2014*, 709–738. [[CrossRef](#)] [[PubMed](#)]
23. Ali, E.S. Optimization of Power System Stabilizers using BAT search algorithm. *Int. J. Electr. Power Energy Syst.* **2014**, *61*, 683–690. [[CrossRef](#)]
24. Xie, J.; Zhou, Y.Q.; Chen, H. A bat algorithm based on Lévy flights trajectory. *Pattern Recognit. Artif. Intell.* **2013**, *26*, 829–837. [[CrossRef](#)]
25. Pérez, J.; Valdez, F.; Castillo, O. A New Bat Algorithm Augmentation Using Fuzzy Logic for Dynamical Parameter Adaptation. In Proceedings of the Mexican International Conference on Artificial Intelligence, Cuernavaca, Mexico, 25–31 October 2015; Springer: Cham, Switzerland, 2015; pp. 433–442. [[CrossRef](#)]
26. Liu, C. Bat algorithm with Levy flight characteristics. *CAAI Trans. Intell. Syst.* **2013**, *3*, 240–246.
27. Yilmaz, S.; Kucuksille, E.U. Improved Bat Algorithm (IBA) on Continuous Optimization Problems. *Lect. Notes Softw. Eng.* **2013**, *1*, 279. [[CrossRef](#)]
28. Cai, X.; Wang, L.; Kang, Q.; Wu, Q. Adaptive bat algorithm for coverage of wireless sensor network. *Int. J. Wirel. Mob. Comput.* **2015**, *8*, 271–276. [[CrossRef](#)]
29. Bahmani-Firouzi, B.; Azizipناه-Abarghooee, R. Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm. *Int. J. Electr. Power Energy Syst.* **2014**, *56*, 42–54. [[CrossRef](#)]
30. Yilmaz, S.; Kucuksille, E.U. A new modification approach on bat algorithm for solving optimization problems. *Appl. Soft Comput.* **2015**, *28*, 259–275. [[CrossRef](#)]
31. Deng, Y.; Duan, H. Chaotic mutated bat algorithm optimized edge potential function for target matching. In Proceedings of the 10th Conference on Industrial Electronics and Applications (ICIEA), Auckland, New Zealand, 15–17 June 2015. [[CrossRef](#)]
32. Xie, J.; Zhou, Y.; Chen, H. A Novel Bat Algorithm Based on Differential Operator and Lévy Flights Trajectory. *Comput. Intell. Neurosci.* **2013**, *2013*, 453–812. [[CrossRef](#)] [[PubMed](#)]
33. Zhu, B.; Zhu, W.; Liu, Z.; Duan, Q.; Cao, L. A Novel Quantum-Behaved Bat Algorithm with Mean Best Position Directed for Numerical Optimization. *Comput. Intell. Neurosci.* **2016**, *2016*, 1–17. [[CrossRef](#)]
34. Cui, Z.; Sun, B.; Wang, G.; Xue, Y.; Chen, J. A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems. *J. Parallel Distrib. Comput.* **2017**, *103*, 42–52. [[CrossRef](#)]
35. Yang, X.S.; Gandomi, A.H. Bat Algorithm: A Novel Approach for Global Engineering Optimization. *Eng. Comput.* **2012**, *29*, 464–483. [[CrossRef](#)]
36. Bora, T.C.; Coelho, L.D.S.; Lebensztajn, L. Bat-Inspired Optimization Approach for the Brushless DC Wheel Motor Problem. *IEEE Trans. Magn.* **2012**, *48*, 947–950. [[CrossRef](#)]
37. Sambariya, D.K.; Prasad, R. Robust tuning of power system stabilizer for small signal stability enhancement using metaheuristic bat algorithm. *Int. J. Electr. Power Energy Syst.* **2014**, *61*, 229–238. [[CrossRef](#)]
38. Sathya, M.R.; Ansari, M.M.T. Load frequency control using Bat inspired algorithm based dual mode gain scheduling of PI controllers for interconnected power system. *Int. J. Electr. Power Energy Syst.* **2015**, *64*, 365–374. [[CrossRef](#)]
39. Sun, S.; Xu, B. Node localization of wireless sensor networks based on hybrid bat-quasi-Newton algorithm. *J. Comput. Appl.* **2015**, *11*, 38–42. [[CrossRef](#)]
40. Cao, Y.; Cui, Z.; Li, F.; Dai, C.; Chen, W. Improved Low Energy Adaptive Clustering Hierarchy Protocol Based on Local Centroid Bat Algorithm. *Sens. Lett.* **2014**, *12*, 1372–1377. [[CrossRef](#)]
41. Cui, Z.; Cao, Y.; Cai, X.; Cai, J.; Chen, J. Optimal LEACH protocol with modified bat algorithm for big data sensing systems in Internet of Things. *J. Parallel Distrib. Comput.* **2017**. [[CrossRef](#)]
42. Cui, Z.; Xue, F.; Cai, X.; Cao, Y.; Wang, G.G.; Chen, J. Detectin of malicious code variants based on deep learning. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3187–3196. [[CrossRef](#)]
43. Hamidzadeh, J.; Sadeghi, R.; Namaei, N. Weighted Support Vector Data Description based on Chaotic Bat Algorithm. *Appl. Soft Comput.* **2017**, *60*, 540–551. [[CrossRef](#)]
44. Alslibi, B.; Venkat, I.; Al-Betar, M.A. A membrane-inspired bat algorithm to recognize faces in unconstrained scenarios. *Eng. Appl. Artif. Intell.* **2017**, *64*, 242–260. [[CrossRef](#)]
45. Cui, Z.; Zhang, J.; Wang, Y.; Cao, Y.; Cai, X.; Zhang, W.; Chen, J. A pigeon-inspired optimization algorithm for many-objective optimization problems. *Sci. China Inf. Sci.* **2019**. [[CrossRef](#)]
46. Tharwat, A.; Hassanien, A.E.; Elnaghi, B.E. A BA-based algorithm for parameter optimization of Support Vector Machine. *Pattern Recognit. Lett.* **2016**, *93*, 13–22. [[CrossRef](#)]

47. Basith, S.; Manavalan, B.; Shin, T.H.; Lee, G. iGHBP: Computational identification of growth hormone binding proteins from sequences using extremely randomised tree. *Comput. Struct. Biotechnol. J.* **2018**, *16*, 412–420. [[CrossRef](#)]
48. Zamuda, A.; Brest, J.; Mezura-Montes, E. Structured Population Size Reduction Differential Evolution with Multiple Mutation Strategies on CEC 2013 real parameter optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 1925–1931. [[CrossRef](#)]
49. Manavalan, B.; Subramaniam, S.; Shin, T.H.; Kim, M.O.; Lee, G. Machine-learning-based prediction of cell-penetrating peptides and their uptake efficiency with improved accuracy. *J. Proteome Res.* **2018**, *17*, 2715–2726. [[CrossRef](#)]
50. Friedman, J.H. Fast sparse regression and classification. *Int. J. Forecast.* **2012**, *28*, 722–738. [[CrossRef](#)]
51. Sun, H.; Wang, K.; Zhao, J.; Yu, X. Artificial bee colony algorithm with improved special centre. *Int. J. Comput. Sci. Math.* **2017**, *7*, 548–553. [[CrossRef](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).