Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2016, Article ID 9820294, 13 pages
http://dx.doi.org/10.1155/2016/9820294

Research Article

A New Modified Artificial Bee Colony Algorithm with Exponential Function Adaptive Steps

Wei Mao,[1] Heng-you Lan,[1,2] and Hao-ru Li[3]

[1]*Department of Mathematics, Sichuan University of Science & Engineering, Zigong, Sichuan 643000, China*
[2]*Key Laboratory of Higher Education of Sichuan Province for Enterprise Informationalization and Internet of Things, Zigong, Sichuan 643000, China*
[3]*School of Automation and Electronic Information, Sichuan University of Science & Engineering, Zigong, Sichuan 643000, China*

Correspondence should be addressed to Heng-you Lan; hengyoulan@163.com

Received 26 July 2015; Revised 10 October 2015; Accepted 19 October 2015

Academic Editor: Yufeng Zheng

As one of the most recent popular swarm intelligence techniques, artificial bee colony algorithm is poor at exploitation and has some defects such as slow search speed, poor population diversity, the stagnation in the working process, and being trapped into the local optimal solution. The purpose of this paper is to develop a new modified artificial bee colony algorithm in view of the initial population structure, subpopulation groups, step updating, and population elimination. Further, depending on opposition-based learning theory and the new modified algorithms, an improved S-type grouping method is proposed and the original way of roulette wheel selection is substituted through sensitivity-pheromone way. Then, an adaptive step with exponential functions is designed for replacing the original random step. Finally, based on the new test function versions CEC13, six benchmark functions with the dimensions $D = 20$ and $D = 40$ are chosen and applied in the experiments for analyzing and comparing the iteration speed and accuracy of the new modified algorithms. The experimental results show that the new modified algorithm has faster and more stable searching and can quickly increase poor population diversity and bring out the global optimal solutions.

1. Introduction

It is well known that algorithms for solving various characteristics optimization problems can be classified into different groups, such as population-based algorithms, stochastic algorithms, deterministic algorithms, and iterative algorithms. An algorithm is called population-based [1] if one works with a group of solutions and tries to improve them. Two important classes of population-based optimization algorithms are exactly evolutionary algorithms and swarm intelligence-based algorithms [2]. Swarm intelligence is an innovative artificial intelligence technique with collective behavior of self-organized systems [3]. Since many swarm intelligence algorithms, such as genetic algorithm (GA) [4], particle swarm optimization (PSO) [5], ant colony optimization (ACO) [6], and biogeography-based optimization [7], have simplicity, ease of implementation, outstanding performance, and other advantages [8], they have shown great success in solving some nonconvex, discontinuous, or nondifferentiable optimization problems. However, these intelligence algorithms are sensitive to value and precision. Thus, inspired by the behavior of honey bees, Karaboga [9] introduced basic artificial bee colony algorithm (ABC) in 2005 and constituted one of the most prominent approaches in the field of bee-inspired algorithms. Further, in consideration of the solution reaching speed, the success rate, and the performance rate, El-Abd [10] provided a complete performance assessment of ABC and compared it with the widely known differential evolution (DE), GA, heuristic algorithms, PSO, and other foraging algorithms (e.g., bacterial algorithm, ACO, and bacterial foraging optimization) by using the well-known benchmark functions in [11].

It was claimed that ABC is the most successful algorithm for multimodal and hybrid functions [12]. This is because ABC has no demand to the objective function, constraint, and external information and is only based on fitness function in the search process [13]. Further, ABC has the following advantages. (i) The mechanism of multiple roles:

using different methods, bees adjust quality of the solutions spontaneously, so as to adapt to the next search process [14]. (ii) The cooperative working mechanism: according to the information from other bees, bees decide whether to find the optimal solution with larger probability [15]. (iii) The strong robustness: the search rules are not certain but are probabilistic and have excellent robustness and a wide range of applicability [16]. (iv) The stability: even if the individual fails, the entire swarm can still complete the task [17]. (v) Less control parameters, simple operation, and ease of implementation [18]: indeed, ABC has been shown to be very competitive with respect to other state-of-the-art foraging and evolutionary algorithms.

However, there exists still an insufficiency to ABC because ABC does well in exploration but is poor at exploitation. As for the improvement and development of ABC, Karaboga and Gorkemli [19] proposed a new update rule for onlooker bees in the hive to improve the local search and convergence characteristics of ABC. Inspired by PSO, Imanian et al. [20] changed the update rule of basic ABC to increase the convergence speed for solving high dimensional and continuous optimization problems. Wang et al. [21] proposed multistrategy ensemble artificial bee colony algorithm (MEABC) to improve the local and global search capability of basic ABC and tested the performance of MEABC by using basic, shifted, and rotated benchmark functions. Gao et al. [22] developed new search equations to adjust exploration and exploitation capability of ABC. In a different approach for ABC, Das et al. [23] proposed a learning routine based on fitness and proximity stimuli and tested the method with standard benchmark functions. Zang et al. [24] designed a logarithmic function adaptive step instead of the original random step.

Moreover, in dealing with some complex problems by applying ABC, there are some defects such as slow search speed, poor population diversity, stagnation in the working process, and trapping into the local optimal solution [13]. Recently, ABC has been extended and improved by many researches. But since ABC is relatively new, the researches in the literatures lack systematicness and are scattered. See, for example, [1–3, 12–30] and the references therein. In 2012, Li et al. [31] pointed out that "ABC has no mechanism to use the global information in the search space, so it easily results in a waste of computing power and gets trapped in local optima". Further, the authors proposed a novel algorithm (named as DEABC, i.e., differential evolution artificial bee colony algorithm), which synthesizes DE and ABC and enhances individuals by sharing information between DE population and bee colony. For related works, one can see [1, 30] and the references therein.

Motivated and inspired by the above works, a new modified artificial bee colony (MABC) algorithm shall be constructed based on adaptive step with exponential functions. By using opposition-based learning theory and an improved *S*-type grouping method, the initial population for MABC will be given, and the original way of roulette wheel selection shall be substituted by sensitivity-pheromone way. Specifically, an adaptive step with exponential functions will be designed to replace the original random step. In order to verify the validity of the improved algorithm, MABC, it is compared with DEABC [30], novel artificial bee colony algorithm (NABC) [8], and ABC; the experiment results tested with six well-known benchmark optimization functions, which are chosen from new test function versions CEC13, show that MABC is superior to ABC, NABC, and DEABC.

The rest of this paper is organized as follows. Section 2 gives a brief introduction to ABC. The new MABC is presented and analyzed in Section 3. Section 4 presents and discusses the experimental results of six benchmark functions with the dimensions $D = 20$ and $D = 40$, respectively. Finally, the conclusion is drawn in Section 5.

2. Brief Review to ABC

In this section, a brief review on ABC is going to be given.

2.1. Thoughts of the Algorithm. A honey bee colony can successfully discover the highest quality food sources in nature. Hence, the idea of ABC comes from intelligent foraging behavior of honey bees to finding good solutions for solving optimization problems. In a general way, according to the ways of searching food, the colony of bees is divided into three kinds: employed bees, onlooker bees, and scout bees. The employed bees are responsible for exploiting the nectar sources. They explore the beforehand food source position and give the quality information of the food to the onlooker bees. The onlooker bees wait in the hive and decide to exploit a food source based on the information shared by the employed bees. In order to find a new nectar source, the scout bees randomly search environment either depending on an internal motivation or based on possible external clues [32]. The position of a nectar source implies a possible solution of the optimization problems, and the profitability of a nectar source corresponds to the quality (fitness) of the possible solution. Each nectar source is exploited by only one employed bee. In other words, the number of nectar sources equals the number of employed bees or onlooker bees [25]. In this process, the employed bees maintain good solution, the onlooker bees improve convergence speed, and the scout bees enhance the ability to remove local optimum [26, 31].

2.2. ABC Iteration Steps. From [24, 33], it follows that main iteration steps of ABC can be listed as follows.

Step 1 (initialization). Randomly generate N solutions (i.e., food sources) $\{x_1, x_2, \ldots, x_N\}$ as initial population in a D dimension searching space, where N is the number of food sources, which equals the half of the colony size, $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ is a D-dimensional solution vector and the ith food source in the initial population for $i = 1, 2, \ldots, N$, and D also denotes the number of optimization parameters.

Step 2 (renewing population). In the stage of collecting honey, each employed bee produces a new nectar source within the neighborhood of the food source. After comparing the new nectar source with the old ones, the high probability

will be memorized. Next, in the stage of follow, every onlooker bee evaluates the profitability of nectar sources taken from all employed bees and then chooses a food source at a certain probability. As in the case of the employed bees, she produces a modification on the source position in her memory and keeps the better nectar source. The regeneration of nectar sources in these two stages is based on the following formula:

$$v_{ij} = x_{ij} + \text{rand} \cdot \left(x_{ij} - x_{kj}\right), \tag{1}$$

where $i, k = 1, 2, \ldots, N$, $j = 1, 2, \ldots, D$, and rand $\in [0, 1]$ is a random number, which controls the generation range of neighborhood of x_{ij}. With the search being close to optimal solution, the range of neighborhood will become smaller and smaller.

Step 3 (nectar source selection). In the stage of follow, the onlooker bees choose food source through comparing the probability which is computed by the fitness value. The nectar sources of high probability are selected in large probability. And the probability of being selected for the food sources is calculated as follows:

$$P_i = \frac{\text{Fit}_i}{\sum_{i=1}^{N} \text{Fit}_i}, \tag{2}$$

where Fit_i ($i = 1, 2, \ldots, N$) is the fitness (profitability) value of the ith solution, which is obtained by the following equation:

$$\text{Fit}_i = \begin{cases} \dfrac{1}{1 + f_i}, & \text{if } f_i \geq 0, \\ 1 + \left|f_i\right|, & \text{if } f_i < 0, \end{cases} \tag{3}$$

where f_i is the objective function value for $i = 1, 2, \ldots, N$, which is specific for the optimization problem. If the new food source position has a quality equal to or better than the old one, then the old food source position is replaced by the new one. Otherwise, the old one is retained, which is the same as the employed bees stage.

Step 4 (population elimination). If a solution has not been improved significantly through a predetermined number of trials, called "max iteration," then the solution is regarded as falling into a local optimal solution and their original position will be abandoned. Thus, the corresponding employed bees will become scout bees and a new solution instead of the eliminated solution is randomly generated, which can be expressed as follows:

$$x_{ij} = \text{rand} \cdot \left(x_{\max j} - x_{\min j}\right) + x_{\min j}, \tag{4}$$

where i, j are the same as in (1) and $x_{\max j}$ and $x_{\min j}$ denote the jth individual maximum and the ith individual minimum values, respectively.

Based on the above iteration steps, the process of ABC [12] can be shown in Algorithm 1.

3. New MABC

From the above introductions, it follows that ABC has many advantages, such as simplicity, ease of implementation, and outstanding performance. However, the algorithm steps indicate that ABC has the following obvious flaws:

(i) The randomly generated initial population will lead the solution to be random distribution in the solution space. Hence, the searching ability of ABC will be affected directly.

(ii) All individuals begin to search directly in the whole solution space, which will reduce searching efficiency.

(iii) The population is regenerated by random step length. When the algorithm performs in the neighborhood of optimal solutions, the searching range will be severely restricted. Thus, the convergence rate and optimization precision of ABC will be influenced.

In order to overcome the existing deficient problem of ABC and related algorithms, the purpose of this paper is to introduce and study a new MABC and to improve the initial population structure, the population grouping, and the population regeneration of ABC.

3.1. Opposition-Based Learning of Population Initialization. ABC is relatively sensitive to the initial population construction. Therefore, the search range of ABC will be restricted and the global search ability of ABC will also be influenced when the initial population distribution in the search space or the local area is random. However, the initial population constructed by using opposition-based learning method has the diversity as much as possible, is better representative, and meets the accuracy of experimental requirements (see, e.g., [6, 13, 34, 35] and the references therein).

Based on the opposition-based learning method, the initial population for MABC will be constructed in this paper. Firstly, an initial solution is given, and next the corresponding reverse initial solution to the initial solution is produced. Then, the two kinds of solutions are sorted. Finally, the better fitness solution is chosen as the initial population. This method will help to enhance the efficiency of MABC and to improve the quality of solutions [34]. The processes can be listed as shown in Algorithm 2.

3.2. S-Type Subpopulation Grouping. Intelligent optimization algorithms commonly use general grouping method as follows. Firstly, the population fitness function value of each individual is calculated. Secondly, the fitness function value is arranged in descending order. Then, the entire ordered population is divided into M subpopulations, and the process of the grouping [24] is as shown in Table 1. However, this grouping method still causes some problems: the relative advantage of individuals is divided into the same species and their disadvantage is concentrated in other uncertain groups. This phenomenon will make very large differences between subpopulations, and this grouping method is not conducive to the expansion of population diversity.

Input: N, max iteration
Output: Best solution
(1) Initialize the food sources using the following equation
$x_{ij} = \text{rand} \cdot (ub_j - lb_j) + lb_j$ for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, D$,
where ub_j and lb_j are upper and lower boundaries of parameter j, respectively
(2) Evaluate the food sources using (3)
(3) Iter = 1, where Iter denotes iteration number
(4) While Iter ≤ max iteration do
(5) Send employed bees to produce new solutions as in (1)
(6) Evaluate the new solutions and apply greedy selection
(7) Calculate the probability using (2)
(8) Send onlooker bees to produce new greedy selection
(9) Evaluate the new solutions and apply greedy selection
(10) Send one scout into the search area for discovering a new food source
(11) Memorize the best solution found so far
(12) Iter = Iter + 1
(13) end while
(14) Return the best solution

Algorithm 1: Flowchart of ABC.

Initialize the population size N.
{The random initialization phase}
for $i = 1$ to N do
for $j = 1$ to D do

$$x_{ij} = \text{rand} \cdot (x_{\max j} - x_{\min j}) + x_{\min j}, \qquad (*)$$

end for
end for
{The opposition-based learning phase}
for $i = 1$ to N do
for $j = 1$ to D do

$$ox_{ij} = x_{\max j} + x_{\min j} - x_{ij}, \qquad (**)$$

end for
end for
where i, j, $x_{\max j}$ and $x_{\min j}$ are the same as in (4). Choosing the best adaptive value N of individual from $\{x_{ij} \cup ox_{ij}\}$ as initial population.

Algorithm 2

To reduce the difference between each group (see [24]), a new grouping method is presented, which is called S-type subpopulation grouping. Similarly, the fitness function value is arranged in descending order. But, relative to the general grouping method, the order of the value in the new groups is different, and the process of new grouping is listed in Table 2.

Indeed, first of all, the fitness function values of each individual of population are calculated, and the fitness function values in descending order are ranked. Next, the entire population is divided into M subpopulations $S_1, S_2, \ldots, S_M$. Let Y_i be the ith individual for $i = 1, 2, \ldots, N$. If the solutions due to S-type subpopulation grouping are odd rounds and there are a large number subpopulations, then there will be more differences between each group. Thus, the final round can be abandoned through S-type subpopulation grouping. Now one can see that S-type subpopulation grouping can overcome the disequilibrium between groups, making the subpopulation have a little difference, which is good to the expansion of population diversity.

Furthermore, a new population will be recomposed by all subpopulations when each subpopulation group is less than max iteration N of intragroup. And then the fitness function values of each population are recalculated and are arranged according to the descending order. Only one of the individuals of equivalent fitness function values is kept. Let the number of remaining individuals at last be K. Then the population can be regenerated as follows:

$$\begin{aligned} &\text{the top } M \text{ individuals are retained,} && \text{if } K \geq M, \\ &M\text{-}K \text{ individuals will randomly be initialized and filled into the population,} && \text{if } K < M. \end{aligned} \qquad (5)$$

Table 1: General grouping process.

Solutions	Subpopulations			
	S_1	S_2	$\cdots$	S_M
The first round	Y_1	Y_2	$\cdots$	Y_M
The second round	Y_{M+1}	Y_{M+2}	$\cdots$	Y_{2M}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2: S-type subpopulation grouping.

Solutions	Subpopulations			
	S_1	S_2	$\cdots$	S_M
The first round	Y_1	Y_2	$\cdots$	Y_M
The second round	Y_{2M}	Y_{2M-1}	$\cdots$	Y_{M+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
The final round	Y_{2nM}	Y_{2nM-1}	$\cdots$	$Y_{(2n-1)M+1}$
	$\vdots$	Y_{2nM+K}	$\cdots$	Y_{2nM+1}

3.3. Sensitivity-Pheromone Option. Since the onlooker bees chosen by roulette way are too greedy and the population diversity is deduced too quickly, it is easily plunged into local optimum by using ABC for solving function optimization problems. In applications, the roulette way is practically affected by the population diversity.

Thus, it is an important idea for sensitivity to replace the roulette way. Sensitivity is put forward by free search algorithm [35] with pheromone (related to optimization of adaptive value) to selected area, which can be expressed as follows:

(i) Calculate the N values of fitness to the individual $f(x)$.

(ii) For each $i = 1, 2, \ldots, N$, calculate the ith pheromone

$$p(i) = \frac{a \cdot f(i)}{f_{\max}} + (1 - a), \quad a = \frac{f_{\max}}{\sum_{i=1}^{N} f_i}, \tag{6}$$

where $f_{\max}$ stands for the maximum values of $f(i)$ for $i = 1, 2, \ldots, N$.

(iii) Generate the jth individual sensitivity randomly: $S(j) \sim U(0, 1)$.

(iv) Find out sensitivity area i matching individual of the first j and randomly find i to meet $p(i) \geq S(i)$.

By the above model, it is easy to see the following: (i) since the sensitivity-pheromone is random, any area can be searched in theory, and the local optimum can largely be avoided by using the model. (ii) The fourth step of the model is to make MABC have a clear direction, and convergence or divergence of the objective function in the search space can be determined quickly. This method is similar to the roulette way of nectar source for the onlooker bees, and so sensitivity and pheromone option instead of roulette way can be applied.

3.4. Updating Step Length with Exponential Function. According to solution search equation (1) of ABC, the new candidate solution is generated through moving the old one towards another selected random solution of the population. This method can increase the probability of locating global optimal solution, but it cannot guarantee obtaining a better solution. Hence, the convergence speed is slow. If each new candidate solution is better than the old one, then convergence speed will become faster. However, if only the better solution is considered, then the algorithm could get trapped into local optimum [32]. In ABC, the updating step length denoted by σ (i.e., rand in (1)) plays a very important role and controls the ability to search (local or global) and the convergence rate. On the other hand, if the value of σ is too large, then the algorithm is easy to lose the global optimal solution. And, on the contrary, the algorithm is easy to show the phenomenon of stagnation and fall into local optimum [33]. Therefore, it is very necessary to design a new suitable step length associated with the performance of the algorithm.

The following exponential function is used in this paper to update step length:

$$\sigma = (\text{rand} - 0.5) \cdot 2^{\text{rand}}, \tag{7}$$

and the improved population moving step length is

$$\Delta = \sigma \cdot \left(x_{ij} - x_{kj}\right) = (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{ij} - x_{kj}\right), \tag{8}$$

for $i, k = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, D$, where rand is the same as in (1). Thus, the location of the updated solution is

$$\begin{aligned} v_{ij} &= x_{ij} + \sigma \cdot \left(x_{ij} - x_{kj}\right) \\ &= x_{ij} + (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{ij} - x_{kj}\right) \end{aligned} \tag{9}$$

and the location of the random generated solutions is

$$x_{ij} = x_{\min j} + (\text{rand} - 0.5) \cdot 2^{\text{rand}} \cdot \left(x_{\max j} - x_{\min j}\right), \tag{10}$$

where $x_{\max j}$ and $x_{\min i}$ are the same as in $(*)$.

Now one can know that with the increase of evolution algebra, the updating step will be increased and become larger and larger. In the early stage of the evolution, the improved evolution algebra can be conducted a comprehensive local search, and make the algorithm better position to the target area. In the middle and later periods of the evolution, the improved evolution algebra will be carried out global search and can speed up the convergence rate so as to reach to optimal solution.

3.5. Flowchart of MABC. To clearly know MABC, the detailed flowchart of MABC is shown in Algorithm 3, where the max iteration as the termination criteria is adopted the same as in Algorithm 1, and CS denotes the number of colony size, which is equal to the sum of the numbers of employed bees and onlooker bees.

(1) Initialization phase
(1.1) Food Number = CS/2, that is, the number of food sources equals to half of the colony size
(1.2) Based on the opposition-based learning of population initialization as in Section 3.1
(1.3) Calculate their fitness values
(1.4) Based on *S*-type sub-population grouping described in Section 3.2
(1.5) trial = zeros (1, Food Number). Initialize trial counters to be zeros
(1.6) Initialize elitists to be food sources
(1.7) Memorize the best food source
Iter = 0;
while Iter ≤ max iteration
  Select a better elitist by sensitivity-pheromone option in Section 3.3
  (2) Employed bees phase
    for i = 1 to Food Number
      (2.1) Update step length of each food source by (7) and generate a new solution by (10)
      (2.2) Evaluate the new solution
      (2.3) Iter = Iter + 1
      (2.4) A sensitivity-pheromone selection is applied between the current solution i and its mutant.
        If the mutant solution is better than the current solution i, replace the solution
        with the mutant and reset the trial counter of solution i. Otherwise, increase
        its trial counter.
    end
  Calculate probabilities using (6)
  (3) Onlooker bees phase
    (3.1) Select a food source by sensitivity-pheromone method in Section 3.3
    (3.2) For each onlooker bee, update step length of each food source by (7) and generate a new solution by (10)
    (3.3) Iter = Iter + 1
    (3.4) A sensitivity-pheromone selection is applied between the current solution and its mutant.
      If the mutant solution is better than the current solution, replace the solution
      with the mutant and reset the trial counter of solution.
      Otherwise, increase its trial counter.
  The optimal solution is memorized
  (4) Scout bees phase
    (4.1) Determine the food sources whose trial counter exceeds the "limit" value,
      and only one scout bee is allowed to occur in each iteration
    (4.2) Use a scout bee to carry out randomly search
  if Food Number ≥ max iteration
end while
Return *the optimal solution*

Algorithm 3: Flowchart of MABC.

4. Simulation and Analysis

In this section, the performance and accuracy of MABC, ABC, NABC, and DEABC will be analyzed and compared through the figures, the total run time, mean value, and standard deviation of six benchmark functions from the new test function versions CEC13, respectively.

4.1. Convergence Performance Analysis. In order to verify the validity of MABC, six benchmark functions listed in Table 3 (see [11, 27]) are used in the experiments. According to their characteristics, these six functions are divided into two groups. The Sphere, SumSquares, and Schwefel 2.22 are unimodal functions, and Ackley, Griewank, and Rastrigin are multimodal functions. For each benchmark function, two dimension sizes, 20 and 40, of solution space are tested. The colony size is 80 and the max iteration is 1500; limit is 100. Each of the experiments is repeated 10 times.

In order to further compare the performance of four proposed algorithms, convergence performance of the six functions for ABC, DEABC, NABC, and MABC is plotted to dimension sizes 20 and 40, respectively. See Figures 1–6.

From the convergence performances of SumSquares, Griewank, and Rastrigin functions, one can see that when $D = 20$, ABC converges with the slowest speed and locates a local optimal solution. DEABC has the fastest speed at the beginning and goes into the global optimal solution. NABC also goes into the global optimal solution, but the speed is slower than those of DEABC and MABC. Further, MABC has the fastest speed at the end and is the first one to attain the true global number. When $D = 40$, ABC and DEABC fall into the global solution. NABC has the fastest speed at the beginning and attains the global optimal solution at the end, but the true global number of NABC is smaller than that of MABC. MABC goes into the global optimal solution at the end, and the speed is faster than that of NABC.

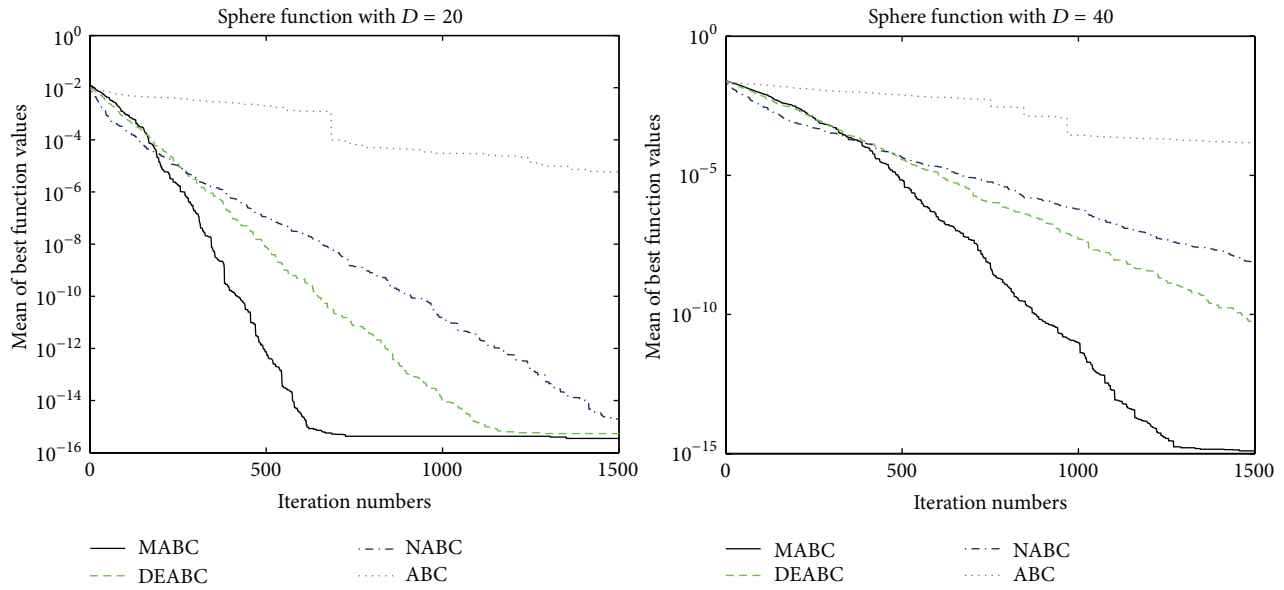

Figure 1: Convergence performance of ABC, DEABC, NABC, and MABC.

Table 3: Numerical benchmark functions.

Name	Formula	Search domain	Minimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
SumSquares	$f_2(x) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
Schwefel 2.22	$f_3(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-100, 100]^D$	0
Griewank	$f_4(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
Rastrigin	$f_5(x) = \sum_{i=1}^{D} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^D$	0
Ackley	$f_6(x) = -20e^{-0.2\sqrt{(\sum_{i=1}^{D} x_i^2)/D}} - e^{(\sum_{i=1}^{D} \cos(2\pi x_i))/D} + 20 + e$	$[-32, 32]^D$	0

The convergence performance of Sphere function exhibits that various behavior for both dimension sizes. When $D = 20$, ABC converges with the slowest speed, and NABC and DEABC have fastest speed in the first place. Later on, MABC is the first to arrive to the global optimal solution, the second one is DEABC, and the third one is NABC. And ABC locates a local optimal solution. When $D = 40$, ABC also goes into the local optimal. NABC, DEABC, and MABC have the similar speed originally; hereafter MABC has quicker speed than NABC and DEABC. Finally, these three algorithms come to the global optimal solution, but the best one is MABC, the second one is DEABC, and the last one is NABC.

The convergence performances of Schwefel 2.22 and Ackley functions demonstrate that, for both dimension sizes of 20 and 40, ABC, NABC, and DEABC all fall into the global optimal solution, but DEABC converges with the fastest speed originally. Later on, MABC has the fastest speed and locates the global optimal solution at the end.

From Figures 1–6, it can be comprehensively deduced that ABC converges with the slowest speed and tends to a local optimal solution. DEABC and NABC have faster speed than ABC but sometimes enter into local optimal solution. MABC has fastest speed at the end and achieves the really global optimal solution.

4.2. Comparison of Mean Value and Standard Deviation. The results of comparing the mean value and standard deviation are shown in Table 4, where the first column shows the benchmark functions, the "D" column represents the dimension of function, the "Mean" column contains the average best values of benchmark functions, and the "SD" column shows the standard deviation of the best value.

For six benchmark functions, one can know that when $D = 20$, the mean values of ABC, NABC, DEABC, and MABC are better than those of them when $D = 40$. For SumSquares, Griewank, and Rastrigin functions, it is

Table 4: Comparison of the mean value and standard deviation.

Function	D	ABC		NABC	
		Mean	SD	Mean	SD
Sphere	20	$9.12643e-4$	$6.80695e-4$	$8.51338e-16$	$2.29525e-16$
	40	0.445472	0.234793	$3.7323e-11$	$1.50446e-12$
SumSquares	20	$1.3749e-5$	$2.33632e-5$	$8.75587e-16$	$2.30537e-16$
	40	0.994557	1.70504	$4.65058e-10$	$2.77676e-10$
Schwefel 2.22	20	$9.42221e-4$	$7.46032e-4$	$2.22277e-8$	$1.00025e-8$
	40	0.0461286	0.0287431	$2.15524e-4$	$6.03173e-5$
Griewank	20	$2.5264e-6$	$1.43918e-6$	$4.44089e-16$	$1.92296e-16$
	40	0.0125412	0.00802599	$1.07636e-12$	$5.37341e-13$
Rastrigin	20	0.0331619	0.0574246	$7.75912e-14$	$3.28186e-14$
	40	15.5119	20.1043	$6.00988e-9$	$2.57688e-9$
Ackley	20	$4.22652e-3$	$3.9353e-3$	$7.97212e-10$	$4.51507e-10$
	40	0.603485	0.0057116	$3.77068e-6$	$7.14915e-7$
Function	D	DEABC		MABC	
		Mean	SD	Mean	SD
Sphere	20	$5.33457e-16$	$6.65399e-18$	$4.21651e-16$	$8.89001e-18$
	40	$5.54516e-13$	$3.68742e-13$	$1.49866e-15$	$2.47044e-16$
SumSquares	20	$6.03584e-16$	$1.3174e-16$	$4.16975e-16$	$1.01262e-16$
	40	$5.20275e-10$	$6.16741e-7$	$3.29277e-15$	$2.28728e-15$
Schwefel 2.22	20	$3.61252e-6$	$1.24222e-6$	$8.79872e-16$	$1.09679e-16$
	40	$1.08779e-4$	$1.65357e-4$	$4.12959e-10$	$1.80491e-10$
Griewank	20	$3.70074e-16$	$2.31111e-16$	$1.11022e-16$	0
	40	$6.54372e-11$	$8.03242e-11$	$8.14164e-16$	$2.34132e-16$
Rastrigin	20	$2.84217e-14$	$2.84217e-14$	$9.4739e-15$	$1.64093e-14$
	40	$4.40958e-7$	$3.26047e-7$	$1.89478e-13$	$3.28186e-14$
Ackley	20	$1.21723e-6$	$1.197e-6$	$3.16784e-14$	$2.05116e-15$
	40	$7.67309e-7$	$5.03943e-7$	$5.01809e-9$	$1.85715e-9$

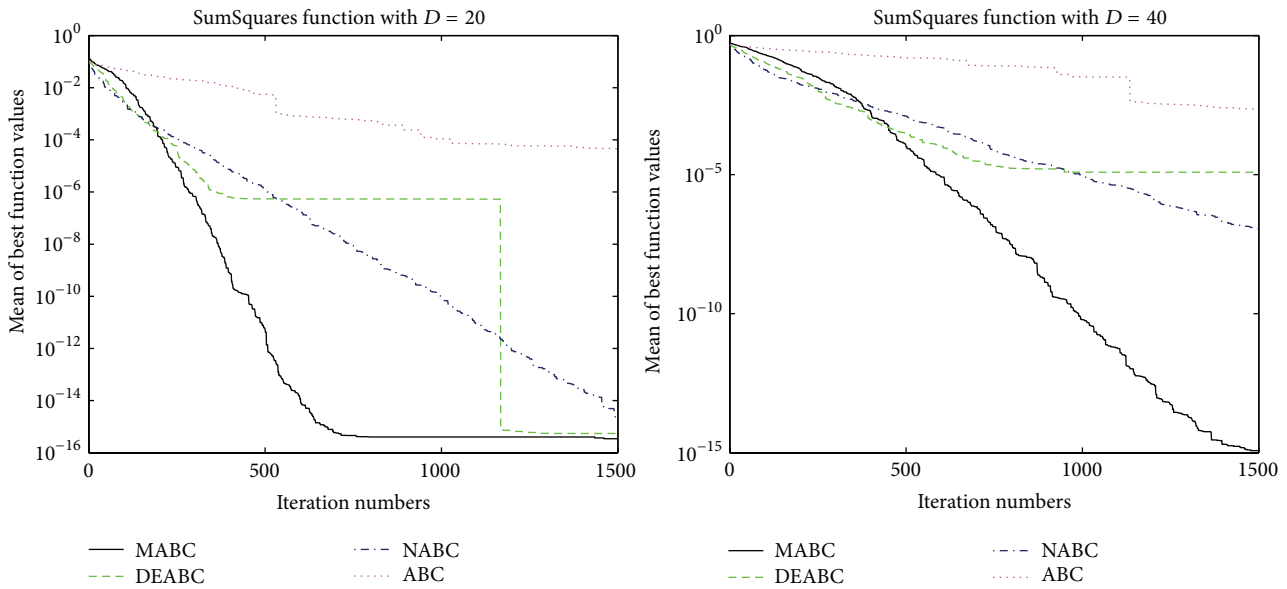

Figure 2: Convergence performance of ABC, DEABC, NABC, and MABC.

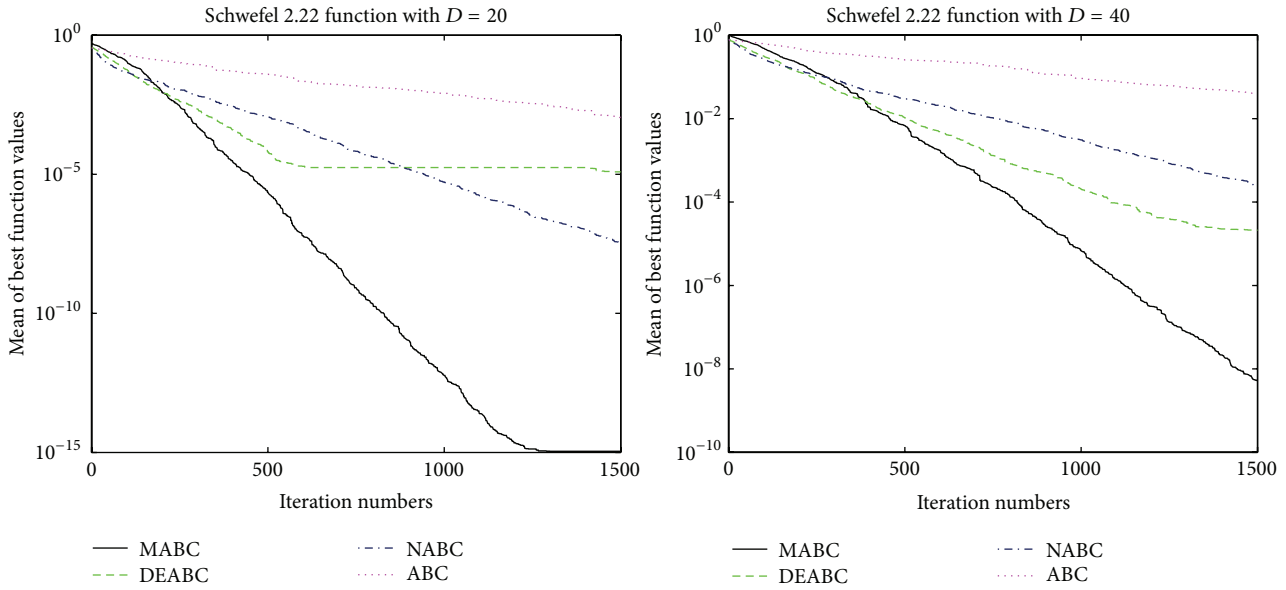

Figure 3: Convergence performance of ABC, DEABC, NABC, and MABC.

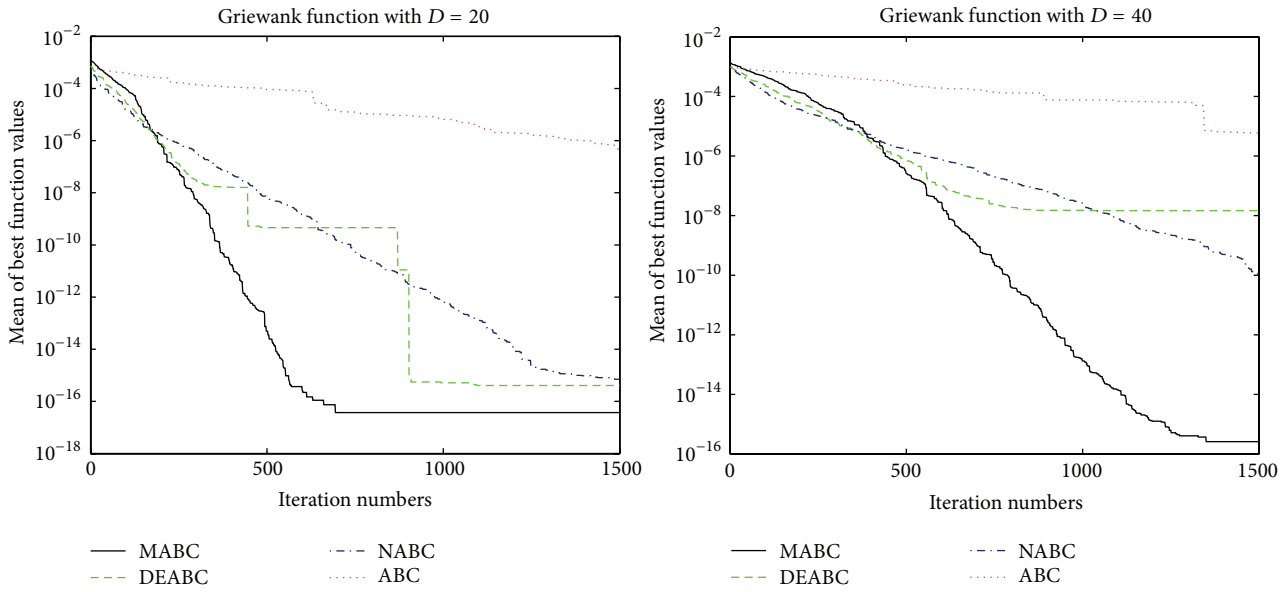

Figure 4: Convergence performance of ABC, DEABC, NABC, and MABC.

observed that when $D = 40$, the mean value of NABC has better mean than that of DEABC. On the other hand, in terms of standard deviation, ABC and NABC are less than DEABC for Sphere and Ackley functions. In addition, MABC has better average best values than DEABC, and DEABC has better average best values than ABC and NABC. The least standard deviation of MABC to six test functions with the dimensions $D = 20$ and $D = 40$ is $8.89001e - 18$, and the standard deviations of MABC are always less than DEABC; meanwhile the standard deviation of DEABC is less than ABC and NABC.

Hence, by the results of six test functions with the dimensions $D = 20$ and $D = 40$, MABC have better results than ABC, NABC, and DEABC. For all test functions, the mean and standard deviation of MABC results are several orders of magnitude better than ABC, NABC, and DEABC, particularly in case of SumSquares, Schwefel 2.22, and Ackley functions. Therefore, MABC has more accurate solution and better stability than ABC, NABC, and DEABC.

4.3. Comparison of Total Run Time. This subsection will show effectively that the searching speed of MABC is fastest through the phenomenon for total run time of solutions (see Figure 7). To all the functions, when $D = 20$ or 40, NABC costs the more time, ABC and DEABC follow closely, MABC

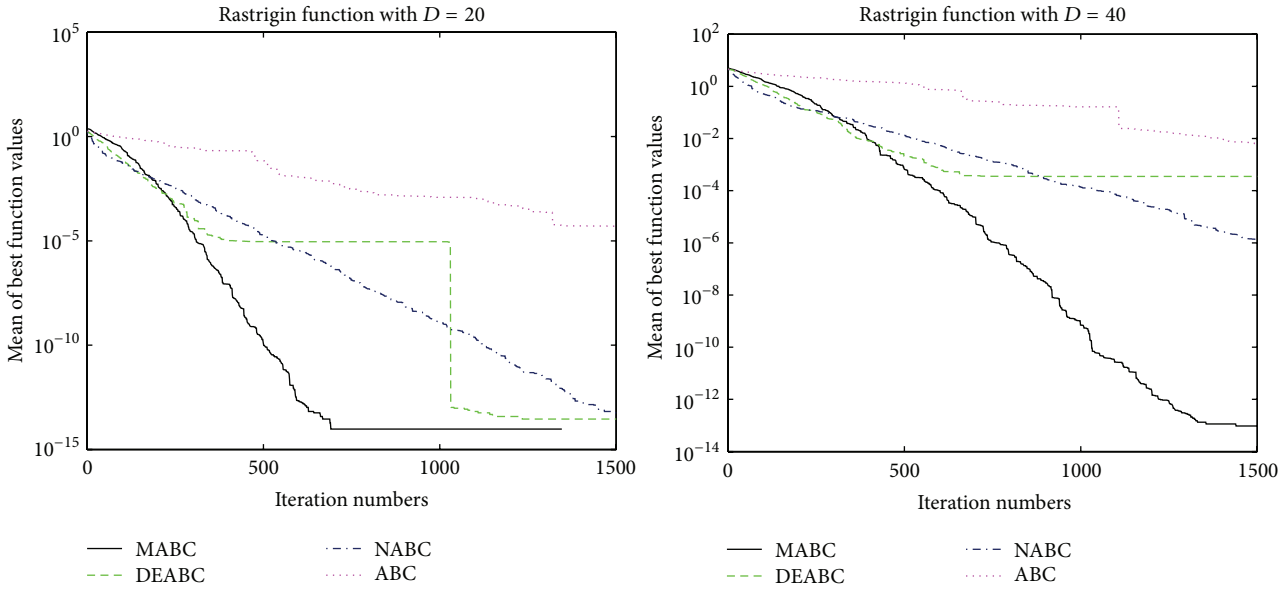

Figure 5: Convergence performance of ABC, DEABC, NABC, and MABC.

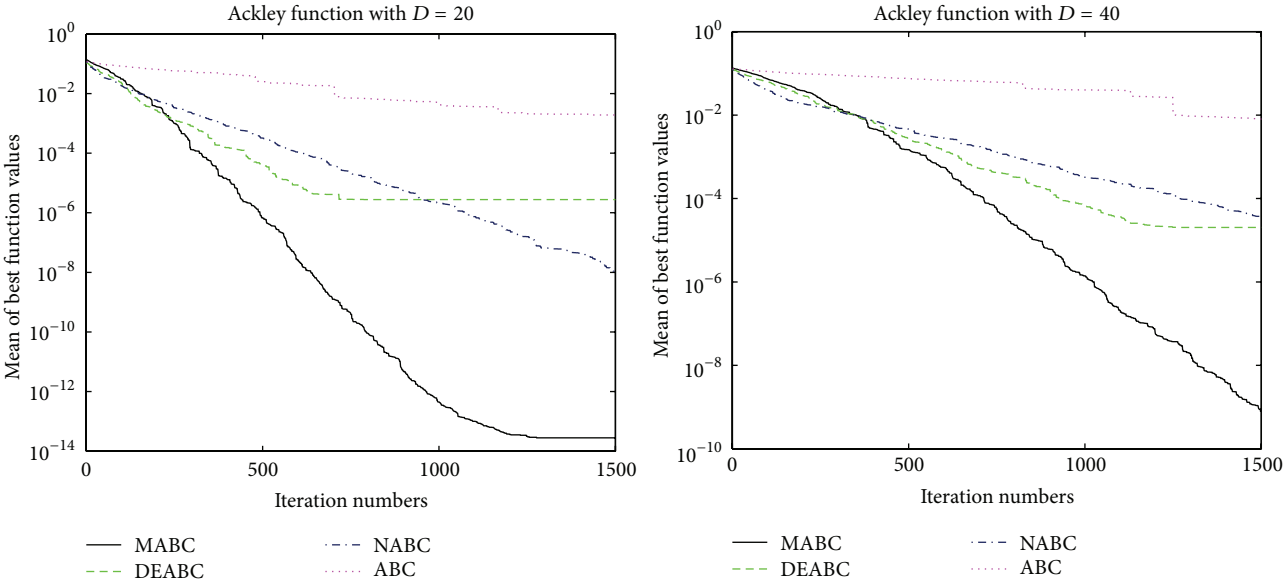

Figure 6: Convergence performance of ABC, DEABC, NABC, and MABC.

spends time the least, and the time of search solution of MABC is independent of dimension sizes. And the Sphere function costs least computational time, while the Rastrigin function takes the maximum one, so it shows that the finding solution time is related to the function types. Further, the time of search solution of MABC is less than that of any one of ABC, NABC, and DEABC.

Moreover, based on the six benchmark functions with the dimension $D = 20$ or 40, Table 5 displays all of the reduction rates, and the reduction rate of total iteration time is at least 73.52% and up to 85.88% for MABC to DEABC, MABC to NABC, and MABC to ABC.

5. Concluding Remarks

In this paper, to improve classical artificial bee colony algorithms, a new modified artificial bee colony (MABC) algorithm was developed, which is poor at exploitation and has some defects such as slow search speed, poor population diversity, the stagnation in the working process, and being trapped into the local optimal solution.

In order to make the initial population diversity distribute evenly and to take full use of the search space information, the initial population was constructed by using the opposition-based learning theory. Next, a new S-type method of grouping

Table 5: Reduction rate of total iteration time.

Function	D	MABC to DEABC	MABC to NABC	MABC to ABC
Sphere	20	85.38%	85.88%	85.43%
	40	85.11%	85.83%	85.19%
SumSquares	20	81.08%	81.41%	81.17%
	40	80.59%	81.38%	80.77%
Schwefel 2.22	20	84.26%	84.64%	84.42%
	40	83.53%	84.50%	83.74%
Griewank	20	77.81%	78.22%	77.83%
	40	75.39%	76.19%	73.52%
Rastrigin	20	83.80%	83.96%	83.84%
	40	82.23%	83.28%	82.35%
Ackley	20	82.26%	82.80%	82.48%
	40	81.20%	81.93%	81.07%

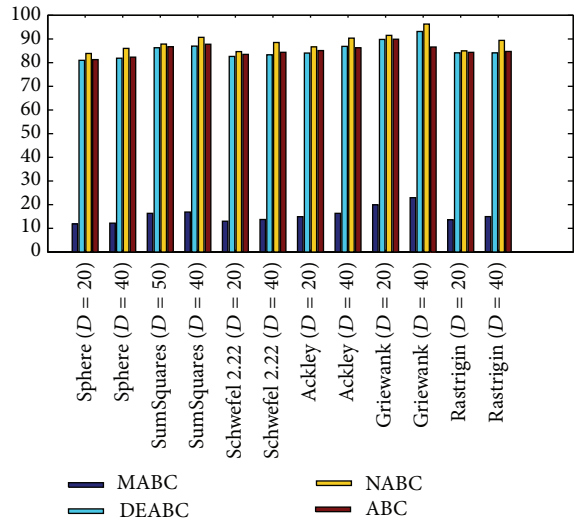

Figure 7: Total run time comparison of ABC, NABC, DEABC, and MABC.

was applied to reduce the difference between each group and replaced a greedy method (i.e., the roulette wheel way) by free search algorithm-sensitivity with pheromone. Then, a suitable exponential function updating length step was designed for replacing the original random step and carrying out faster and more stable searching of the global optimal solution.

Finally, from the new test function versions CEC13, six benchmark functions with the dimensions $D = 20$ and $D = 40$ were chosen and applied in the experiments for analyzing and comparing the iteration speed and accuracy of MABC to differential evolution artificial bee colony algorithm (DEABC), novel artificial bee colony algorithm (NABC), and basic artificial bee colony algorithm (ABC). With respect to the six benchmark functions with the dimensions $D = 20$ and $D = 40$, respectively, one can see from the experimental results in Tables 4 and 5 that the smallest standard deviation of the new modified algorithm is $8.89001e-18$, and the reduction rate of total iteration time for the new modified algorithm relative to the other three proposed artificial bee colony algorithms is at least 73.52% and up to 85.88%. Moreover, the experimental results presented in Figures 1–7 show that MABC has faster and more stable searching and can increase poor population diversity and avoid falling into the local optimal solutions.

MABC is superior to ABC, NABC, and DEABC, and the feasibility and effectiveness of MABC are very easy to be implemented. However, some other state-of-the-art algorithms may be selected as comparable algorithms, and more complex test functions and more diverse set of benchmark functions may be implemented to validate the benefits of MABC. Thus, the following works as *open questions* will be worth further studying:

(i) Some applications of MABC in the areas of pattern classification, fuzzy control, nonlinear system modeling, parameter tuning of proportional-integral-derivative controller, and so on.

(ii) Selecting more state-of-the-art algorithms as comparable algorithms.

(iii) Simulating more diverse set of benchmark functions for corresponding swarm intelligence algorithms.

(iv) Implementing more complex functions for validating the benefits of MABC and other corresponding improving swarm intelligence algorithms.

Conflict of Interests

The authors declare that they have no competing interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by Sichuan Province Cultivation Fund Project of Academic and Technical Leaders and the Open Research Fund of Key Laboratory of Higher

Education of Sichuan Province for Enterprise Informationalization and Internet of Things (2013WZJ01) and the Scientific Research Project of Sichuan University of Science & Engineering (2015RC07) and cofinanced by National Natural Science Foundation of China (61573010 and 11501391).

References

[1] W. L. Xiang, S. F. Ma, and M. Q. An, "Habcde: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution," *Applied Mathematics and Computation*, vol. 238, pp. 370–386, 2014.

[2] H.-C. Tsai, "Integrating the artificial bee colony and bees algorithm to face constrained optimization problems," *Information Sciences*, vol. 258, pp. 80–93, 2014.

[3] M. F. Tasgetiren, Q.-K. Pan, P. N. Suganthan, and A. Oner, "A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion," *Applied Mathematical Modelling*, vol. 37, no. 10-11, pp. 6758–6779, 2013.

[4] K. S. Tang, K. F. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 22–37, 1996.

[5] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., pp. 760–766, Springer US, 2010.

[6] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[7] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[8] S. Zhang and S. Liu, "A novel artificial bee colony algorithm for function optimization," *Mathematical Problems in Engineering*, vol. 2015, Article ID 129271, 10 pages, 2015.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report tr06, Computer Engineering Department, Engineering Faculty, Erciyes University, Kayseri, Turkey, 2005.

[10] M. El-Abd, "Performance assessment of foraging algorithms vs. evolutionary algorithms," *Information Sciences*, vol. 182, no. 1, pp. 243–263, 2012.

[11] T. Y. Lim, M. A. Al-Betar, and A. T. Khader, "Adaptive pair bonds in genetic algorithm: an application to real-parameter optimization," *Applied Mathematics and Computation*, vol. 252, pp. 503–519, 2015.

[12] Y. Xiang, Y. Peng, Y. Zhong, Z. Chen, X. Lu, and X. Zhong, "A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization," *Computational Optimization and Applications*, vol. 57, no. 2, pp. 493–516, 2014.

[13] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[14] J. C. Bansal, H. Sharma, and S. S. Jadon, "Artificial bee colony algorithm: a survey," *International Journal of Advanced Intelligence Paradigms*, vol. 5, no. 1-2, pp. 123–159, 2013.

[15] A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Artificial bee colony algorithm, its variants and applications: a survey," *Journal of Theoretical and Applied Information Technology*, vol. 47, no. 2, pp. 434–459, 2013.

[16] H. Habbi, Y. Boudouaoui, D. Karaboga, and C. Ozturk, "Self-generated fuzzy systems design using artificial bee colony optimization," *Information Sciences*, vol. 295, pp. 145–159, 2015.

[17] L. N. Vitorino, S. F. Ribeiro, and C. J. A. Bastos-Filho, "A mechanism based on artificial bee colony to generate diversity in particle swarm optimization," *Neurocomputing*, vol. 148, pp. 39–45, 2015.

[18] M. S. Kiran and O. Findik, "A directed artificial bee colony algorithm," *Applied Soft Computing Journal*, vol. 26, pp. 454–462, 2015.

[19] D. Karaboga and B. Gorkemli, "A quick artificial bee colony (qABC) algorithm and its performance on optimization problems," *Applied Soft Computing Journal*, vol. 23, pp. 227–238, 2014.

[20] N. Imanian, M. E. Shiri, and P. Moradi, "Velocity based artificial bee colony algorithm for high dimensional continuous optimization problems," *Engineering Applications of Artificial Intelligence*, vol. 36, pp. 148–163, 2014.

[21] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, and J.-s. Pan, "Multi-strategy ensemble artificial bee colony algorithm," *Information Sciences*, vol. 279, pp. 587–603, 2014.

[22] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations," *Information Sciences*, vol. 270, pp. 112–133, 2014.

[23] S. Das, S. Biswas, and S. Kundu, "Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization," *Applied Soft Computing*, vol. 13, no. 12, pp. 4676–4694, 2013.

[24] M. X. Zang, X. Ma, and Y. M. Duan, "Improved artificial bee colony algorithm," *Journal of Xidian University*, vol. 42, no. 2, pp. 65–70, 2015.

[25] C. Zhang, J. Zheng, and Y. Zhou, "Two modified artificial bee colony algorithms inspired by grenade explosion method," *Neurocomputing*, vol. 151, no. 3, pp. 1198–1207, 2015.

[26] M. Mernik, S.-H. Liu, D. Karaboga, and M. Črepinšek, "On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation," *Information Sciences*, vol. 291, pp. 115–127, 2015.

[27] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Information Sciences*, vol. 300, pp. 140–157, 2015.

[28] T. G. Chen and R. B. Xiao, "Modeling design iteration in product design and development and its solution by a novel artificial bee colony algorithm," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 240828, 13 pages, 2014.

[29] B. Li, "Research on WNN modeling for gold price forecasting based on improved artificial bee colony algorithm," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 270658, 10 pages, 2014.

[30] S. Wei and N. Z. Lu, "An improvement for artificial bee algorithm," *Modern Computer*, vol. 6, no. 17, pp. 25–29, 2014.

[31] L. Li, F. M. Yao, L. J. Tan et al., "A novel DE-ABC-based hybrid algorithm for global optimization," in *Proceedings of the 7th International Conference on Intelligent Computing (ICIC '11), Zhengzhou, China, August 2011*, D. S. Huang, Y. Gan, P. Premaratne, and K. Han, Eds., Lecture Notes in Computer Science, pp. 558–565, Springer, Berlin, Germany, 2012.

[32] S.-M. Chen, A. Sarosh, and Y.-F. Dong, "Simulated annealing based artificial bee colony algorithm for global numerical

optimization," *Applied Mathematics and Computation*, vol. 219, no. 8, pp. 3575–3589, 2012.

[33] S. C. Satapathy and A. Naik, "Modified teaching-learning-based optimization algorithm for global numerical optimization—a comparative study," *Swarm and Evolutionary Computation*, vol. 16, pp. 28–37, 2014.

[34] Q. Xu, L. Wang, N. Wang, X. Hei, and L. Zhao, "A review of opposition-based learning from 2005 to 2012," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 1–12, 2014.

[35] K. Penev, "Free search—comparative analysis 100," *International Journal of Metaheuristics*, vol. 3, no. 2, pp. 118–132, 2014.