Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9495281, 22 pages
https://doi.org/10.1155/2020/9495281

Research Article

Bat Algorithm Based on an Integration Strategy and Gaussian Distribution

Jianqiang Huang and Yan Ma

College of Information and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China

Correspondence should be addressed to Yan Ma; ma-yan@shnu.edu.cn

Received 8 July 2020; Revised 21 September 2020; Accepted 23 September 2020; Published 10 October 2020

Academic Editor: Calogero Orlando

The bat algorithm (BA) is a recent heuristic optimization algorithm based on the echolocation behavior of bats. However, the bat algorithm tends to fall into local optima and its optimization results are unstable because of its low global exploration ability. To solve these problems, a novel bat algorithm based on an integration strategy (IBA) is proposed in this paper. Through the integration strategy, an appropriate operator is adaptively selected to perform global search, so that the global search ability of the IBA is improved. Furthermore, the IBA disturbs the local optimum through a linear combination of Gaussian functions with different variances to avoid becoming trapped in local optima. The IBA also updates the velocity equation with an adaptive weight to further balance the exploration and exploitation. Moreover, the global convergence of the IBA is proved based on the convergence criterion of a stochastic algorithm. The performance of the IBA is evaluated on CEC2013 benchmark functions and compared with that of the standard BA as well as several of its variants. The results show that the IBA is superior to other algorithms.

1. Introduction

Optimization usually involves highly nonlinear complex problems with many design variables and complex constraints [1]. Generally, the form of nonlinear constrained optimization problem can be formulated as follows:

$$\begin{cases} \text{Minimize } f(x), \\ \text{s.t. } g_l(x) \le 0, \quad l = 1, 2, \ldots, k, \\ h_j(x) = 0, \quad j = 1, 2, \ldots, p, \end{cases} \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ is n-dimensional decision variables, $f(x)$ is the objective function, $g_l(x) \le 0$ denotes the inequality constraints, and $h_j(x) = 0$ denotes the equality constraints. Traditional deterministic methods or algorithms do not cope well when solving a large number of problems in practice, especially when the objective function is multimodal with many local optima. Over the past years, over a dozen metaheuristic algorithms have been developed based on inspiration from different natural processes. For instance, the genetic algorithm [2] is based on the biological evolution processes and ant colony optimization [3] is based on swarm behavior. Harmony search is an algorithm inspired by the music composition process of musicians. The particle swarm optimization (PSO) algorithm [4] is inspired from swarming behaviors such as bird flocking and fish schooling in nature. An evolutionary algorithm named the oriented cuckoo search (OCS) algorithm [5] was motivated by the aggressive breeding habits of a bird called the "cuckoo." These algorithms have been used to solve nonlinear complex problems because of their simple structures and abilities to obtain a solution, and they are referred to as nature-inspired algorithms or bioinspired algorithms.

In recent years, many such metaheuristic algorithms have been proposed. Wu et al. [6] proposed an enhanced harmony search algorithm with circular region perturbation, and Gupta and Deep [7] introduced a new crossover operator called the double distribution crossover. An aggregative learning gravitational search algorithm was proposed by Lei et al. [8], and Mohamed et al. [9] proposed a novel

nature-inspired algorithm called the gaining sharing knowledge algorithm, which mimics the process of gaining and sharing knowledge during the human life span. Attention should also be drawn to novel algorithms [10–14] based on sine cosine algorithms. Moreover, many metaheuristic algorithms were proposed for solving constrained nonlinear programming problems (CNLPPs). Han et al. [15] developed a new hybrid moth search-fireworks algorithm to solve numerical and constrained engineering optimization problems. Baykasoğlu et al. [16] introduced a new metaheuristic, single seekers society algorithm, for solving unconstrained and constrained continuous optimization problems. Shadravan et al. [17] presented a novel nature-inspired metaheuristic optimization algorithm, called sailfish optimizer, which is inspired by a group of hunting sailfish. Kaur et al. [18] proposed a novel hybrid multiobjective optimization algorithm by synthesizing the strengths of multiobjective spotted hyena optimizer and salp swarm algorithm. Montemurro et al. [19] presented a new penalty-based approach, developed within the framework of genetic algorithms for constrained optimization problems. Montemurro et al. [20] presented a nonclassical genetic algorithm to solve the design of laminates with a minimum number of layers. Costa et al. [21] provided a general methodology to approximate sets of data points through nonuniform rational basis spline (NURBS) curves.

The bat algorithm (BA) is a nature-inspired metaheuristic algorithm. It was proposed by Yang in 2010 to imitate the echolocation behavior of bats [22]. The BA has been widely applied in many applications, such as engineering optimization [23, 24] and pattern recognition [25]. Next, we introduce three aspects of the BA in detail: parameters, algorithm structure, and application.

1.1. Parameters. Four main parameters are involved in the standard BA: pulse frequency, pulse rate, velocity, and a constant. For the standard BA, it is difficult to find a balance between global search and local search, which leads to a slow convergence rate. To solve this problem, Gandomi and Yang [26] introduced chaos into the standard BA (CBA) to increase its global search mobility for robust global optimization. In CBA, different chaotic systems are used to replace the parameters in BA. Xie et al. [27] proposed an improved BA based on the Lévy flight trajectory. This algorithm can effectively jump out of local optima using the strategy of an adaptively adjusted frequency. Gan et al. [28] proposed a new BA based on iterative local search and stochastic inertia weight. A stochastic inertial weight is considered in the velocity updating equation, which can enhance the diversity and flexibility of the bat population.

1.2. Algorithm Structure. The optimization performance of the standard BA mainly depends on the interaction and influence between individuals, which may lead to a local optimum. Liu and Chunming [29] introduced the Lévy flight behaviors of bats and took full advantage of the trait of uneven random walks to enable the algorithm to avoid becoming trapped in a locally optimal solution. To enhance the ability of the algorithm to escape from locally optimal values, Boudjemaa et al. [30] proposed the fractional Lévy flight BA (FLFBA), in which the velocity is updated through fractional calculus. Fister et al. [31] proposed a hybrid BA by combining it with differential evolution. To improve the global searching ability, Al-Betar and Awadallah [32] divided the whole bat population into two subgroups and specified the movement of bat individuals from one group to another by mobility. Jaddi et al. [33] proposed to modify the velocity equation of the standard BA to better balance exploration and exploitation in the population, and Ghanem and Jantan [34] proposed an enhanced BA to enhance the diversity of the standard BA using a special mutation operator.

1.3. Applications. Recently, BA has been widely used in the fields of optimization, modeling, and control. Dao et al. [35] used parallel BA to solve a workshop scheduling problem. Osaba et al. [36] proposed a discrete BA to solve the vehicle path problem of drug waste collection and distribution. Aiming at the data loss problem in high-dimensional data, Leke and Marwala [37] proposed to estimate missing data based on BA. To improve the accuracy of the generated fuzzy rules, Cheruku et al. [38] analyzed big data for diabetes detection by combining rough set feature selection with optimized BA. Nakamura et al. [39] proposed a binary BA to solve feature selection problems and proved that the algorithm outperforms other swarm intelligence algorithms. Goyal and Patterh [40] proposed a modified BA to evaluate the precision of node localization in wireless sensor networks.

Although the aforementioned algorithms addressed the problems of BA, they cannot balance exploration and exploitation capabilities, and the stability of the results cannot be guaranteed. Hence, these methods cannot really improve the performance of the standard BA. To tackle the above problems, this paper proposes a novel BA that uses an integration strategy (IBA) to enhance the search ability while maintaining the stability of the results. In this paper, we present three main contributions: (i) we adaptively select the appropriate operator for performing global search through the integration strategy, which can improve the global search ability of the algorithm; (ii) we disturb a local optimum through a linear combination of Gaussian functions with different variances, so that the IBA has the ability to jump out of the local optima; and (iii) we update the velocity equation of the standard BA with an adaptive weight to balance the exploration and exploitation and keep the algorithm stable.

In addition, different constraint-handling methods were proposed to solve CNLPPs: (i) hybrid methods; (ii) repair algorithms; (iii) unique representatives and operators; (iv) isolation of objectives and constraints; (v) penalty functions. In this paper, we use penalty function to solve SNLPPs because penalty function is the simplest one to solve the constrained problem among the above methods.

The rest of the paper is organized as follows: Section 2 describes the structure of standard BA. Section 3 describes

the main idea of IBA and its flowchart. Section 4 proves the global convergence of IBA. Experimental and comparative analyses of the results obtained from IBA are presented in Section 5. Finally, Section 6 concludes the paper. The acronyms and their meaning are given in Table 1.

2. BA

The BA is inspired by the echolocation behavior of bats when sensing distances. When bats hunt at night, they typically emit short, loud sound impulses and listen to the echo bouncing back from an obstacle or prey. A bat can use its special auditory mechanism to identify the size and position of the object. Yang [22] proposed the BA based on this echolocation characteristic of bats. The steps of BA can be summarized as follows:

Step 1. Initialize the bat parameters, as shown in Table 2.

Step 2. Update the global best position X^*, pulse frequency, velocity, and position of the i^{th} bat as follows:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \quad \beta \in [0, 1], \tag{2}$$

$$V_i^{t+1} = V_i^t + \left(X_i^t + X^*\right) f_i, \tag{3}$$

$$X_t^{t+1} = X_i^t + V_i^t, \tag{4}$$

where V_i^t and X_i^t are the velocity and position at time t, V_i^{t+1} and X_i^{t+1} are the velocity and position at time $t+1$, and β is a random number between 0 and 1.

Step 3. If the random number is greater than r_i, a new solution for the bat is generated by the following equation:

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

where ε is a random number, $\varepsilon \in [-1, 1]$, and A^t represents the average loudness of all bats at time t.

Step 4. If the random number is lower than A_i and $f(X_i) < f(X^*)$, the new solution is accepted. Next, update A_i and r_i, respectively, as follows:

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

$$r_i^t = r_i^0 \left[1 - e^{-\gamma t}\right], \tag{7}$$

where A_i^{t+1} and A_i^t denote the loudness at times t and $t+1$, respectively; r_i^0 and r_i^t are the initial pulse rate and pulse rate at time t, respectively, α is a constant parameter in range [0, 1], γ is a constant parameter, and $\gamma > 0$. As $t \longrightarrow \infty$, $A_i^t \longrightarrow 0$ and $r_i^t \longrightarrow r_i^0$.

Step 5. Sort the bats based on their fitness and find the current optimal solution X^*.

Step 6. Return to Step 2 until the maximum number of iterations is reached; output the globally optimal solution.

3. BA Based on an Integrated Strategy and Gaussian Distribution

This section explains the basic principle of IBA in detail, which is based on integrated strategy and local search with adaptive parameters. The goal is to address the problems of standard BA: local-optima traps, slow convergence speed, and unstable optimization results. This paper presents the following improvements: (i) an adaptive weight; (ii) a representation of the random disturbance using a linear combination of two Gaussian distributions with different variances; (iii) determination of optimal solution with an integrated strategy; and (iv) local search.

3.1. Constraint-Handling Technique-Based Method. For the constrained nonlinear programming problem (CNLPP), the penalty method is a common method, whose core idea is to transform the constrained problem into unconstrained problem with penalty function. In general, for the metaheuristic algorithm, the equality constraint in equation (1) can be modified as follows:

$$\left|h_q(x)\right| - \delta \le 0 \, (q = k+1, k+2, \ldots, k+p), \tag{8}$$

where δ is a plus tolerance number to equality constraints. Therefore, we define the following penalty function:

$$G_l(x) = \begin{cases} \max\left(0, g_l(x)\right), \; l = 1, 2, \ldots k, \\ \max\left(0, \left|h_l(x) - \delta\right|\right), \quad l = k+1, k+2, \ldots, k+p. \end{cases} \tag{9}$$

We define the new objective function according to equation (1):

$$\Pi(x, \lambda) = f(x) + \sum_{l=1}^{k+p} \left(\lambda G_l(x)\right), \tag{10}$$

where λ is a penalty parameter. Then, the new objective function can be further expressed as

$$\Pi\left(x, \mu_e, \nu_j\right) = f(x) + \sum_{e=1}^{k+p} \mu_e \phi_e^2(x) + \sum_{j=1}^{k+p} \nu_j \psi_j^2(x), \tag{11}$$

where φ_e and ψ_j denote inequality constraints and equality constraints, respectively, and μ_e and ν_j denote the penalty parameters of inequality constraints and equality constraints. The value of the penalty parameter should be taken as large as possible depending on the solution quality needed. From the above analysis, we can see that when the equality constraints are satisfied, the effect of μ_e to objective function is zero. However, when the equality constraints are not satisfied, μ_e is heavily penalized as it significantly increases. Similarly, it is the same with the penalty parameter ν_j for the case of inequality constraints.

Table 1: List of acronyms.

Acronym	Meaning
ACO	Ant colony optimization
APSO	Adaptive particle swarm optimizer
BA	Bat algorithm
CBA	Chaotic bat algorithm
CLPSO	A comprehensive learning particle swarm optimizer
CMA	Constant modulus algorithm
CNLPPs	Constrained nonlinear programming problems
CPSOH	A cooperative approach to particle swarm optimization
FIPS	The fully informed particle swarm
FLFBA	Fractional Lévy flight bat algorithm for global optimizations
FLFBA	Fractional Lévy flight bat algorithm
FPSO	Frankenstein's PSO
G-CMA-ES	A restart CMA evolution strategy with increasing population size
HRCGA	Global and local real-coded genetic algorithm
IBA	A novel bat algorithm by using integration strategy
IBA-1	IBA without integration strategy
IBA-2	IBA without Gaussian function
IBA-3	IBA without adaptive weight
ILSA	An improved local searching algorithm
JADE	Adaptive differential evolution with optional external archive
LBA	Bat algorithm with Lévy distribution
LSA	Local search algorithm
OCS	Oriented cuckoo search
PDF	Probability density function
PSO	Particle swarm optimization algorithm
SLPSO	Social learning particle swarm optimization
SPSO	A novel supervised particle swarm optimization

Table 2: BA parameters.

M	The size of bat population
N	Max number of iterations
I	The number of bat, in range [1, M]
X^*	The current global best location (solution)
X_i	The position of i^{th} bat
V_i	The velocity of i^{th} bat
f_i	The pulse frequency of the i^{th} bat and its range is between $f_{\min}$ and $f_{\max}$
$f(X)$	Fitness function
r_i	The pulse rate of i^{th} bat
A_i	The loudness of i^{th} bat
α	The constant parameter in range [0, 1] used to update the loudness A_i
γ	The constant parameter in range [0, 1] used to update the pulse rate r_i

When BA is used to solve the CNLPPs, its searching mechanism can be expressed as the following optimization problem:

$$\begin{aligned} &\min \quad f(X) \\ &\text{s.t.} \quad X \in R^D, \end{aligned} \tag{12}$$

where $f(x)$ is fitness function and R^D denotes the searching space of BA. Suppose position $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$ is a feasible solution. In initialization, the bat individuals are generated randomly in a searching space. The bat parameters, including pulse frequency, velocity, and position, are updated according to equations (2)–(4). It can be seen from equations (2)–(4) that the velocity V_i and position X_i are updated according to the randomly generated pulse frequency f_i. After the update, BA searches the local-optimal solution according to the average loudness of all bats in a random manner, as shown in equation (5). It needs to be noted that the local search is carried out with the pulse rate r_i. The local search will be conducted if the random number is greater than r_i. From equation (5), we can see that the range of local search is dependent on the average loudness. Then, if the random number is lower than A_i and the value of current solution is lower than the value the optimal solution, the new solution will be accepted according to the rules of the feasible solution. The loudness and pulse rate are updated according to equations (6) and (7).

Similar to [41], the feasibility-based rules for BA can be defined as follows: (1) any feasible solution is superior to any infeasible solution. (2) Between two feasible solutions, the one having a better objective function value is preferred. (3) Between two infeasible solutions, the one having a smaller constraint value is preferred. To summarize, these rules are to choose a solution that lies closer to the feasible region.

According to the above description, we present the pseudocode of BA in Algorithm 1.

3.2. Parameter Improvement. It is generally known that a suitable value for the inertia weight provides a balance between the global and local exploration ability of the algorithm. Shi and Eberhart [42] pointed out that a better performance would be obtained if the inertia weight was chosen a time varying, linearly decreasing quantity. It was inferred that the system should start with a high inertia weight for global exploration and this weight should decrease to facilitate finer local explorations in later iterations. The concave model [43], as a nonlinear model can meet these requirements for inertia weight. However, the inertia weight generated by a concave function will greatly accelerate the convergence rate, which tends to make the algorithm fall into local optima. Inspired by Kentzoglanakis and Poole [44], we define an adaptive weight w as follows, with the help of the sine function:

$$w = \begin{cases} 1 - \sin\left(\pi - \dfrac{\pi * t}{2 * T_{\max}}\right), & t < T_{\max}, \\ 1 - \sin\left(\pi - \dfrac{\pi * (t - \varepsilon)}{2 * T_{\max}}\right), & t = T_{\max}, \end{cases} \tag{13}$$

where $\varepsilon \longrightarrow 0+$. We then modify the velocity update equation as follows to solve this issue:

$$V_i^{t+1} = wV_i^t + \left(X_i^t + X^*\right) f_i. \tag{14}$$

Figure 1 shows how the values of w vary with time t. It can be seen from Figure 1 that the values of w tend to decrease as the number of iterations increases. In equation (13), t is subtracted by ε to avoid outputting 0 for w when t is equal to $T_{\max}$. We introduce an adaptive weight into equation (13), which will make the velocity update more flexible. At the beginning of the iterations, the individual has a higher speed when the value of the weight is large, which can speed up the search process and improve its global search ability. In contrast, the individual has a lower speed when the value of the weight is smaller in the last stages of the iterations, which can improve its local search capability and ensure the stability of the algorithm.

The optimal position of the bat population is adjusted using random number $X_{\text{new}} = X_{\text{old}} + 0.5\,(N(0,1) + N(0,2))A^t$, which follows a uniform distribution in the interval [−1, 1]. To enhance the search performance of the algorithm, He et al. [45] introduced Gaussian perturbations into the standard BA, instead of a uniform distribution. Inspired by Chellapilla et al. [46], we modify the random disturbance in the standard BA into a linear combination of two standard Gaussian distributions as follows:

$$X_{\text{new}} = X_{\text{old}} + 0.5\,(N(0,1) + N(0,2))A^t, \tag{15}$$

where $N(0,1)$ is a random number drawn from a distribution with zero mean and a standard deviation of one; $N(0,2)$ is a random number drawn from a Gaussian distribution with zero mean and a standard deviation of two. Figure 2 shows the probability density function (PDF) of the linear combination. Two standard Gaussian PDFs are also plotted for comparison. For analysis, based on Figure 2, the range of x-axis is split into two categories, around the mean (−1.8–1.8) and far from the mean (<−1.8, or >1.8). In comparison with N (0, 1), this linear combination generates fewer random numbers around the mean. In comparison with N (0, 2), the linear combination generates fewer random numbers far from the mean. Thus, the linear combination can achieve better disturbance performance and avoid allowing an individual falling into local optima.

3.3. Improved Local Search Algorithm. The standard BA will fall into the local optima during the iterations. To solve this issue, we propose an improved local search algorithm (ILSA). The basic principle of ILSA is to find the exact optimal solution according to multiple fitness values. ILSA operates as follows:

Step 1. Generate the neighborhood set of the best position X^* using the following equation:

$$N\colon X^* \longrightarrow \text{rand}^* X^*, \tag{16}$$

where rand is a random number in the range of [0, 1]. We obtain a neighborhood set $N(X^*) = \{X_1^*, X_2^*\}$ from equation (16). We also assume that $X_1^* < X_2^*$.

Step 2. Calculate the fitness value $f(X_1^*)$ of $X^* = X_1^*$ according to the objective function $f(X_1^*) < f(X^*)$.

Step 3. If $f(X_1^*) < f(X^*)$, the current optimal solution is $X^* = X_1^*$, $f(X^*) = f(X_1^*)$; otherwise, select $\text{mix}_k(t)$ from $N(X^*)$ and calculate its fitness value $f(X_2^*)$. If $f(X_2^*) < f(X^*)$, the current optimal solution is $X^* = X_2^*$, $f(X^*) = f(X_2^*)$.

Step 4. Output the best solution X^* and stop the search.

In terms of the optimal algorithm, there exists a certain specific operator in the iterations whose performance is superior than other operators [47]. Therefore, the global search capability of the algorithm can be further improved by selecting the appropriate operators at different times. In this paper, we propose an optimal operator selection strategy for the velocity update. The main idea is to update the velocity by selecting the appropriate operator and further improve the exploration ability of the algorithm. The selection strategy determines whether the IBA can jump out of

(1) Initialize position X_i and velocity V_i, $i = 1, 2, \ldots, M$;
(2) Initialize pulse rates r_i and loudness A_i;
(3) Define pulse frequency f_i of the ith bat and in the range between $f_{\min}$ and $f_{\max}$;
(4) Evaluate all the elements in the population by objective function $f(X)$ and the constraint value of each bat by the constrained functions.
(5) Initialize the number of iteration $t = 1$;
(6) **while** ($t <$ max number of iterations N)
(7) **For** each bat
(8) Update the locations/solutions (equation (4));
(9) **If** rand $> r_i$
(10) Select a solution from the best solutions;
(11) Generate a local solution around the best solution (equation (5));
(12) **End if**
(13) Evaluate the fitness values and constraint values of the offspring
(14) **If** rand $< A_i \& f(X_i) < f(X^*)$
(15) Accept the new solutions as the feasibility-based rules;
(16) Update the fitness;
(17) Update the pulse rate r_i and loudness A_i (equations (6) and (7));
(18) **End if**
(19) **End for**
(20) Rank the bats and find the current best X^*;
(21) $t = t+1$
(22) **End while**
(23) Postprocess results and visualization;

Algorithm 1: BA.

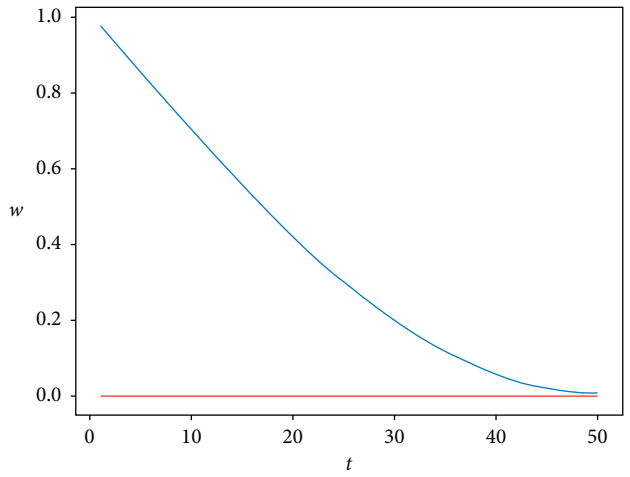

Figure 1: Adaptive weight, which (w) varies with the number of iterations.

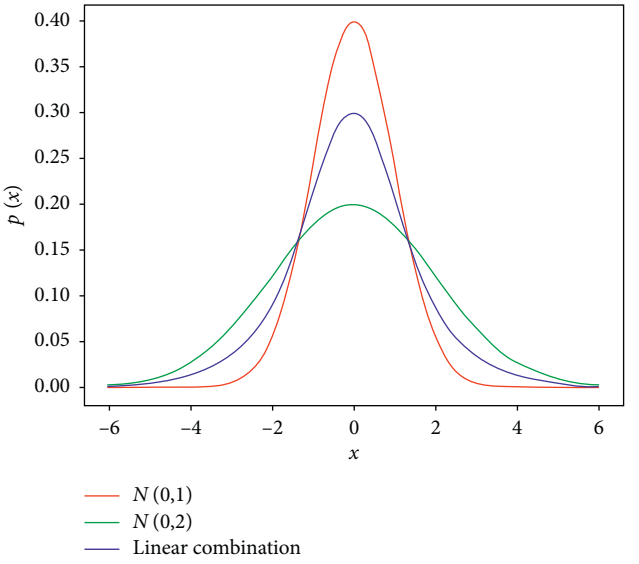

Figure 2: PDF of the linear combination in comparison with two standard Gaussian PDFs.

the local optima. If the IBA can jump out the local optima, we randomly select other operators. Otherwise, we select the current operator.

Based on the above idea, equation (14) can be modified as follows:

$$V_i^{t+1} = wV_i^t + \text{mix}_k(t), \tag{17}$$

where $\text{mix}_k(t)$ represents the k^{th} velocity update operator.

In this paper, we select the following three velocity update operators:

(1) An operator based on the standard BA [48]:

$$\left(X_i^t - X^*\right) f_i. \tag{18}$$

(2) An operator based on chaotic BA [27]:

$$\left(X_i^t - X^*\right) \text{CM}_i f_i, \tag{19}$$

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right) \text{CM}_i, \tag{20}$$

where CM_i is the generic name of chaotic maps consisting of some map functions.

(3) An operator based on an improved BA [49]:

$$\left(X_i^t - W_t\right) f_i, \tag{21}$$

where W^t is the worst position.

We present the pseudocode of the IBA in Algorithm 2.

4. Convergence Analysis of IBA

IBA is a stochastic optimization algorithm just like other heuristic optimization algorithms [50]. In this section, we give the global convergence proof of the IBA based on the convergence criteria of stochastic algorithms [51]. We first introduce the definition and theorem and then prove the global convergence of IBA.

4.1. Definition of IBA

Definition 1. State and state space of IBA: X is the state of bat and L represents the feasible solution space, denoted as $X \in L.\psi$ indicates the state of IBA consists of all bats, denoted as $\psi = (X_1, X_2, \ldots, X_N)$. Furthermore, S is the state space of IBA, denoted as $S = (\psi_1, \psi_2, \ldots, \psi_N)$.

Definition 2. State of IBA transition probability: in IBA, the process of changing from one state X_i to another state X_j is defined as the state transition of a bat, denoted as $E_\psi(X_i) = X_j$. The probability equation is defined as

$$P\left(E_S(\psi_i) = \psi_j\right) = P\left(E_\psi(X_i)\right) = X_j. \tag{22}$$

Lemma 1. *(see [52]). In probability space, z denotes the existing solution of the random algorithm, D represents the operator of generating solution by the algorithm, and ζ indicates the generating solution of the random algorithm. For the objective function f, if $f(D(z,\zeta)) \le f(z)$, then the inequality can be expressed as follows:*

$$f(D(z,\zeta)) \le f(\zeta). \tag{23}$$

Lemma 2. *(see [52]). For any Borel set in the domain S of an objective function, if $L(A) > 0$, the formula is shown as follows:*

$$\prod_{k=0}^{+\infty} \left(1 - \mu_k(A)\right) = 0, \tag{24}$$

where $L(A)$ is the Lebesgue measure of set A and $\mu_k(A)$ is the probability of generating set A.

Lemma 3. *(see [52]). The necessary and sufficient conditions for global convergence of the IBA are that $\{p(k)\}_{k=1}^{+\infty}$ is the optimal sequence of locations, and Lemmas 1 and 2 are both simultaneously satisfied.*

4.2. Convergence Analysis of IBA

Theorem 1. *IBA satisfies Lemma 1.*

Proof. In every iteration of ILSA, the current optimal solution X_{old} is perturbed, the best solution X^* is disturbed, and the acceptance criterion is used. For this reason, IBA obtains the best solution in every iteration and Theorem 1 is proved. □

Definition 4. The optimal state set is $B = \{\psi^* = (X_1, X_2, \ldots, X_N) \mid f(X_N) = f(b^*), \psi^* \in S\}$, where b^* is the optimal solution $B \subset L$.

Theorem 2. *B is a closed set in IBA.*

Proof. $\forall \psi_i \in B, \forall \psi_j \notin B$, and the transition probability from state ψ_i to ψ_j after the M^{th} $(M \ge 1)$ step is as follows:

$$P^M\left(T_S\left(\psi_i = \psi_j\right)\right) = \prod_{k=1}^{M} P\left(T_S(\psi_i) = \psi_{j-k+1}\right). \tag{25}$$

The state transition probability of IBA is shown as follows:

$$P\left(T_S(\psi_i) = \psi_{j-i+1}\right) = \prod_{m=1}^{N} P\left(T_\psi(X_{im}) = X_{j-l+1,m}\right). \tag{26}$$

$\forall \psi_i \in B, \forall \psi_j \notin B$, and $\exists_{\psi_{j-k+1}} \notin B (1 \le k \le M)$ in the M^{th} iteration, and then $f(X_{j-k+1}) \le f(X_{j-k}) = f(b^*)$. From the acceptance criterion of ILSA, we have $P(T_S(\psi_{j-k}) = \psi_{j-k+1}) = 0$, $P^M(T_S(\psi_i) = \psi_j) = 0$, and hence, Theorem 2 is proved. □

Theorem 3. *The nonempty closed set G does not exist in state space S.*

Proof. We make the following assumption: a nonempty closed set E exists in the state space S. Let $\psi_i^* = (b_{i1}^*, b_{i2}^*, \ldots, b_{ic}^*) \in B$, $\psi_j = (X_{j1}, X_{j2}, \ldots, X_{\text{jn}}) \in E$, then $f(\zeta_{\text{jn}}) \le f(b_{\text{ic}}^*)$, then it has $P^M(T_S(\psi_j) = \psi_j^*) > 0$. As we can see from Theorem 2, E is not a closed set, which is a contradiction with Lemma 3. Theorem 3 is proved, which gives us $G \cap B = \Phi$. □

Theorem 4. *The state of IBA becomes optimal state set B as the iteration times tend to infinity.*

Proof. As can be seen from Theorems 2 and 3, state space S is not composed of closed sets, which is beyond the optimal state set B. If $\zeta_j \notin B$, we have $\lim_{n \to \infty} P(T_S(X_n) = X_j) = 0$. Therefore, optimal state set B includes the state of IBA, and Theorem 4 is proved. □

Theorem 5. *IBA satisfies Lemma 2.*

Proof. As we can see from Theorem 4, when the number of iterations approaches infinity, the probability that the globally optimum solution is searched becomes 1 and

(1) Define the objective function $f(X), X = (x_1, x_2, \ldots, x_D)^T$;
(2) Initialize position X_i and velocity V_i, $i = 1, 2, \ldots, M$;
(3) Initialize pulse rates r_i and loudness A_i;
(4) Define pulse frequency f_i of the i^{th} bat and in the range between $f_{\min}$ and $f_{\max}$;
(5) Evaluate all the elements in the population by objective function $f(X)$;
(6) Initialize the number of iteration $t = 1$;
(7) **while** ($t <$ max number of iterations N)
(8) **For** each bat
(9) **If** ILSA jump out of the local optima
(10) Select other velocity update operators randomly and update velocities (equations (17)–(21));
(11) **Else**
(12) Select the current velocity update operator, update velocities (equations (17)–(21));
(13) **End if**
(14) Update the locations/solutions (equation (4));
(15) **If** rand $> r_i$
(16) Select a solution from the best solutions;
(17) Generate a local solution around the best solution (equation (5));
(18) **End if**
(19) **If** rand $< A_i \& f(X_i) < f(X^*)$
(20) Accept the new solutions;
(21) Update the fitness;
(22) Update the pulse rate r_i and loudness A_i (equations (6) and (7));
(23) **End if**
(24) **End for**
(25) Obtain the disturbed solutions X_1^*, X_2^* (equation (16));
(26) **If** $f(X_1^*) < f(X^*)$
(27) $X^* = X_1^*$;
(28) **Else**
(29) **If** $f(X_2^*) < f(X^*)$
(30) $X^* = X_2^*$;
(31) **End if**
(32) **End if**
(33) Rank the bats and find the current best X^*;
(34) $t = t+1$
(35) **End while**
(36) Postprocess results and visualization;

Algorithm 2: IBA.

$0 < \mu_k[G] < 1$; then, $\prod_{k=0}^{\infty}(1 - \mu_k[G]) = 0$, where G is the best state set B. Therefore, Theorem 2 is proved.

Based on the above theoretical analyses, it can be concluded that Lemmas 1 and 2 can be satisfied at the same time and IBA is globally convergent. □

5. Simulation Results

In this section, we will prove the superiority of the algorithm on CECE2013 benchmark function compared with other algorithms. It should be noted that the proposed algorithm is applied to solve the unconstrained optimization problem and cannot handle constrained optimization problem. Firstly, CEC2013 benchmark functions and parameter settings are introduced. After that, simulations were carried out.

5.1. CEC2013 Function and Algorithm Parameter Setting. Simulation on the CEC2013 benchmark set was done to evaluate the performance of the proposed IBA. The test set consists of three groups:

(i) F1–F5 are unimodal functions
(ii) F6–F20 are multipeak functions
(iii) F21–F28 are compound functions

The simulations were undertaken in MATLAB 2016a/Simulink (2016a, MathWorks, Natick, MA, USA).We compare IBA with the following eight algorithms:

(i) Bat algorithm (BA) [22]
(ii) Chaotic bat algorithm (CBA) [26]
(iii) Bat algorithm with Lévy distribution (LBA) [29]
(iv) Fractional Lévy flight bat algorithm (FLFBA) for global optimizations [30]
(v) Oriented cuckoo search (OCS) [5]
(vi) IBA without the integration strategy (IBA-1)
(vii) IBA without the Gaussian function (IBA-2)
(viii) IBA without the adaptive weight (IBA-3)

Table 3 shows the parameter settings for the nine algorithms according to the CEC2013 benchmark [53]. Note

Table 3: Parameter settings.

The size of bat population M	100
Frequency range	[0, 5]
Initial loudness	0.95
Initial pulse	0.9
Parameter α	0.99
Parameter γ	0.9
Location range	$[-100, 100]^D$
Number of runs	51
Problem dimension D	30
Max number of iterations N	3000

Table 4: Comparison of the average error obtained by the IBA and other algorithms.

Function	BA	CBA	LBA	FLFBA	OCS	IBA-1	IBA-2	IBA-3	IBA
F1	$1.96E+02$	$2.30E+00$	$8.25E-01$	$3.59E-01$	$4.51E-05$	$1.26E+01$	$2.29E-04$	$8.82E+01$	$\mathbf{3.26E-05}$
F2	$3.69E+06$	$4.47E+06$	$3.54E+06$	$2.21E+06$	$3.17E+05$	$3.66E+05$	$5.56E+03$	$2.21E+06$	$\mathbf{2.26E+03}$
F3	$3.44E+08$	$6.71E+08$	$4.78E+08$	$3.46E+08$	$9.96E+06$	$3.12E+07$	$6.89E+06$	$3.22E+08$	$\mathbf{5.53E+06}$
F4	$3.20E+04$	$3.10E+04$	$1.45E+04$	$6.13E+03$	$7.53E+03$	$3.11E+04$	$9.11E-01$	$1.90E+04$	$\mathbf{8.60E-02}$
F5	$5.86E-01$	$1.73E+00$	$4.74E-01$	$1.36E-01$	$4.26E-01$	$2.71E-01$	$1.43E-02$	$4.32E-01$	$\mathbf{1.31E-02}$
F6	$5.63E+01$	$6.28E+01$	$5.07E+01$	$4.52E+01$	$5.79E+01$	$4.11E+01$	$1.22E+01$	$5.01E+01$	$\mathbf{1.12E+01}$
F7	$2.16E+02$	$2.31E+02$	$1.77E+02$	$1.62E+02$	$2.07E+02$	$2.11E+02$	$2.01E+02$	$2.02E+02$	$\mathbf{1.01E+02}$
F8	$2.09E+01$	$2.10E+01$	$2.09E+01$	$2.10E+01$	$2.10E+01$	$2.09E+01$	$2.09E+01$	$2.09E+01$	$\mathbf{2.09E+01}$
F9	$3.57E+01$	$3.51E+01$	$3.40E+01$	$3.59E+01$	$2.79E+01$	$3.47E+01$	$3.03E+01$	$3.11E+01$	$\mathbf{2.65E+01}$
F10	$1.32E+00$	$1.48E+00$	$1.23E+00$	$1.05E+00$	$1.40E-01$	$1.03E+01$	$6.95E-01$	$9.89E-1$	$\mathbf{7.50E-02}$
F11	$4.07E+02$	$4.27E+02$	$1.49E+02$	$\mathbf{3.18E+01}$	$6.83E+02$	$3.78E+02$	$6.91E+01$	$5.99E+02$	$6.23E+01$
F12	$4.06E+02$	$4.30E+02$	$7.42E+02$	$7.17E+02$	$3.07E+02$	$3.77E+02$	$3.45E+02$	$3.96E+02$	$\mathbf{2.23E+02}$
F13	$4.37E+02$	$4.36E+02$	$5.59E+02$	$5.10E+02$	$3.84E+02$	$3.23E+02$	$2.89E+02$	$3.66E+02$	$\mathbf{2.26E+02}$
F14	$4.78E+03$	$2.62E+03$	$3.17E+03$	$\mathbf{1.14E+03}$	$2.39E+03$	$4.01E+03$	$4.24E+03$	$4.51E+03$	$4.23E+03$
F15	$4.89E+03$	$3.87E+03$	$4.76E+03$	$4.79E+03$	$\mathbf{3.55E+03}$	$4.87E+03$	$4.46E+03$	$4.71E+03$	$4.43E+03$
F16	$2.16E+00$	$6.06E-01$	$1.33E+00$	$1.53E+00$	$1.65E+00$	$2.11E+00$	$8.99E-01$	$2.12E+00$	$\mathbf{3.40E-01}$
F17	$8.92E+02$	$2.73E+02$	$3.36E+02$	$1.62E+02$	$\mathbf{1.61E+02}$	$8.88E+02$	$1.98E+02$	$8.56E+02$	$1.71E+02$
F18	$9.44E+02$	$2.61E+02$	$3.28E+02$	$3.31E+02$	$1.86E+02$	$7.98E+02$	$1.66E+02$	$7.76E+02$	$\mathbf{1.40E+02}$
F19	$6.07E+01$	$4.35E+01$	$1.89E+01$	$1.26E+01$	$7.38E+01$	$5.56E+01$	$523E+00$	$6.01E+01$	$\mathbf{7.02E+00}$
F20	$1.44E+01$	$1.44E+01$	$1.47E+01$	$1.48E+01$	$\mathbf{1.19E+01}$	$1.34E+01$	$1.26E+01$	$1.41E+01$	$1.25E+01$
F21	$3.38E+02$	$3.27E+02$	$3.22E+02$	$\mathbf{3.03E+02}$	$3.39E+02$	$3.38E+02$	$3.34E+02$	$3.37E+02$	$3.33E+02$
F22	$5.94E+03$	$3.15E+03$	$3.32E+03$	$\mathbf{1.20E+03}$	$2.81E+03$	$5.91E+03$	$5.61E+03$	$5.88E+03$	$5.48E+03$
F23	$5.77E+03$	$\mathbf{5.03E+03}$	$6.03E+03$	$5.81E+03$	$6.10E+03$	$5.72E+03$	$5.59E+03$	$5.73E+03$	$5.55E+03$
F24	$3.15E+02$	$\mathbf{2.92E+02}$	$3.22E+02$	$3.23E+02$	$2.97E+02$	$3.13E+02$	$2.96E+02$	$3.03E+02$	$2.93E+02$
F25	$3.49E+02$	$3.32E+02$	$3.53E+02$	$3.52E+02$	$\mathbf{3.01E+02}$	$3.32E+02$	$3.28E+02$	$3.39E+02$	$3.25E+02$
F26	$\mathbf{2.00E+02}$	$2.83E+02$	$3.54E+02$	$3.35E+02$	$2.55E+02$	$2.01E+02$	$2.03E+02$	$2.00E+02$	$2.02E+02$
F27	$1.28E+03$	$1.19E+03$	$1.33E+03$	$1.34E+03$	$1.84E+03$	$1.25E+03$	$1.20E+03$	$1.21E+03$	$\mathbf{1.12E+03}$
F28	$3.42E+03$	$2.89E+03$	$4.68E+03$	$4.34E+03$	$3.45E+03$	$3.39E+03$	$2.26E+03$	$3.34E+03$	$\mathbf{2.23E+03}$
$w/t/l$	26/1/1	22/0/6	24/1/3	23/0/5	22/0/6	25/1/2	27/1/0	26/1/1	

that the parameters of the algorithms are not optimized. As can be seen from Table 3, each algorithm is terminated when the number of iterations reaches 3,000.

In our algorithm, we used the following indicators to evaluate the experimental results:

$$\text{error} = \left| \frac{\sum_{i=1}^{51} F_i}{51} - F_{\text{value}} \right|, \tag{27}$$

where F_i is the i^{th} solution and F_{value} is the actual solution set of the benchmark function. In the following experiments, we take the average solution of each algorithm over the 51 trial runs, where the value of 51 is set according to the CEC2013 benchmark.

5.2. Comparison of the IBA with Other Algorithms. The average error in equation (27) obtained by the evaluated algorithms on different test functions is shown in Table 4. On the last line of Table 4, w denotes the number of times IBA performs better than other algorithms, t refers to the number of times IBA performs similar to the other algorithms, and l indicates the number of times IBA performs worse than the other algorithms. In addition, the best results in Table 4 are presented in bold.

As shown in Table 4, IBA outperformed other algorithms on 26, 22, 24, 23, 21, 25, 27, and 26 functions when compared with BA, CBA, LBA, FLFBA, OCS, IBA-1, IBA-2, and IBA-3, respectively. BA performers the worst. Compared with the IBA, CBA, and LBA, FLFBA has better performance on

functions F14, F21, and F22, respectively. Therefore, we conclude that the IBA can find effective solutions on most of the benchmark test functions.

Figure 3 shows the results of the convergence for different test functions. As is clear in the figure, the proposed IBA performs well in terms of convergence in most cases. However, the IBA performs poorly compared with other algorithms on functions F3, F5, F6, F7, F14, F15, and F18. This is because a suboptimal strategy was selected that made the algorithm fall into a local optimum. It can be seen clearly from Figure 3 that the convergence rate of the IBA is faster than that of OCS on most functions.

The performance of IBA-2 is superior to those of IBA-1 and IBA-3 on most functions, which indicates that the Gaussian distribution has little impact on the algorithm accuracy, but it can accelerate the convergence rate of the IBA. IBA-1 converges more slowly than IBA-3 on most functions, which indicates that the integration strategy can improve accuracy. In terms of search accuracy and convergence, the performances of IBA-1 and IBA-2 are superior to that of the BA, which means that the adaptive weight can improve the stability and search speed of the IBA. From these results, it can be concluded that the IBA has high accuracy and better convergence rate than the original BA algorithm.

The results of the Friedman test [52, 54] can be seen in Table 5. Smaller rank values indicate better performance of the algorithm. Compared with the other eight algorithms, IBA has the smallest rank. Thus, we can come to the conclusion that the IBA is the best algorithm among the nine methods.

The results of the Wilcoxon test [55] are shown in Table 6, for BA, CBA, LBA, FLFBA, OCS, IBA-1, IBA-2, and IBA-3, $p < 0.05$. As Table 6 shows, IBA performs better than the other four algorithms.

To evaluate the performance of the proposed algorithm and related algorithms at different numbers of high dimensions, scaling simulations were performed on the CEC2008 benchmark set. The parameter settings of the algorithms are the same as in Table 3. The results of IBA and the other algorithms in different dimensions are compared in Tables 7 and 8.

Figures 4 and 5 show the results of the convergence for different test functions. As is clear from the figure, the proposed IBA performs well in terms of convergence in most cases, although it performs poorly in some functions.

As it can be seen from Tables 6 and 7, as the dimensions of the functions increase, the performances of all algorithms decrease. However, IBA performs better than the other algorithms for most of the functions. IBA had better results for six functions (F1, F3, F4, F5, F6, and F7) in 100 dimensions and for six functions (F1, F2, F3, F5, F6, and F7) in 1,000 dimensions.

The above results show that the IBA is superior to the algorithms at different numbers of high dimensions.

The results of the Friedman test are listed in Table 9 for different dimensions. Compared with the other five algorithms, IBA has the smallest ranking value. Thus, we can conclude that the IBA is the best algorithm of the six methods.

The results of the Wilcoxon test are listed in Table 10 for different dimensions. As can be seen in Table 10, IBA performs better than the other five algorithms.

5.3. Comparison on Two Real-World Application Problems. To test the feasibility and performance of IBA on real-world applications, we chose two problems from the real world [56]: the design of a gear train [57] and parameter estimation for frequency-modulated (FM) sound waves [58]. We compared the IBA with the following ten algorithms:

(i) Social learning PSO (SLPSO) [56]
(ii) Adaptive PSO (APSO) [59]
(iii) Comprehensive learning PSO (CLPSO) [60]
(iv) Cooperative approach to PSO (CPSOH) [61]
(v) Fully informed particle swarm (FIPS) [62]
(vi) Supervised PSO (SPSO) [63]
(vii) Adaptive differential evolution with optional external archive (JADE) [64]
(viii) Global and local real-coded genetic algorithm (HRCGA) [65]
(ix) Frankenstein's PSO (FPSO) [66]
(x) Restart CMA evolution strategy with increasing population size (G-CMA-ES) [67]

The first problem is to optimize the gear ratio for a compound gear train that contains three gears. The function of the problem is as follows:

$$f(x) = \left(\frac{1}{6.931} - \frac{x_1 x_2}{x_2 x_3}\right)^2, \tag{28}$$

where $x_j \in [12, 60]$, $j = 1, 2, 3, 4$.

The second problem is to estimate the parameters of an FM synthesizer. The components of the parameters vector are as follows: $X = \{a_1, w_1, a_2, w_2, a_3, w_3\}$, and the expressions for the estimated target sound waves are shown as follows:

$$\begin{aligned} y(t) = {} & a_1 \cdot \sin(w_1 \cdot t \cdot \theta) + a_2 \cdot \sin(w_2 \cdot t \cdot \theta) \\ & + a_3 \cdot \sin(w_3 \cdot t \cdot \theta), \end{aligned} \tag{29}$$

$$\begin{aligned} y_0(t) = {} & ((1.0) \cdot \sin 5.0 \cdot t \cdot \theta - (1.5) \cdot \sin(4.8) \cdot t \cdot \theta \\ & + (2.0) \cdot \sin(4.9) \cdot t \cdot \theta), \end{aligned} \tag{30}$$

where $\theta = 2\pi/100$, and the parameters are defined in the range $[-6.4, 6.35]$. The fitness function is the sum of the squared errors between the estimated wave in equations (29) and (30) as follows:

$$f(\vec{X}) = \sum_{t=100}^{100} (y(t) - y_0(t))^2. \tag{31}$$

A comparison of the performances on two real-world problems is listed in Table 11. A set of heuristic algorithms were used for the comparison. The parameter settings of IBA are the same as those listed in Table 3. The IBA and the

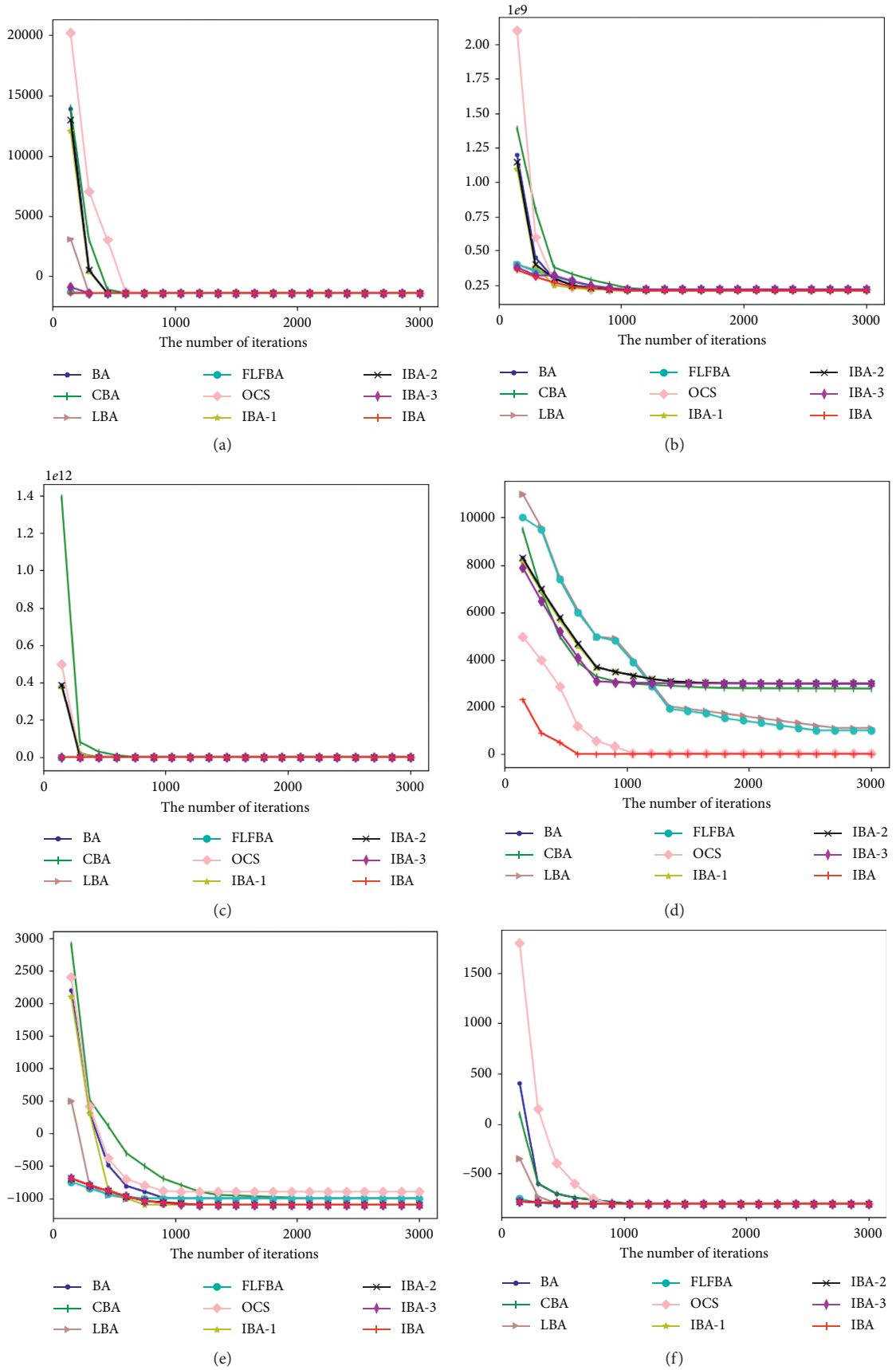

Figure 3: Continued.

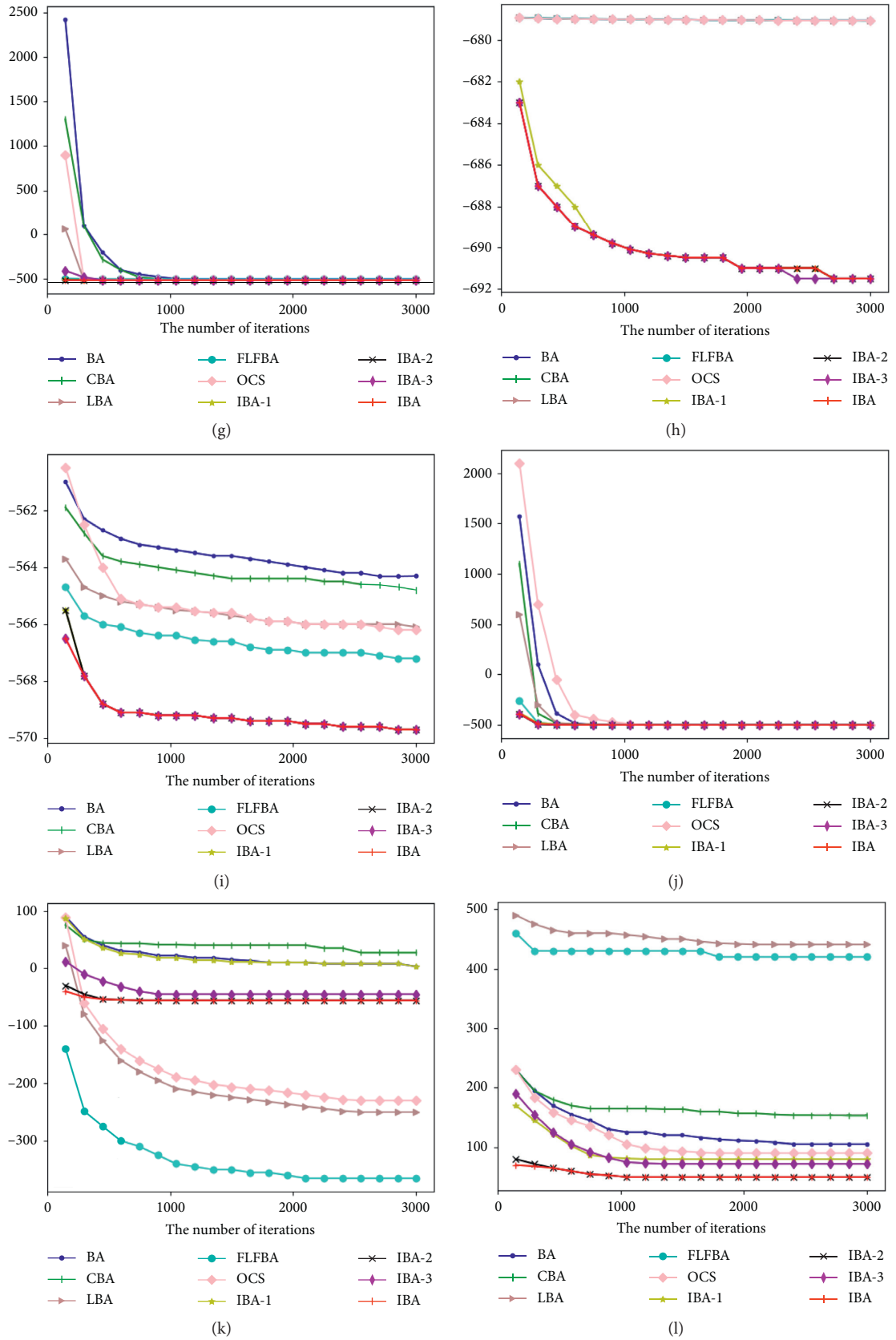

Figure 3: Continued.

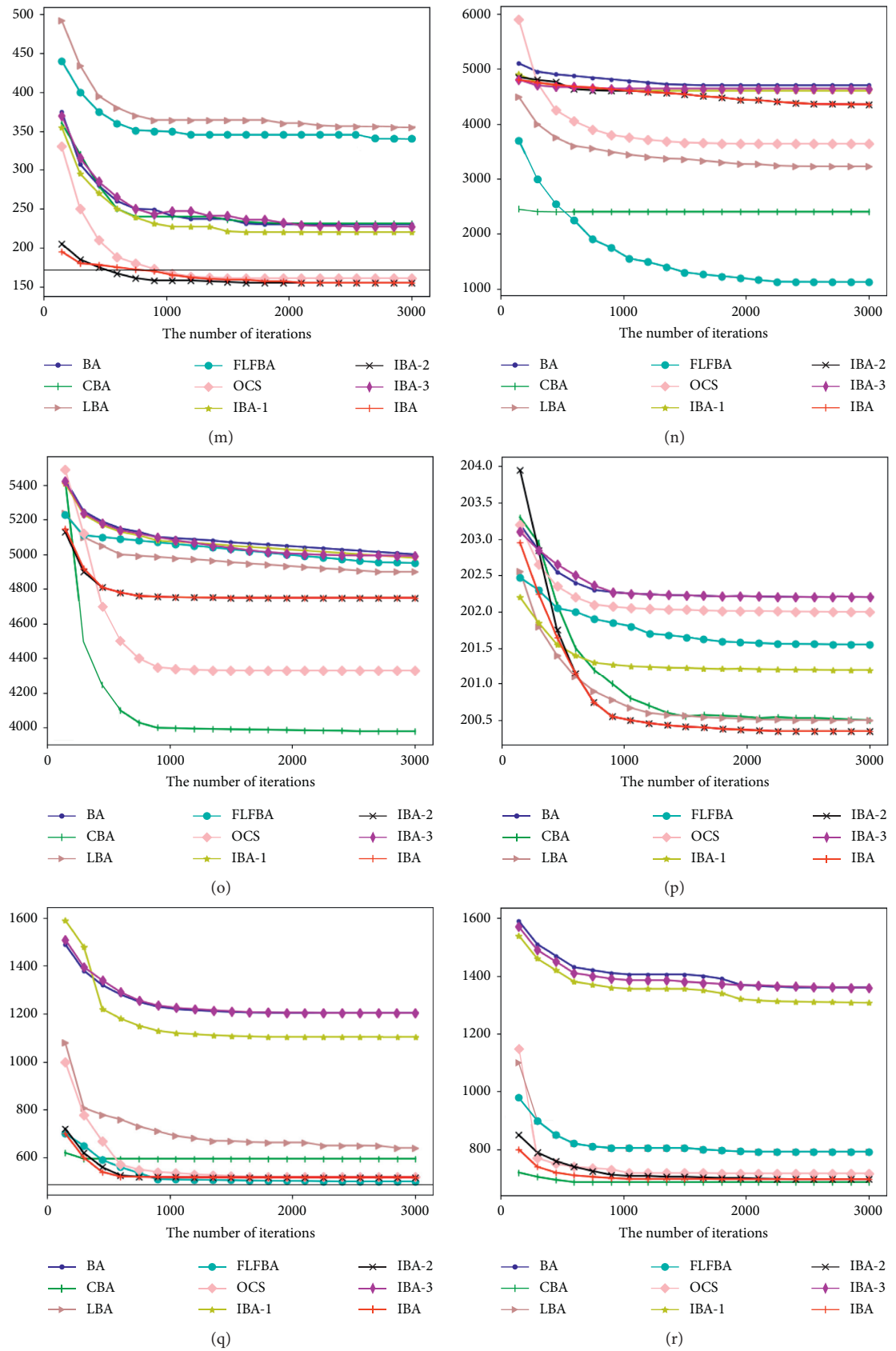

Figure 3: Continued.

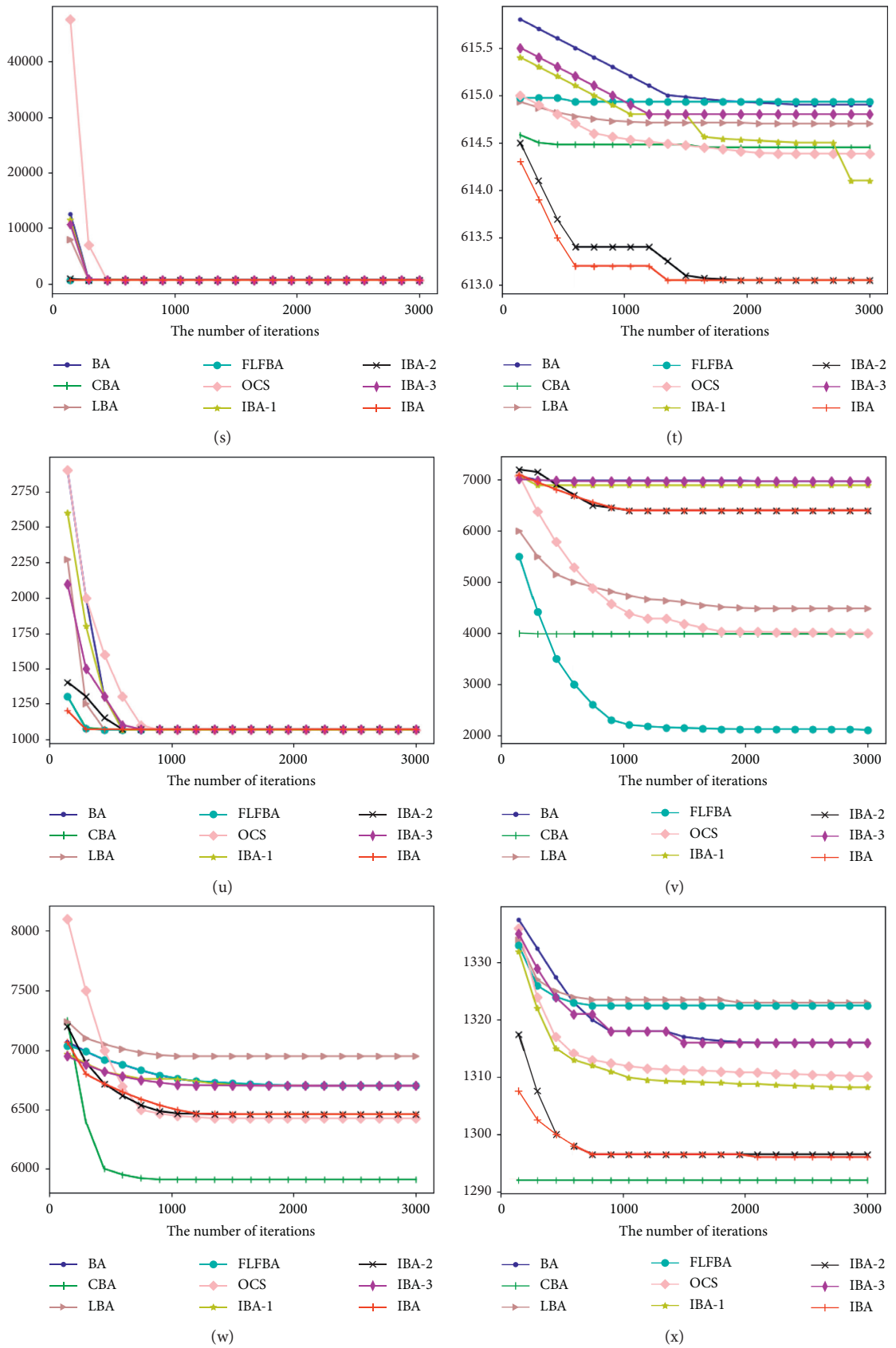

Figure 3: Continued.

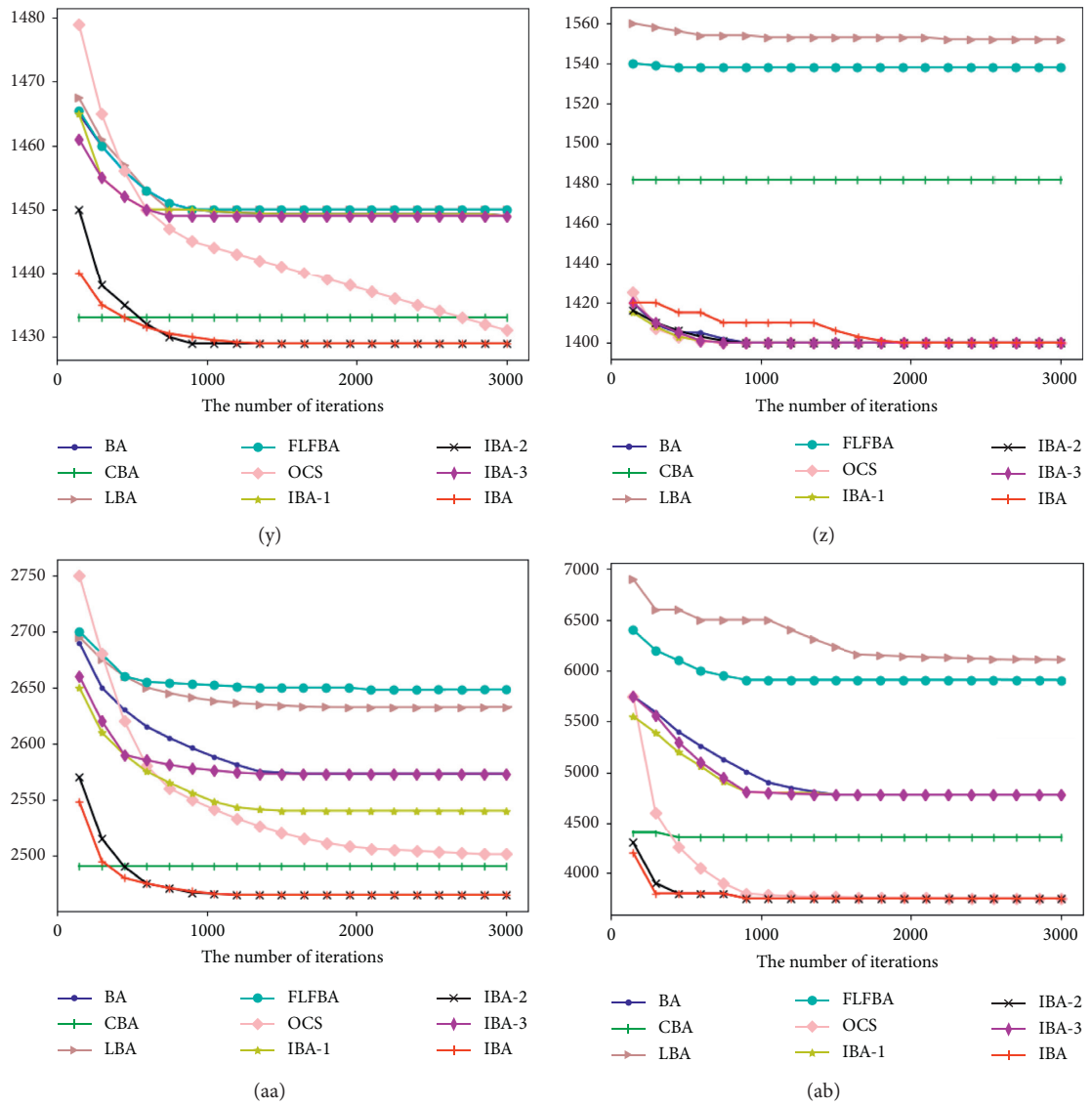

Figure 3: Convergence curves of different algorithms on the CEC2013 benchmark set. (a) F1, (b) F2, (c) F3, (d) F4, (e) F5, (f) F6, (g) F7, (h) F8, (i) F9, (j) F10, (k) F11, (l) F12, (m) F13, (n) F14, (o) F15, (p) F16, (q) F17, (r) F18, (s) F19, (t) F20, (u) F21, (v) F22, (w) F23, (x) F24, (y) F25, (z) F26, (aa) F27, and (ab) F28.

Table 5: Friedman test results of nine algorithms.

Algorithm	Rank
BA	7.18
CBA	5.43
LBA	6.05
FLFBA	5.34
OCS	4.57
IBA-1	5.55
IBA-2	3.45
IBA-3	5.48
IBA	1.95

Table 6: Wilcoxon test of five algorithms.

IBA VS	p value
BA	0
CBA	0.023
LBA	0.001
FLFBA	0.006
OCS	0.016
IBA-1	0
IBA-2	0
IBA-3	0

related algorithms were 30 times for a crucial analysis, where "Min," "Max," "Mean," and "Std" denote the best, worst, mean, and standard deviation values, respectively.

As can be seen from Table 11, IBA can easily solve the first real-world problem and obtained the best values. For the second one, the proposed IBA can find the optimal solution and obtained the smallest standard deviation values, which shows that IBA has good ability to find the global

Table 7: Comparison of the average error obtained with IBA and the other algorithms on the CEC2008 benchmark (100 dimensions).

Function	BA	CBA	LBA	FLBA	OCS	IBA
1	$6.79E+00$	$6.55E+02$	$1.04E+01$	$5.66E+00$	$1.55E+01$	$\mathbf{1.56E-13}$
2	$8.21E+01$	$7.66E+01$	$\mathbf{5.36E+01}$	$8.45E+01$	$6.15E+01$	$6.12E+01$
3	$3.59E+03$	$3.51E+03$	$6.07E+03$	$6.33E+03$	$1.71E+06$	$\mathbf{2.26E+02}$
4	$1.51E+03$	$7.85E+02$	$1.06E+03$	$1.13E+03$	$7.27E+02$	$\mathbf{5.11E+02}$
5	$1.42E+03$	$4.57E+02$	$1.25E-01$	$8.88E+00$	$2.12E+00$	$\mathbf{5.11E-03}$
6	$2.07E+01$	$1.49E+01$	$3.5E+01$	$1.78E+01$	$1.95E+01$	$\mathbf{5.02E+00}$
7	$-8.53E+02$	$-8.55E+02$	$-8.96E+02$	$-9.32E+02$	$-1.06E+03$	$\mathbf{-1.28E+03}$
$w/t/l$	7/0/0	7/0/0	6/0/1	7/0/0	7/0/0	

Table 8: Comparison of the average error obtained with IBA and the other algorithms on the CEC2008 benchmark (1000 dimensions).

Function	BA	CBA	LBA	FLBA	OCS	IBA
1	$1.13E+06$	$1.86E+06$	$5.76E+03$	$1.56E+05$	$5.66E+01$	$\mathbf{7.72E-07}$
2	$1.42E+02$	$1.23E+02$	$1.28E+02$	$1.45E+02$	$1.25E+02$	$\mathbf{1.11E+01}$
3	$1.55E+11$	$3.11E+11$	$5.25E+07$	$4.87E+09$	$3.56E+06$	$\mathbf{3.61E+03}$
4	$1.99E+04$	$9.86E+03$	$1.77E+04$	$1.69E+04$	$\mathbf{7.25E+03}$	$8.12E+03$
5	$3.62E+04$	$1.55E+04$	$4.51E+02$	$7.33E+03$	$3.26E+03$	$\mathbf{3.47E-03}$
6	$2.41E+01$	$1.99E+01$	$2.11E+01$	$2.33E+01$	$2.36E+01$	$\mathbf{1.89E+01}$
7	$-6.63E+03$	$-7.55E+03$	$-7.12E+03$	$-7.29E+03$	$-7.56E+03$	$\mathbf{-1.28E+04}$
$w/t/l$	7/0/0	7/0/0	7/0/0	7/0/0	6/0/1	

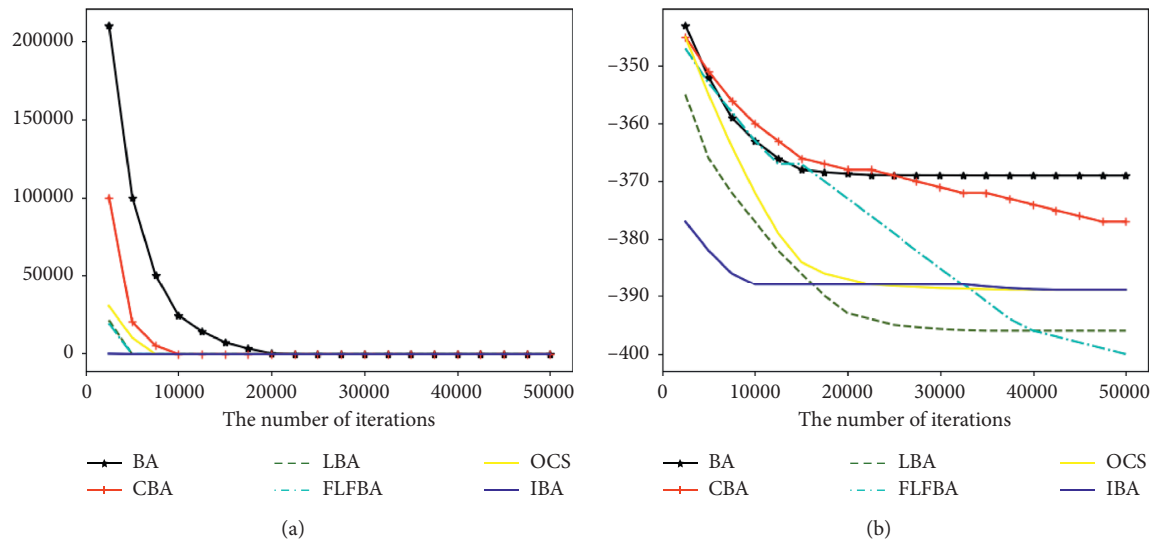

Figure 4: Continued.

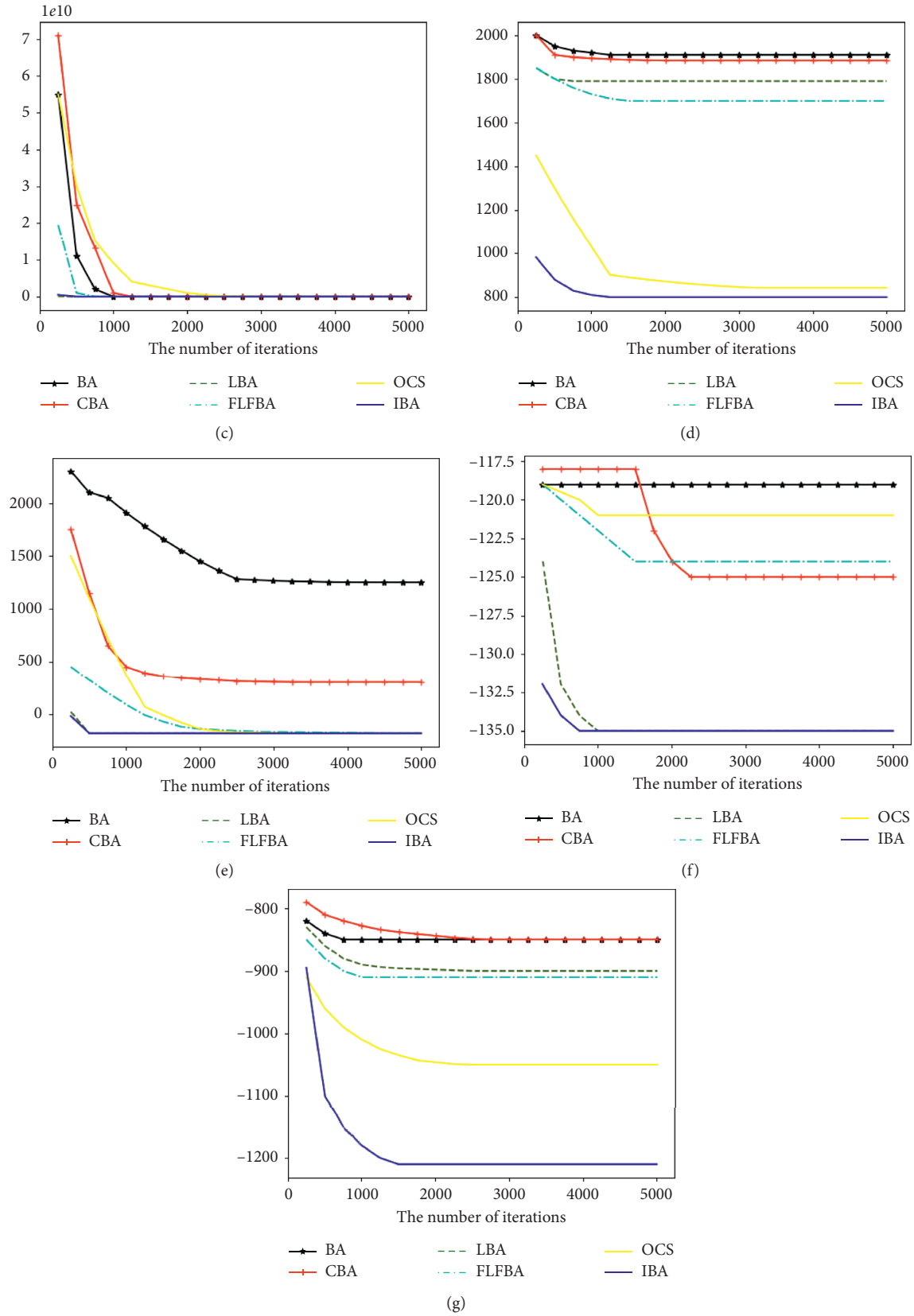

Figure 4: Convergence curves of different algorithms on the CEC2008 benchmark (100 dimensions): (a) F1, (b) F2, (c) F3, (d) F4, (e) F5, (f) F6, and (g) F7.

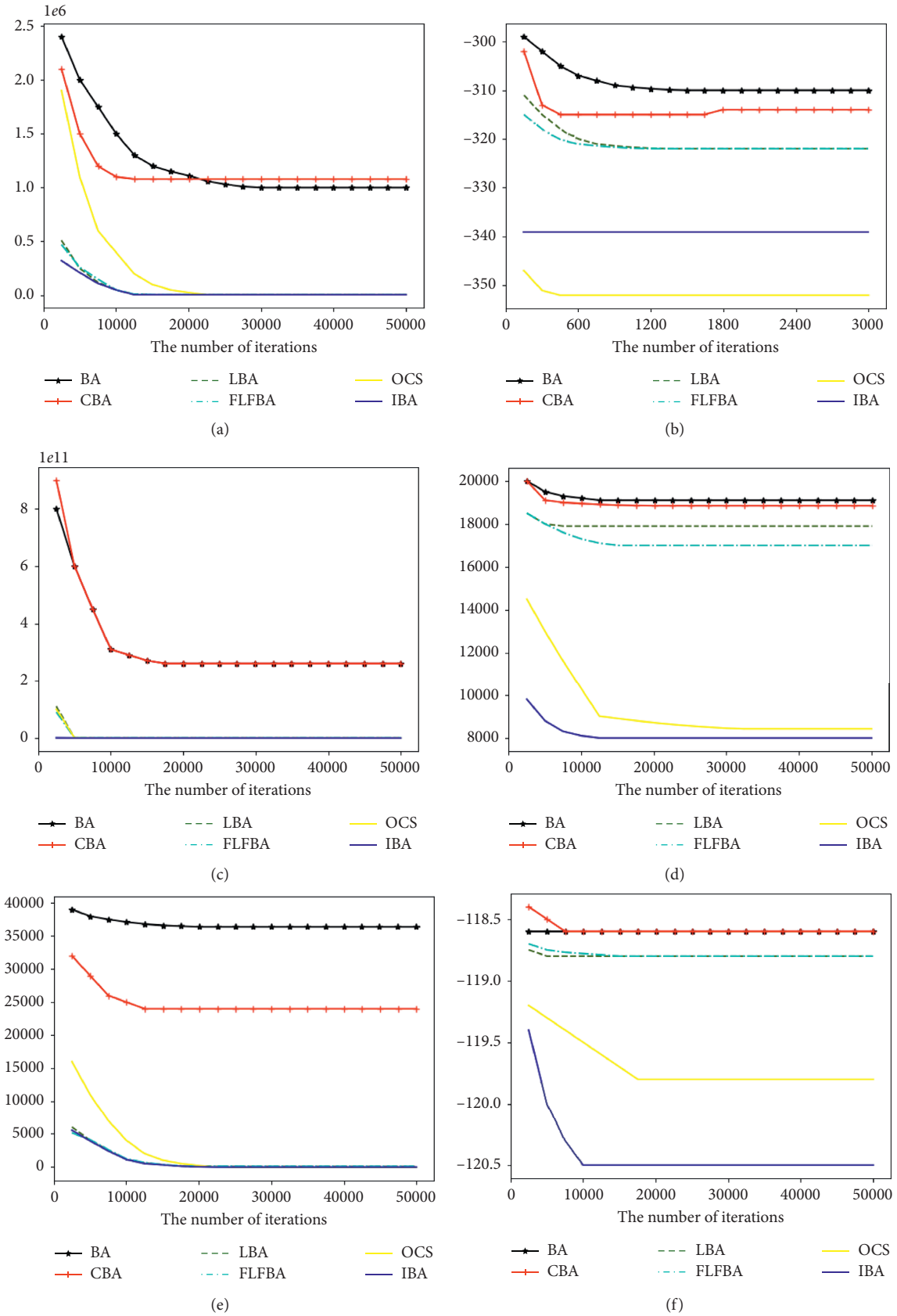

Figure 5: Continued.

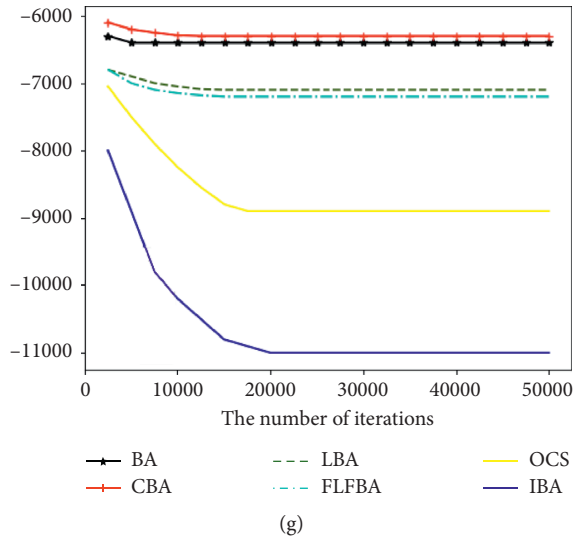

Figure 5: Convergence curves of different algorithms on the CEC2008 benchmark (1,000 dimensions): (a) F1, (b) F2, (c) F3, (d) F4, (e) F6, (f) F7, and (g) F7.

Table 9: Friedman test results of six algorithms.

	100 dimensions	1000 dimensions
Algorithms	Rankings	Rankings
BA	4.86	5.57
CBA	3.86	4.14
LBA	3.57	3.86
FLBA	4.00	4.43
OCS	3.57	1.86
IBA	**1.14**	**1.14**

Table 10: Wilcoxon test of six algorithms.

	100 dimensions	1000 dimensions
IBA vs	p value	p value
BA	0.018	0.008
CBA	0.018	0.008
LBA	0.043	0.008
FLBA	0.018	0.018
OCS	0.018	0.091

Table 11: Comparison on two real-world problems.

	Gear ratio				Estimation error			
IBA	$\mathbf{7.71E-32}$	$\mathbf{2.65E-24}$	$\mathbf{2.24E-25}$	$\mathbf{6.26E-25}$	**0**	24.76	17.62	**5.62**
SLPSO	$2.70E-12$	$6.19E-09$	$2.22E-09$	$9.83E-09$	**0**	**13.79**	4.18	26.99
APSO	$2.70E-12$	$1.31E-08$	$1.59E-09$	$1.44E-08$	**0**	34.22	11.33	41.13
CLPSO	$2.70E-12$	$1.36E-09$	$1.99E-10$	$2.22E-09$	**0**	14.08	**3.82**	23.53
CPSOH	$1.54E-10$	$2.02E-06$	$2.80E-07$	$2.33E-06$	0.01	42.52	27.08	60.61
FIPS	$8.88E-10$	$8.90E-07$	$3.27E-08$	$8.77E-07$	3.45	15.11	5.93	25.75
SPSO	$2.70E-12$	$2.56E-07$	$1.39E-08$	$2.57E-07$	**0**	18.27	9.88	33.85
JADE	$2.70E-12$	$1.36E-09$	$2.10E-10$	$2.26E-09$	**0**	13.92	7.55	26.18
HRCGA	$2.70E-12$	$1.18E-09$	$1.53E-10$	$1.88E-09$	**0**	17.59	8.41	32.54
FPSO	$2.06E-09$	$1.70E-04$	$1.57E-05$	$1.80E-04$	**0**	15.82	5.22	28.31
G-CMA-ES	$2.70E-12$	$7.32E-01$	$2.44E-02$	$1.31E-01$	3.326	55.09	38.75	16.77

optimum when dealing with real-world problems with good stability and is stable.

6. Summary

Although heuristic algorithms perform well in optimization problems, it is easy for them to fall into local optima and output unstable results. To solve these problems, the IBA was proposed to improve the performance of the BA. The convergence of the algorithm was proved by mathematical analysis and simulation experiments. In the future, we will use the IBA to solve modeling, optimization, and control problems in different fields◦ In particular, we will further improve our algorithm to solve the constrained optimization problem.

Data Availability

The code used in this paper is released, which was written in MATLAB, has been publicly available at https://github.com/yzbyhjq/iba.git.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors are grateful to the support of the National Natural Science Foundation of China (61373004).

References

[1] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," *Computer Methods in Applied Mechanics & Engineering*, vol. 191, pp. 1245–1287, 2002.

[2] J. H. Holland, *Genetic algorithms*, Scientific American, London, UK, 1992.

[3] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Cybernetics*, vol. 26, no. 1, pp. 29–41, 1996.

[4] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Mhs95 Sixth International Symposium on Micro Machine & Human Science*, Nagoya, Japan, October 2002.

[5] Z. Cui, B. Sun, G. Wang, Y. Xue, and J. Chen, "A novel oriented cuckoo search algorithm to improve DV-hop performance for cyber-physical systems," *Journal of Parallel & Distributed Computing*, vol. 103, pp. 42–52, 2016.

[6] W. Wu, H. Ouyang, A. W. Mohamed, C. Zhang, and S. Li, "Enhanced harmony search algorithm with circular region perturbation for global optimization problems," *Applied Intelligence*, vol. 50, no. 3, pp. 951–975, 2020.

[7] S. Gupta and K. Deep, "A novel random walk grey wolf optimizer," *Swarm & Evolutionary Computation*, vol. 44, pp. 101–112, 2018.

[8] Z. Lei, S. Gao, S. Gupta, J. Cheng, and G. Yang, "An aggregative learning gravitational search algorithm with self-adaptive gravitational constants," *Expert Systems with Applications*, vol. 152, 2020.

[9] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 7, pp. 1501–1529, 2020.

[10] S. Gupta, K. Deep, and A. P. Engelbrecht, "A memory guided sine cosine algorithm for global optimization," *Engineering Applications of Artificial Intelligence*, vol. 93, Article ID 103718, 2020.

[11] S. Gupta and K. Deep, "A hybrid self-adaptive sine cosine algorithm with opposition based learning," *Expert Systems with Applications*, vol. 119, 2019.

[12] S. Gupta and K. Deep, "Improved sine cosine algorithm with crossover scheme for global optimization," *Knowledge-Based Systems*, vol. 165, 2018.

[13] S. Gupta and K. Deep, "Hybrid sine cosine artificial bee colony algorithm for global optimization and image segmentation," *Neural Computing & Applications*, vol. 32, 2019.

[14] S. Gupta and K. Deep, "A memory-based grey wolf optimizer for global optimization tasks," *Applied Soft Computing*, vol. 93, Article ID 106367, 2020.

[15] X. Han, L. Yue, Y. Dong, Q. Xu, and X. Xu, "Efficient hybrid algorithm based on moth search and fireworks algorithm for solving numerical and constrained engineering optimization problems," *The Journal of Supercomputing*, vol. 76, pp. 1–26, 2020.

[16] A. Baykasoğlu, A. Hamzadayi, and S. Akpinar, "Single seekers society (SSS): bringing together heuristic optimization algorithms for solving complex problems," *Knowledge-Based Systems*, vol. 165, 2018.

[17] S. Shadravan, H. R. Naji, and V. K. Bardsiri, "The sailfish optimizer: a novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 80, pp. 20–34, 2019.

[18] S. Kaur, L. K. Awasthi, and A. L. Sangal, "Hmoshssa: a hybrid meta-heuristic approach for solving constrained optimization problems," *Engineering with Computers*, vol. 2, pp. 1–37, 2020.

[19] M. Montemurro, A. Vincenti, and P. Vannucci, "The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 256, pp. 70–87, 2013.

[20] M. Montemurro, A. Vincenti, and P. Vannucci, "Design of the elastic properties of laminates with a minimum number of plies," *Mechanics of Composite Materials*, vol. 48, no. 4, pp. 369–390, 2012.

[21] G. Costa, M. Montemurro, and J. Pailhès, "A general hybrid optimization strategy for curve fitting in the non-uniform rational basis spline framework," *Journal of Optimization Theory & Applications*, vol. 176, 2018.

[22] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," *Nature Inspired Cooperative Strategies for Optimization*, vol. 284, pp. 65–74, 2010.

[23] T. C. Bora, L. D. S. Coelho, and L. Lebensztajn, "Bat-inspired optimization approach for the brushless DC wheel motor problem," *IEEE Transactions on Magnetics*, vol. 48, no. 2, pp. 947–950, 2012.

[24] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013.

[25] A. L. Tamiru and F. M. Hashim, "Application of bat algorithm and fuzzy systems to model exergy changes in a gas turbine,"

Artificial Intelligence, Evolutionary Computing and Metaheuristics, Springer, Berlin, Germany, 2013.

[26] A. H. Gandomi and X.-S. Yang, "Chaotic bat algorithm," *Journal of Computational Science*, vol. 5, no. 2, pp. 224–232, 2014.

[27] J. Xie, Y. Q. Zhou, and H. Chen, "A bat algorithm based on lévy flights trajectory," *Pattern Recognition & Artificial Intelligence*, vol. 26, pp. 829–837, 2013.

[28] C. Gan, W. Cao, M. Wu et al., "A new bat algorithm based on iterative local search and stochastic inertia weight," *Expert Systems with Application*, vol. 104, 2018.

[29] C. Liu and Y. E. Chunming, "Bat algorithm with the characteristics of lévy flights," *Caai Transactions on Intelligent Systems*, vol. 8, pp. 240–246, 2013.

[30] R. Boudjemaa, F. Ouaar, and D. Oliva, "Fractional lévy flight bat algorithm for global optimisation," *International Journal of Bio-Inspired Computation*, vol. 15, no. 2, p. 100, 2020.

[31] I. Fister, D. Fister, and X.-S. Yang, "A hybrid bat algorithm," *Elektrotehniski Vestnik*, vol. 80, no. 1, 2013.

[32] M. A. Al-Betar and M. A. Awadallah, "Island bat algorithm for optimization," *Expert Systems with Applications*, vol. 107, pp. 126–145, 2018.

[33] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, *Multi-population Cooperative Bat Algorithm-Based Optimization of Artificial Neural Network Model*, Elsevier Science Inc., Amsterdam, Netherlands, 2015.

[34] W. Ghanem and A. Jantan, "An enhanced bat algorithm with mutation operator for numerical optimization problems," *Neural Computing & Applications*, vol. 31, pp. 1–35, 2017.

[35] T. K. Dao, T. S. Pan, T. T. Nguyen, and J. S. Pan, "Parallel bat algorithm for optimizing makespan in job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 29, pp. 1–12, 2015.

[36] E. Osaba, X. S. Yang, I. Fister, J. Del Ser, P. Lopez-Garcia, and A. J. Vazquez-Pardavila, "A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection," *Swarm & Evolutionary Computation*, vol. 44, 2018.

[37] C. A. Leke and T. Marwala, "Missing data estimation using bat algorithm," *Deep Learning and Missing Data in Engineering Systems*, Springer, Berlin, Germany, 2019.

[38] R. Cheruku, D. R. Edla, V. Kuppili, and R. Dharavath, "RST-BatMiner: a fuzzy rule miner integrating rough set feature selection and bat optimization for detection of diabetes disease," *Applied Soft Computing*, vol. 67, 2017.

[39] R. Y. M. Nakamura, L. A. M. Pereira, K. A. Costa, D. Rodrigues, J. P. Papa, and X. S. Yang, "BBA: a binary bat algorithm for feature selection," in *Proceedings of the 25th SIBGRAPI Conference on Graphics*, Ouro Preto, Brazil, August 2012.

[40] S. Goyal and M. S. Patterh, "Modified bat algorithm for localization of wireless sensor network," *Wireless Personal Communications*, vol. 86, no. 2, pp. 657–670, 2016.

[41] G. T. Pulido and C. A. C. Coello, "A constraint-handling mechanism for particle swarm optimization," in *Proceedings of the 2004 Congress on Evolutionary Computation*, Portland, OR, USA, June 2004.

[42] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the International Conference on Evolutionary Programming*, San Diego, CA, USA, March 1998.

[43] G. R. Cai, S. L. Chen, S. Z. Li, and W. Z. Guo, "Study on the nonlinear strategy of inertia weight in particle swarm optimization," in *Proceedings of the International Conference on Natural Computation*, Shandong, China, October 2008.

[44] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1749-1750, Québec Canada, July 2009.

[45] X.-S. He, W.-J. Ding, and X.-S. Yang, "Bat algorithm based on simulated annealing and Gaussian perturbations," *Neural Computing & Applications*, vol. 25, pp. 459–468, 2013.

[46] K. Chellapilla, "Combining mutation operators in evolutionary programming," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 3, pp. 91–96, 1998.

[47] Y. Xin, L. Yong, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[48] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, August 2002.

[49] B. Azizipanah-Abarghooee and R. Azizipanahabarghooee, "Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 56, pp. 42–54, 2014.

[50] W. Y. Szeto, Y. Wu, and S. C. Ho, "An artificial bee colony algorithm for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 215, no. 1, pp. 126–135, 2011.

[51] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.

[52] J. H. Friedman, "Fast sparse regression and classification," *International Journal of Forecasting*, vol. 28, no. 3, pp. 722–738, 2012.

[53] J. Liang, B. Qu, P. Suganthan, and A. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Technical Report, Zhengzhou University, Zhengzhou China, 2013.

[54] S. Yilmaz and E. U. Kucuksille, "A new modification approach on bat algorithm for solving optimization problems," *Applied Soft Computing*, vol. 28, pp. 259–275, 2014.

[55] S. Hui, K. Wang, Z. Jia, and Y. Xiang, "Artificial bee colony algorithm with improved special centre," *International Journal of Computing Science & Mathematics*, vol. 7, pp. 548–553, 2016.

[56] C. Li, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems Man & Cybernetics Part B*, vol. 42, pp. 627–646, 2012.

[57] E. Sandgren, "Nonlinear nteger and discrete programming in mechanical design optimization," *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990.

[58] S. Das and P. N. Suganthan, "Problem definitions and evaluation criteria for CEC 2011 competition on testing evolutionary algorithms on real world optimization problems," Technical Report, Jadavpur University, Kolkata, India, 2011.

[59] Z. H. Zhan and J. Zhang, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 6, 2008.

[60] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[61] F. V. D. Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, 2004.

[62] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[63] B. Daniel and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, Honolulu, HI, USA, June 2007.

[64] J. Zhang and A. Anderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 945–958, 2009.

[65] C. García-Martínez, M. Lozano, F. Herrera, D. Molina, and A. M. Sánchez, "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1088–1113, 2008.

[66] D. O. Montes, M. A. T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 1120–1132, 2009.

[67] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proceedings IEEE Congress on Evolutionary Computation*, Edinburgh, UK, September 2005.