Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3419410, 11 pages
https://doi.org/10.1155/2019/3419410

Research Article

Improving Artificial Bee Colony Algorithm Using a Dynamic Reduction Strategy for Dimension Perturbation

Gan Yu[1], **Hongzhi Zhou**[1] and **Hui Wang**[2,3]

[1]*College of Information Engineering, Fuyang Normal University, Fuyang 236041, China*
[2]*School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China*
[3]*Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang 330099, China*

Correspondence should be addressed to Hongzhi Zhou; honeyzhz@126.com

Received 15 April 2019; Accepted 27 June 2019; Published 14 July 2019

Academic Editor: Erik Cuevas

To accelerate the convergence speed of Artificial Bee Colony (ABC) algorithm, this paper proposes a Dynamic Reduction (DR) strategy for dimension perturbation. In the standard ABC, a new solution (food source) is obtained by modifying one dimension of its parent solution. Based on one-dimensional perturbation, both new solutions and their parent solutions have high similarities. This will easily cause slow convergence speed. In our DR strategy, the number of dimension perturbations is assigned a large value at the initial search stage. More dimension perturbations can result in larger differences between offspring and their parent solutions. With the growth of iterations, the number of dimension perturbations dynamically decreases. Less dimension perturbations can reduce the dissimilarities between offspring and their parent solutions. Based on the DR, it can achieve a balance between exploration and exploitation by dynamically changing the number of dimension perturbations. To validate the proposed DR strategy, we embed it into the standard ABC and three well-known ABC variants. Experimental study shows that the proposed DR strategy can efficiently accelerate the convergence and improve the accuracy of solutions.

1. Introduction

Artificial Bee Colony (ABC) is an efficient optimization tool, which mimics the foraging behavior of honey bees [1]. It has some superior characteristics, such as simple concept, few control parameters, and strong search ability. In the last decade, ABC has been widely applied to many optimization problems [2].

However, some studies pointed out that ABC has snow convergence and weak exploitation ability in solving complex problems [3]. In ABC, there are many bees and food sources (called solutions). The process of finding food sources is abstracted to search potential solutions. For bees, Karaboga defined a simple model to search new solutions (food sources) by changing one dimension of the current solutions (parent solutions) [4]. This may result in very small differences between new solutions and their corresponding solutions. Thus, the convergence speed becomes very slow.

To overcome the above issue, a Dynamic Reduction (DR) strategy for dimension perturbation in ABC is proposed. In DR, the number of dimension perturbations is initialized to a predefined value. As the iteration increases, the number of dimension perturbations dynamically decreases. By changing the number of dimension perturbations, the convergence speed can be improved. To validate the proposed DR strategy, we embed it into ABC and other three improved versions. Simulation results show that the DR strategy can efficiently accelerate the convergence and improve the accuracy of solutions.

The rest of the paper is organized as follows. Section 2 introduces the standard ABC. A short literature review of ABC is presented in Section 3. Our strategy is described in Section 4. Experimental results and analysis are given in Section 5. Finally, this work is concluded in Section 6.

2. Standard ABC

Intelligent optimization algorithms (IOAs) have attracted much attention in the past several years. Recently, different

IOAs were proposed to solve various optimization problems [5–13]. ABC is a popular optimization algorithm based on swarm intelligence [1]. It is motivated by the foraging behavior of bees, labor division, and information sharing. It is usually used to solve continuous or discrete optimization problems. According to the labor division mode of bees, ABC employs different search strategies to complete the optimization task together with few control parameters, strong stability, and simple implementation.

In ABC, there are three types of bees including employed bees, onlooker bees, and scouts. These bees are related to three search processes: employed bee phase, onlooker bee phase, and scout phase. The quantity of employed bees is equal to the onlooker bees.

Let us consider an initial population with SN solutions $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$, where i=1,2,...,SN, SN is the population size, and D is the dimension size. In the employed bee phase, each employed bee is responsible for searching the neighborhood of a solution. For the i-th solution X_i, the corresponding employed bee finds a new solution V_i according to the search strategy [1]:

$$v_{i,j} = x_{i,j} + \varphi_{i,j}\left(x_{i,j} - x_{k,j}\right), \tag{1}$$

where φ is a random value within [-1, 1], X_k is a different solution randomly chosen from the swarm and $k \neq I$, and j is an integer randomly chosen from the set $\{1, 2, \ldots, D\}$. The standard ABC employs an elite selection method to determine which solution is chosen from V_i and X_i. If V_i is better than X_i, V_i will be selected into the next iteration and the current X_i is replaced.

From (1), the differences between V_i and X_i are the j-th dimension. For the rest of D-1 dimensions, both V_i and X_i are the same. Thus, V_i and X_i are very similar. Even if X_i is replaced by V_i, the new solution V_i is also near X_i. It means that the step size for the current search is very small, because it only jumps in one dimension. As a result, the search will be slow.

In the onlooker bee phase, the onlooker bees focus on deep search. Unlike the employed phase, the onlooker bees do not search the neighborhoods of all solutions in the swarm. ABC firstly calculates the selection probability of each solution. Based on the probability, a solution is chosen and a new solution is generated in its neighborhood. The probability ($prob_i$) for the i-th solution is defined by [1]

$$prob_i = \frac{fitness_i}{\sum_{j=1}^{SN} fitness_j}, \tag{2}$$

where $fitness_i$ is the fitness of X_i and it is computed by [1]

$$fitness_i = \begin{cases} \dfrac{1}{1 + fval_i}, & \text{if } fval_i \geq 0 \\ 1 + \left|fval_i\right|, & \text{otherwise,} \end{cases} \tag{3}$$

where $fval_i$ is the function value of X_i.

Similar to the employed bees, the onlooker bees also use (1) to attain a new solution. Then, the function value of V_i is compared with X_i. If V_i is better than X_i, V_i will be selected into the next iteration and the current X_i is replaced.

For the elite selection method, when V_i is worse than X_i, the improvement of X_i is failure; otherwise the improvement is successful. For each solution X_i in the population, a counter $trail_i$ is used to calculate the number of failures. If $trail_i$ is very large, it means that X_i may fall into local minima and cannot jump out. Under such circumstance, X_i is initialized as follows [1]:

$$x_j = Low_j + r_j \cdot \left(Up_j - Low_j\right), \tag{4}$$

where r_j is a random number within [0, 1] and [Low, Up] is the definition domain.

3. A Brief Review of Recent Work on ABC

In the past decade, research on ABC has attracted wide attention. Different ABCs and their applications were proposed. In this section, a brief review of this work is presented.

The standard ABC obtained good performance on many optimization problems, but it showed some difficulties in complex problems [3]. To enhance the performance of ABC, various ABC algorithms were proposed. In [3], the global best solution ($Gbest$) was utilized to modify the search equation and experiments proved its effectiveness. Alatas [14] used different chaotic maps to adjust the parameters of ABC. In [15], a Rosenbrock ABC was proposed, in which the rotational direction of Rosenbrock was employed to enhance the exploitation capacity. Experiments showed that Rosenbrock ABC could improve the convergence and accuracy. Motivated by the mutation equation of Differential Evolution (DE), a new search equation was designed. As mentioned before, only one dimension is perturbed in the standard ABC [16]. In [17], a parameter MR was defined to determine the probability of dimension perturbations. A larger MR means more dimension perturbations. Experimental results show a suitable MR can accelerate the convergence of ABC. In [18], an external archive was used in ABC to guide the search. Li et al. [19] employed multiple strategies including the best solution, inertia weight, and Lévy mutation to enhance the search.

In [20], multiple strategies ensemble of ABC was proposed, in which multiple search strategies were employed instead of a single search strategy. Similar to [20], Kiran et al. [21] used five search strategies in ABC and obtained promising performance. To determine which search strategies are chosen, a roulette wheel selection mechanism was used. In [22], a bare bones ABC based on Gaussian sampling was proposed. Moreover, a new method was used to calculate the selection probability in the onlooker bee phase. In [23], an adaptive method was used to change the swarm size of ABC. The search equation was also modified based on DE/rand/1 mutation. In [24], depth-first search and elite guided search were used in ABC. In [25], recombination operation was introduced into ABC to enhance exploitation. Xiang et al. [26] introduced decomposition technique into ABC and extended ABC to solve many-objective optimization problems. Experiments on 13 problems with up to 50 objectives showed that the decomposition technique can effectively help ABC achieve good results.

Dokeroglu et al. [27] proposed an island parallel ABC to solve the Quadratic Assignment Problem (QAP), in which Tabu search was used to balance exploitation and exploration. Ni et al. [28] used an improved version of ABC for optimizing cumulative oil steam ratio. The recombination and random perturbation were employed to maintain diversity and improve the search. In [19, 29], ABC was applied to image contrast enhancement, where the objective function is designed based on a new image contrast measure. For a generalized covering traveling salesman problem, Pandiri and Singh [30] proposed a new ABC with variable degree of perturbation. Pavement resurfacing aims to extend the service life of pavement. Panda and Swamy [31] used a new ABC to find the optimal scenarios in pavement resurfacing optimization problem. Sharma et al. [32] designed a new ABC based on beer froth decay phenomenon to find the optimal job sequence in job shop scheduling problem.

4. Dynamic Reduction of Dimension Perturbation in ABC

The standard ABC uses (1) as the search strategy for both employed and onlooker bees. Based on (1), only one dimension is perturbed between the parent solution X and offspring V. Such one-dimensional perturbation will result in large similarities between X and V. Consequently, the convergence speed may easily be slowed down during the search. This is the possible reason that many studies reported that ABC was not good at exploitation.

To tackle the above issue, a Dynamic Reduction (DR) strategy for dimension perturbation is proposed in ABC. Initially, the number of dimension perturbations is fixed to a large value (less than the dimensional size D). More dimension perturbations can result in larger differences between offspring and their parent solutions. This will be helpful to accelerate the search and find better solutions quickly. As the growth of iterations, the number of dimension perturbations dynamic decreases. Less dimension perturbations can reduce the dissimilarities between offspring and their parent solutions. This will be beneficial for finding more accurate solutions.

Assume that $DP(t)$ is the number of dimension perturbations at iteration t. Based on the DR strategy, $DP(t)$ is dynamically updated by

$$DP(t) = \left(1 - \frac{t}{T_{\max}}\right) \cdot D_0, \tag{5}$$

where T_{max} is the maximum number of iterations and D_0 is an initial value for dimension perturbation. In this paper, D_0 is set to $\lambda \cdot D$, and $\lambda \in (0, 1]$. At the beginning, $DP(0)$ is equal to D_0. As the iteration increases, $DP(t)$ gradually decreases from D_0 to zero. When $DP(t)<1$, the number of dimension perturbations is less than one. It is obvious that this case is not accepted. To avoid this case, a simple method is employed as follows:

$$DP(t) = \begin{cases} DP(t), & \text{if } DP(t) \geq 1 \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

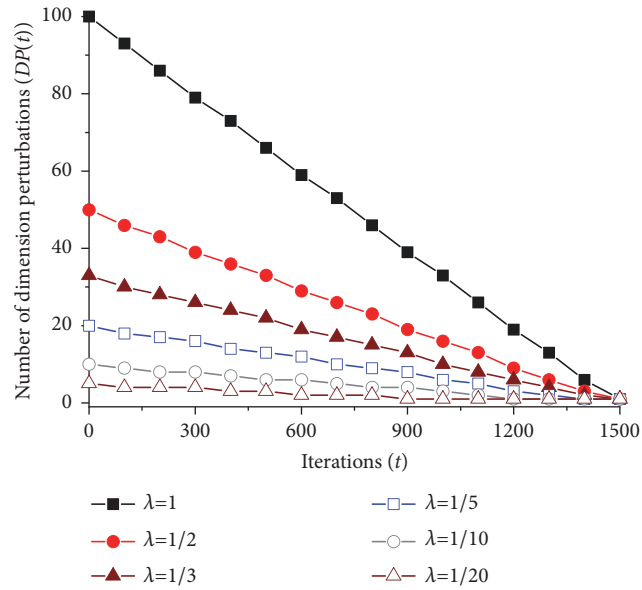

Figure 1: The dynamic changes of $DP(t)$ under different λ.

To clearly illustrate the DR strategy, Figure 1 shows the dynamic changes of $DP(t)$ during the iteration. In this case, D and T_{max} are set to 100 and 1500, respectively. As seen, the number of dimension perturbations gradually decreases with growth of iterations. The initial value of $DP(t)$ is related to D_0. By setting different values of λ, we can clearly observe the characteristics of our DP strategy. A larger λ means more dimension perturbations; a smaller λ means less dimension perturbations. So, it is not an easy task to choose the best parameter λ. In Section 5.2, the effects of the parameter λ are investigated.

From (6), the number of dimension perturbations $DP(t)$ varies from D_0 to 1. However, it is not convenient to implement (6) in ABC. To overcome this problem, a probability (ρ) is used to replace the number of dimension perturbations ($DP(t)$). The probability $\rho(t)$ for dimension perturbation at the t-th iteration is defined by

$$\rho(t) = \frac{DP(t)}{D}. \tag{7}$$

Let us consider an extreme case for (7). When $DP(t)$=1 and D=1000, $\rho(t)$ is equal to 0.001. It is possible that no dimension is perturbed for a small $\rho(t)$. To prevent this case, a simple method is used to ensure that the number of dimension perturbations is not less than 1. For the i-th solution X_i, a random value is generated for each dimension of X_i. If the random value satisfies the probability $\rho(t)$, the corresponding dimension of X_i is chosen for dimension perturbation according to the search equation (i.e., (1) for the standard ABC). If no dimension perturbation occurs, a dimension index is randomly chosen and executes the dimension perturbation. The main procedure of the dimension perturbation is described in Table 1, where *rand*(0,1) is a random value between 0 and 1.0.

Table 1: Procedure of the dimension perturbation.

Line	Dimension Perturbation (i)
1	**For** j=1 to D
2	**If** $rand(0,1)<\rho(t)$
3	Execute the perturbation for the j-th dimension according to the search equation;
4	**End If**
5	**End For**
6	**If** *no dimension is perturbed*
7	Randomly select a dimension j;
8	Execute the perturbation for the j-th dimension according to the search equation;
9	**End If**

Table 2: Benchmark functions.

Name	Dimension Size	Search Range	Global Optimum
Sphere (f_1)	30	[-100,100]	0
Schwefel 2.22 (f_2)	30	[-10,10]	0
Schwefel 1.2 (f_3)	30	[-100,100]	0
Schwefel 2.21 (f_4)	30	[-100,100]	0
Rosenbrock (f_5)	30	[-30,30]	0
Step (f_6)	30	[-100,100]	0
Quartic (f_7)	30	[-1.28,1.28]	0
Schwefel 2.26 (f_8)	30	[-500,500]	-12569.5
Rastrigin (f_9)	30	[-5.12,5.12]	0
Ackley (f_{10})	30	[-32,32]	0
Griewank (f_{11})	30	[-600,600]	0
Penalized (f_{12})	30	[-50,50]	0

5. Experimental Results

5.1. Benchmark Functions. To validate the performance of the proposed Dynamic Reduction strategy for dimension perturbation, twelve classical benchmark functions are employed. These functions were used in many optimization papers [21, 33–35]. Table 2 lists the function names, search range, dimension size, and global optimum. For detailed definitions of these functions, please refer to [35].

5.2. Investigations of the Parameter λ. In the proposed DR strategy, the number of dimension perturbations $DP(t)$ is proportional to D_0, and $D_0=\lambda\cdot D$ is used in this paper. The parameter λ plays a significant role in controlling the number of dimension perturbations. A larger λ means more dimension perturbations and a smaller λ represents less dimension perturbations. In this section, the standard ABC is used as an example. The DR strategy is embedded into ABC and a new ABC variant, namely, DR-ABC, is constructed. We focus on investigating the effects of different λ on the performance of ABC. This is helpful to choose a reasonable parameter λ.

In the experiments, the parameter λ is set to 1.0, 1/2, 1/3, 1/5, 1/10, and 1/20, respectively. SN=50 and $limit$=100 are used. When the number of function evaluations (FEs) reaches $MAXFES$, DR-ABC is terminated. For D=30, $MAXFES$ is equal to 5000·D=1.5E+05. Then, T_{max} is equal to 1500. For each function, DR-ABC is run 50 times.

Table 3 shows the results of DR-ABC under different λ, where "Mean" represents the mean best function value. From the results, λ=1/2 achieves the best performance on f_1 and f_2. For function f_3, λ=1/5 and λ=1/10 outperform other λ values and ABC. A large λ is better for function f_4, while a small λ is better for functions f_5 and f_8. For f_5, ABC is better than λ=1.0, 1/2, 1/3. and 1/5. All λ values can help ABC find better solutions on f_7, f_{10}, and f_{11}. For the rest of function f_{12}, λ=1.0 and 1/2 obtains worse results than ABC. With the increase of λ, the performance of DR-ABC is approaching the standard ABC.

Figure 2 presents some convergence graphs of ABC and DR-ABC under different λ. For f_1, the standard ABC is faster than DR-ABC at the beginning search stage. As the iteration increases, the Dynamic Reduction strategy can accelerate the convergence and λ=1/3 obtains the fastest convergence speed. For f_3, λ=1.0 and 1/2 do not improve the convergence speed at the middle and last search stages. It demonstrates that small λ values are helpful to improve the convergence. For f_4, λ=1/20 shows the worst convergence among all λ values. Similar to f_3, large λ values do not show advantages at the beginning and middle search stages.

Based on above results, it is not an easy task to choose the best λ value. Too large or too small λ values may slow down the convergence at the beginning search stage. To choose a reasonable λ value, the mean rank values of ABC and

Table 3: Results of DR-ABC under different λ.

Functions	DR-ABC (λ=1.0)	DR-ABC (λ=1/2)	DR-ABC (λ=1/3)	DR-ABC (λ=1/5)	DR-ABC (λ=1/10)	DR-ABC (λ=1/20)	ABC
	Mean	Mean	Mean	Mean	Mean	Mean	Mean
f_1	3.42E-17	3.84E-21	4.06E-21	3.42E-20	2.34E-18	4.25E-17	9.48E-16
f_2	1.59E-12	7.09E-14	1.44E-13	6.65E-13	1.76E-11	6.79E-11	1.89E-10
f_3	9.51E+03	9.24E+03	8.44E+03	4.75E+03	4.80E+03	5.90E+03	1.01E+04
f_4	1.53E+01	1.68E+01	1.74E+01	1.66E+01	2.49E+01	2.98E+01	3.13E+01
f_5	2.65E+01	2.28E+01	1.87E+01	5.68E+00	8.61E-02	3.98E-01	1.47E+00
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	8.65E-02	8.75E-02	9.08E-02	7.45E-02	1.48E-01	1.68E-01	2.05E-01
f_8	-11347.8	-12184.8	-12503.9	-12569.5	-12569.5	-12569.5	-12450.7
f_9	1.21E+01	4.07E+00	2.06E-05	3.44E-08	5.01E-12	5.77E-13	3.73E-14
f_{10}	8.43E-10	5.94E-11	6.39E-11	5.82E-11	3.81E-10	6.29E-10	1.26E-09
f_{11}	4.37E-14	2.23E-15	0.00E+00	0.00E+00	0.00E+00	1.67E-15	4.78E-13
f_{12}	7.73E-06	1.47E-15	3.25E-17	3.02E-17	3.02E-17	3.06E-17	8.79E-16

Table 4: Mean rank of ABC and DR-ABC under different λ.

Algorithms	Mean Rank
ABC	5.58
DR-ABC (λ=1.0)	5.17
DR-ABC (λ=1/2)	4.08
DR-ABC (λ=1/3)	3.50
DR-ABC (λ=1/5)	2.38
DR-ABC (λ=1/10)	3.21
DR-ABC (λ=1/20)	4.08

DR-ABC with each λ are calculated according to Friedman test [35]. Table 4 presents the mean rank values of the seven ABC algorithms. From the results, DR-ABC (λ=1/5) obtains the best rank value 2.38. It means that λ=1/5 is the relatively best parameter setting. The standard ABC has the worst rank 5.58. It demonstrates that the proposed Dynamic Reduction with any λ values can help ABC find more accurate solutions.

5.3. Different Methods for Dimension Perturbation. In this paper, we propose a Dynamic Reduction method for dimension perturbations. When generating offspring, the number of dimension perturbations is dynamically decreased as the iteration increases. In [17], Akay and Karaboga designed a parameter *MR* to control the probability of dimension perturbations. In the search process, the parameter *MR* is fixed. Then, the number of dimension perturbations is fixed to $MR \cdot D$. This method is called fixed number of dimension perturbations ABC (FNDP-ABC). Moreover, the number of dimension perturbations may be randomly determined. In this section, different methods for dimension perturbation are investigated and the involved methods are listed as follows:

(i) ABC (with only one-dimensional perturbation);

(ii) fixed number of dimension perturbations in ABC (FNDP-ABC);

(iii) random number of dimension perturbations in ABC (RNDP-ABC);

(iv) the proposed Dynamic Reduction of dimension perturbation in ABC (DR-ABC).

The parameter settings of DR-ABC, FNDP-ABC, RNDP-ABC, and ABC are the same with Section 5.2. For DR-ABC, the parameter λ is equal to 1/5. Then, the number of dimension perturbations decreases from 6 to 1, because of $D_0=\lambda \cdot D=6$. For FNDP-ABC, we use an average value 3 as the fixed number of dimension perturbations (the probability *MR* is equal to 3/30=0.1). For RNDP-ABC, the number of dimension perturbations is randomly chosen between 1 and 6.

Table 5 lists the comparison results of different methods for dimension perturbation. From the results, DR-ABC, FNDP-ABC, and RNDP-ABC outperform ABC on all problems except for f_5, f_6, and f_9. On f_5 and f_9, all dimension perturbation strategies are not effective. All algorithms can find the global minima on f_6. For other nine functions, three strategies can improve the quality of solutions. Among FNDP-ABC, RNDP-ABC, and DR-ABC, DR-ABC performs better than RNDP-ABC and FNDP-ABC. RNDP-ABC is better than FNDP-ABC. Results demonstrate that the Dynamic Reduction is better than fixed and random strategies.

Figure 3 gives the convergence graphs of ABC with different dimension perturbation strategies. For functions f_1, f_2, and f_4, DR-ABC is the fastest algorithm. For the first two functions, the convergence characteristics of FNDP-ABC and RNDP-ABC are similar and they are faster than ABC (with only one-dimensional perturbations). For f_3, RNDP-ABC is faster than FNDP-ABC, DR-ABC, and ABC at the middle search stage. FNDP-ABC and RNDP-ABC converge much faster than ABC on f_4. From these convergence figures, all dimension strategies can accelerate the search and our proposed dynamic reduction is better than the other two methods.

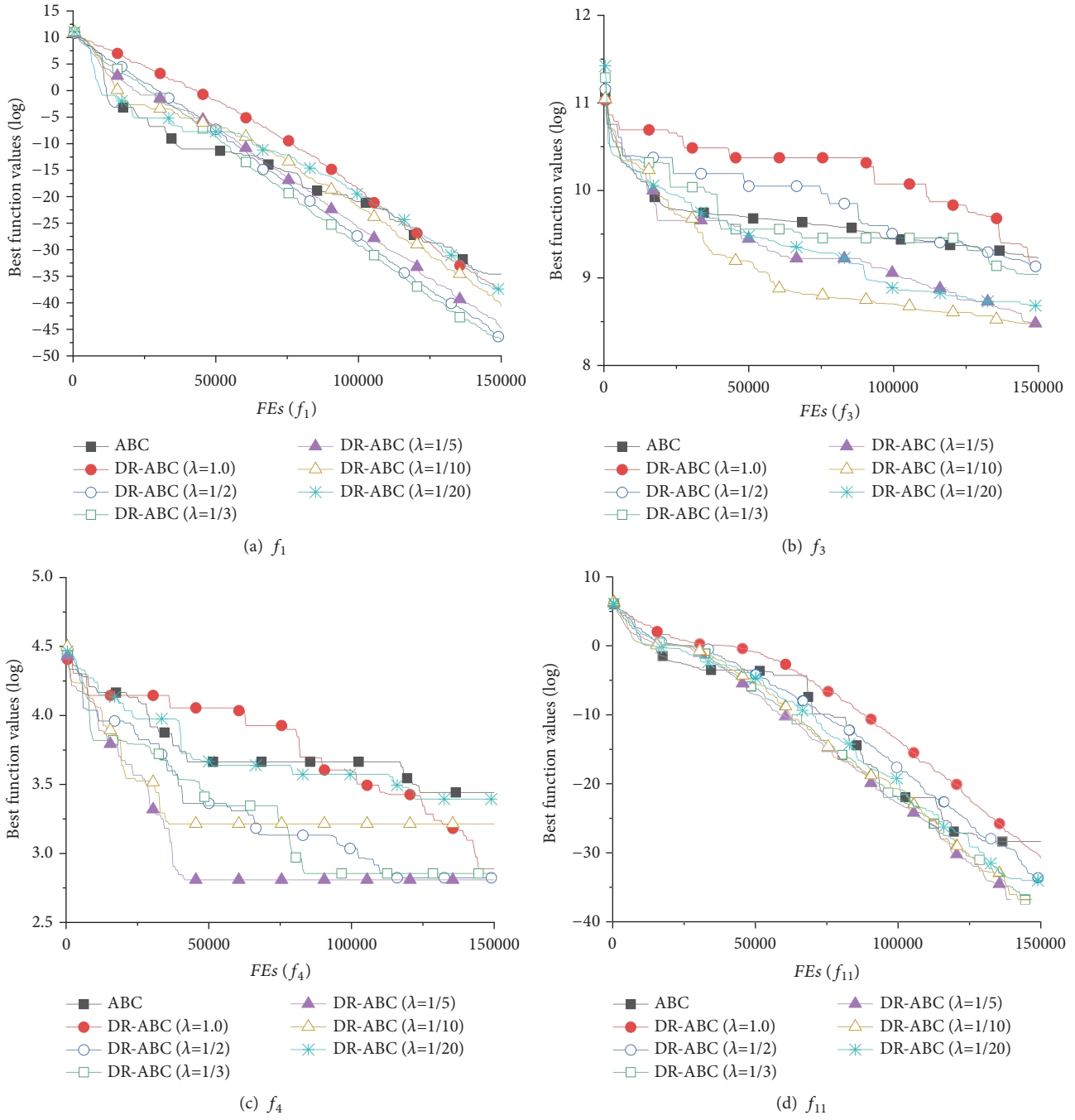

Figure 2: The convergence graphs of ABC and DR-ABC under different λ.

5.4. Extending the Dynamic Reduction Strategy into Other Well-Known ABC Algorithms. The above experiments proved that our proposed Dynamic Reduction strategy for dimension perturbation is effective to enhance the optimization performance of ABC. In this section, the Dynamic Reduction strategy is extended to three other well-known ABC algorithms including GABC [3], MABC [16], and ABCVSS [21]. The involved ABCs are listed as follows:

(i) *Gbest* guided ABC (GABC) [3];

(ii) GABC with Dynamic Reduction strategy for dimension perturbation (DR-GABC);

(iii) modified ABC (MABC) [16];

(iv) MABC with Dynamic Reduction strategy for dimension perturbation (DR-MABC);

(v) ABC with variable search strategy (ABCVSS) [21];

(vi) ABCVSS with Dynamic Reduction strategy for dimension perturbation (DR-ABCVSS).

In the experiments, we investigate whether the Dynamic Reduction strategy is still effective in other ABC algorithms. The population size *SN*, *limit*, λ, and *MAXFES* are set to 50, 100, 1/5, and 1.5E+05, respectively. We use the original settings

Table 5: Comparison results of different methods for dimension perturbation.

Functions	ABC	FNDP-ABC	RNDP-ABC	DR-ABC
	Mean	Mean	Mean	Mean
f_1	9.48E-16	1.12E-18	5.59E-19	3.42E-20
f_2	1.89E-10	1.35E-12	1.06E-12	6.65E-13
f_3	1.01E+04	6.08E+03	5.52E+03	4.75E+03
f_4	3.13E+01	2.24E+01	1.93E+01	1.66E+01
f_5	1.47E+00	8.97E+00	7.27E+00	5.68E+00
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	2.05E-01	1.21E-01	1.11E-01	7.45E-02
f_8	-12450.7	-12569.5	-12569.5	-12569.5
f_9	3.73E-14	2.43E-05	4.36E-05	3.44E-08
f_{10}	1.26E-09	1.02E-10	1.54E-10	5.82E-11
f_{11}	4.78E-13	0.00E+00	0.00E+00	0.00E+00
f_{12}	8.79E-16	3.03E-17	3.03E-17	3.02E-17
f_{12}	9.48E-16	1.12E-18	5.59E-19	3.42E-20

Table 6: Results of different ABC algorithms with the dynamic reduction strategy.

Functions	GABC	DR-GABC	MABC	DR-MABC	ABCVSS	DR-ABCVSS
	Mean	Mean	Mean	Mean	Mean	Mean
f_1	4.61E-16	6.04E-42	3.11E-40	6.52E-47	8.40E-32	2.54E-52
f_2	1.59E-15	2.45E-24	1.45E-21	4.96E-28	6.46E-18	5.82E-31
f_3	6.41E+03	1.85E+03	9.71E+03	3.02E+03	7.98E+03	9.21E+02
f_4	1.57E+01	2.48E-01	4.81E+00	7.34E-02	2.18E+01	9.92E-02
f_5	1.68E-01	2.57E+01	7.86E-01	2.44E+01	2.49E+01	2.38E+01
f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_7	7.61E-02	3.23E-02	3.87E-02	6.89E-03	7.48E-02	1.59E-02
f_8	-12569.5	-12569.5	-12569.5	-12569.5	-12569.5	-12569.5
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{10}	3.61E-14	1.84E-14	3.26E-14	1.48E-14	4.68E-14	1.48E-14
f_{11}	1.94E-13	0.00E+00	0.00E+00	0.00E+00	1.58E-11	0.00E+00
f_{12}	5.54E-16	3.02E-17	3.02E-17	3.02E-17	3.02E-17	3.02E-17

in their corresponding references for other parameters in MABC, GABC, and ABCVSS [3, 16, 21]. For each function, each algorithm is run 50 times.

Table 6 lists the results of different ABC algorithms with the Dynamic Reduction strategy. From the results of DR-GABC and GABC, DR-GABC is better than GABC on eight functions, while GABC performs better than DR-GABC on only one function (f_5). Especially for f_1, f_2, f_3, f_4, and f_{11}, the Dynamic Reduction strategy significantly improves the performance of GABC. Similar to GABC, DR-MABC is worse than MABC on f_5. The Dynamic Reduction strategy helps MABC find more accurate solutions on six functions. For the rest of the five functions, both of them have the same results. DR-ABCVSS is significantly better than ABCVSS on five functions f_1, f_2, f_3, f_4, and f_{11}. For f_5, f_7, and f_{10}, DR-ABCVSS also outperforms ABCVSS.

Figure 4 lists the convergence graphs of different ABC algorithms with the Dynamic Reduction strategy. For functions f_1 and f_2, DR-ABCVSS is the fastest algorithm. DR-MABC and DR-GANC obtain the second and third place, respectively. DR-GABC is faster than DR-MABC on f_3. For f_4, DR-MABC converges faster than DR-GABC at the last search stage and DR-GABC is faster at the middle search stage. For all functions, all dynamic reduction based ABCs are better than GABC, MABC, and ABCVSS. By comparing each pair of ABC and its dynamic reduction based version, we can find that the Dynamic Reduction strategy can effectively accelerate the convergence speed.

Table 7: Mean rank of different ABC algorithms with the dynamic reduction strategy.

Algorithms	Mean Rank
GABC	4.54
DR-GABC	3.25
MABC	3.67
DR-MABC	2.54
ABCVSS	4.71
DR-ABCVSS	2.29

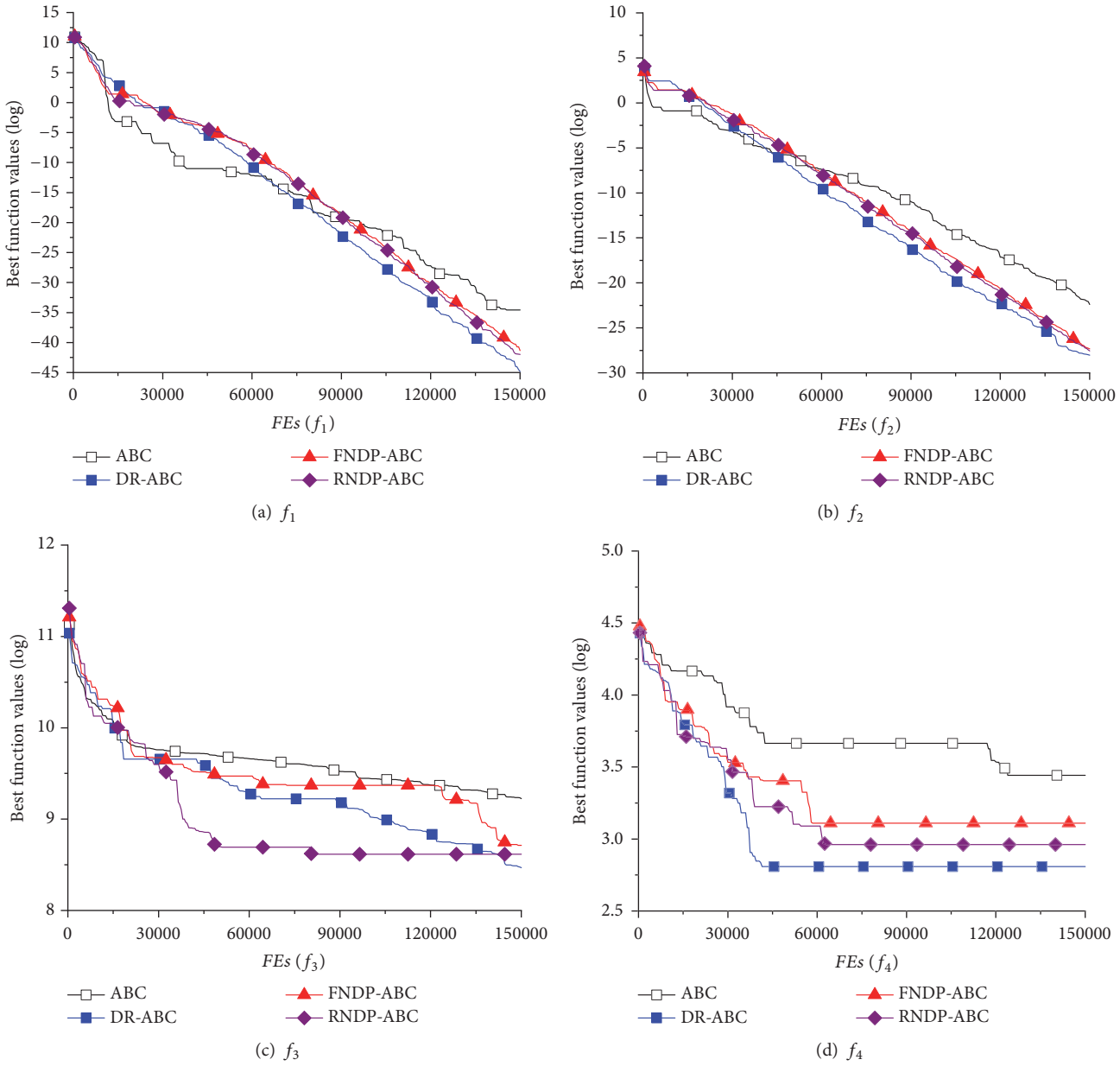

Figure 3: The convergence graphs of ABC with different dimension perturbation strategies.

Table 7 presents the mean rank of DR-GABC, GABC, DR-MABC, MABC, DR-ABCVSS, and ABCVSS. From the rank values of each pair of ABC and its Dynamic Reduction based version, the Dynamic Reduction strategy can help its parent ABC algorithm obtain better rank. For example, GABC achieves a rank value 4.54 and DR-GABC obtains a better rank 3.25. Especially for ABCVSS, it seems that this algorithm is the worst one on the benchmark set. By embedding the Dynamic Reduction strategy into ABCVSS, DR-ABCVSS achieves the best rank and becomes the best algorithm among six ABCs. The results confirm the effectiveness of our Dynamic Reduction strategy.

6. Conclusions

In this paper, a Dynamic Reduction (DR) strategy for dimension perturbation is proposed to accelerate the search of ABC. In the standard ABC, a new solution (offspring) is generated by perturbing one dimension of its parent solution. Based on one-dimensional perturbation, both new solutions and their parent solutions have high similarities. This will easily cause slow convergence speed. In our DR strategy, the number of dimension perturbations is assigned a large value at the initial search stage. More dimension perturbations can result in larger differences between offspring and their parent solutions. This will be helpful to accelerate the search. With

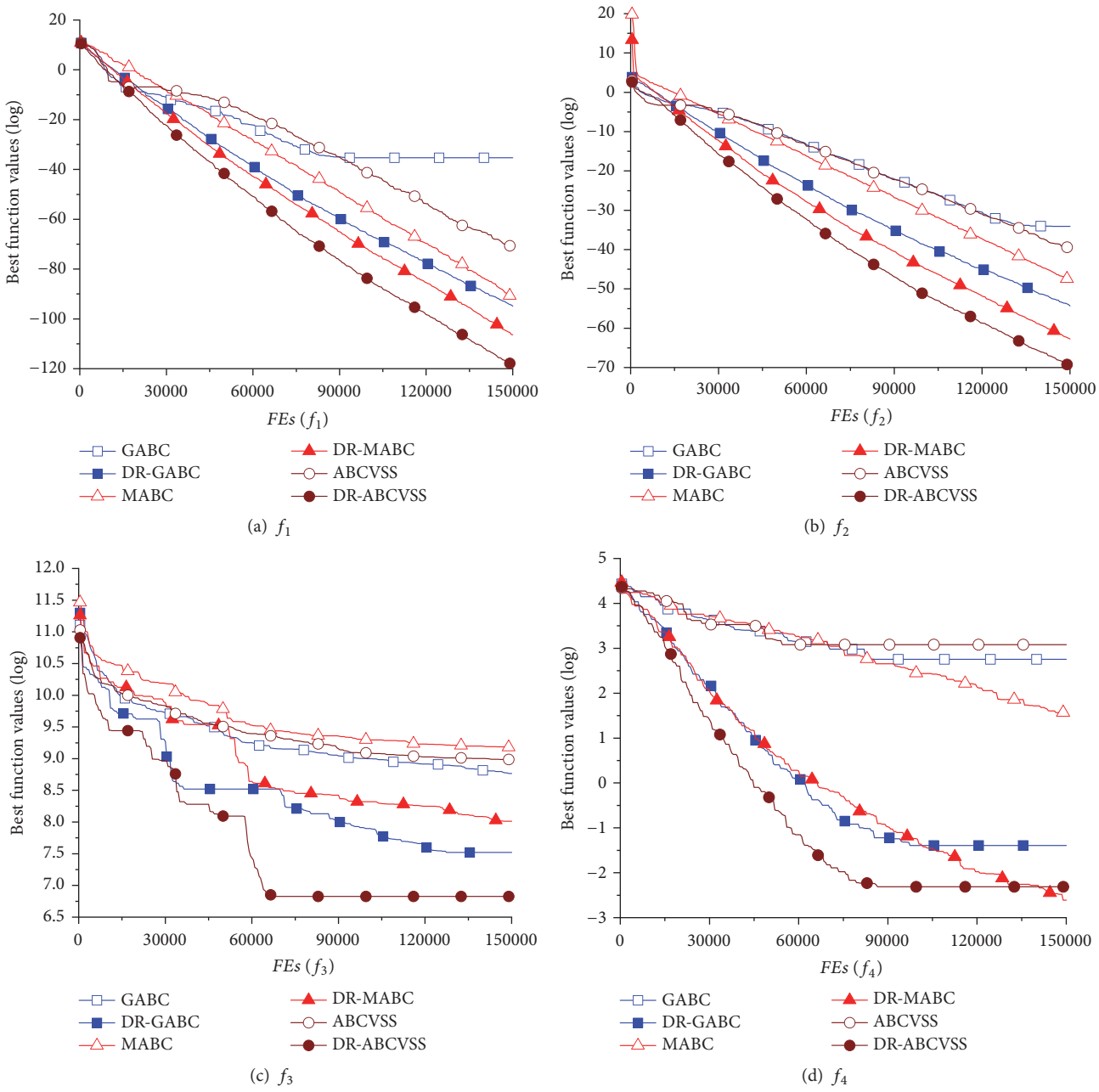

Figure 4: The convergence graphs of different ABC algorithms with the Dynamic Reduction strategy.

the growth of iterations, the number of dimension perturbations dynamically decreases. Less dimension perturbations can reduce the dissimilarities between offspring and their parent solutions. Based on the DR, it can achieve a balance between exploration and exploitation by dynamically changing the number of dimension perturbations. Experiments are carried out on twelve benchmark functions to validate the effectiveness of the DR strategy.

The parameter λ affects the performance of the DR strategy. Experimental results show DR-ABC with different λ results in different performance. Too large or too small λ values may slow down the convergence speed at the beginning search stage. By calculating the mean ranks of multiple λ values, λ=1/5 is considered as the relatively best choice. For all different λ values, DR-ABC outperforms the standard ABC. Results demonstrate the DR strategy is effective for improving the performance of ABC.

For dimension perturbations, there are three different kinds of methods: fixed number of dimension perturbations (FNDP), random number of dimension perturbations (RNDP), and the proposed DR. Results show the DR strategy is better than FNDP and RNDP.

By extending the DR strategy into ABCVSS, GABC, and MABC, we get three new ABC variants, DR-ABCVSS,

DR-GABC, and DR-MABC, respectively. Results show that DR-ABCVSS, DR-GABC, or DR-MABC is better than its corresponding parent algorithm (ABCVSS, GABC, or MABC) and the DR strategy can effectively accelerate their convergence speed.

From the results of DR-ABC with different λ values, we can find that a fixed λ is not effective during the whole search process. For different search stages, different λ values may be needed. In our future work, an adaptive λ strategy will be investigated.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

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

Acknowledgments

This work was supported by the Quality Engineering Project of Anhui Province (no. 2018sxzx38), the Project of Industry-University-Research Innovation Fund (no. 2018A01010), Anhui Provincial Education Department's Excellent Youth Talent Support Program (no. gxyq2017159), the Anhui Provincial Natural Science Foundation (no. 1808085MF202), the Science and Technology Plan Project of Jiangxi Provincial Education Department (no. GJJ170994), and the Anhui Provincial Key Project of Science Research of University (no. KJ2019A0950).

References

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

[2] A. Kumar, D. Kumar, and S. K. Jarial, "A review on artificial bee colony algorithms and their applications to data clustering," *Cybernetics and Information Technologies*, vol. 17, no. 3, pp. 3–28, 2017.

[3] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Applied Mathematics and Computation*, vol. 217, no. 7, pp. 3166–3173, 2010.

[4] D. Karaboga and B. Akay, "A comparative study of artificial Bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[5] X. Cai, X. Z. Gao, and Y. Xue, "Improved bat algorithm with optimal forage strategy and random disturbance strategy," *International Journal of Bio-Inspired Computation*, vol. 8, no. 4, pp. 205–214, 2016.

[6] Y. Wang, P. Wang, J. Zhang et al., "A novel bat algorithm with multiple strategies coupling for numerical optimization," *Mathematics*, vol. 7, no. 2, p. 135, 2019.

[7] Z. Cui, F. Li, and W. Zhang, "Bat algorithm with principal component analysis," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 3, pp. 603–622, 2019.

[8] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, "A hybrid particle swarm optimization algorithm using adaptive learning strategy," *Information Sciences*, vol. 436/437, pp. 162–177, 2018.

[9] F. Wang, H. Zhang, Y. Li, Y. Zhao, and Q. Rao, "External archive matching strategy for MOEA/D," *Soft Computing*, vol. 22, no. 23, pp. 7833–7846, 2018.

[10] Z. Cui, J. Zhang, Y. Wang et al., "A pigeon-inspired optimization algorithm for many-objective optimization problems," *Science China Information Sciences*, vol. 62, Article ID 70212, 2019.

[11] H. Wang, W. Wang, H. Sun, and S. Rahnamayan, "Firefly algorithm with random attraction," *International Journal of Bio-Inspired Computation*, vol. 8, no. 1, pp. 33–41, 2016.

[12] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[13] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, and Y. Li, "A new dynamic firefly algorithm for demand estimation of water resources," *Information Sciences*, vol. 438, pp. 95–106, 2018.

[14] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[15] F. Kang, J. J. Li, and Z. Y. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508–3531, 2011.

[16] W. Gao and S. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, pp. 687–697, 2012.

[17] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[18] H. Wang, Z. J. Wu, X. Y. Zhou, and S. Rahnamayan, "Accelerating artificial bee colony algorithm by using an external archive," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 517–521, IEEE, Cancun, Mexico, June 2013.

[19] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization," *Applied Soft Computing*, vol. 12, no. 1, pp. 320–332, 2012.

[20] 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.

[21] 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.

[22] W. F. Gao, F. T. S. Chan, L. L. Huang, and S. Y. Liu, "Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood," *Information Sciences*, vol. 316, pp. 180–200, 2015.

[23] L. Z. Cui, G. H. Li, Z. X. Zhu et al., "A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization," *Information Sciences*, vol. 414, pp. 53–67, 2017.

[24] L. Cui, G. Li, Q. Lin et al., "A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation," *Information Sciences*, vol. 367-368, pp. 1012–1044, 2016.

[25] G. H. Li, L. Z. Cui, X. H. Fu, Z. K. Wen, N. Lu, and J. Lu, "Artificial bee colony algorithm with gene recombination for

numerical function optimization," *Applied Soft Computing*, vol. 52, pp. 146–159, 2017.

[26] Y. Xiang, Y. Zhou, L. Tang, and Z. Chen, "A decomposition-based many-objective artificial bee colony algorithm," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 287–300, 2019.

[27] T. Dokeroglu, E. Sevinc, and A. Cosar, "Artificial bee colony optimization for the quadratic assignment problem," *Applied Soft Computing*, vol. 76, pp. 595–606, 2019.

[28] H. Ni, Y. Liu, and Y. Fan, "Optimization of injection scheme to maximizing cumulative oil steam ratio based on improved artificial bee colony algorithm," *Journal of Petroleum Science and Engineering*, vol. 173, pp. 371–380, 2019.

[29] J. Chen, W. Yu, J. Tian, L. Chen, and Z. Zhou, "Image contrast enhancement using an artificial bee colony algorithm," *Swarm and Evolutionary Computation*, vol. 38, pp. 287–294, 2018.

[30] V. Pandiri and A. Singh, "An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem," *Applied Soft Computing*, vol. 78, pp. 481–495, 2019.

[31] T. R. Panda and A. K. Swamy, "An improved artificial bee colony algorithm for pavement resurfacing problem," *International Journal of Pavement Research and Technology*, vol. 11, no. 5, pp. 509–516, 2018.

[32] N. Sharma, H. Sharma, and A. Sharma, "Beer froth artificial bee colony algorithm for job-shop scheduling problem," *Applied Soft Computing*, vol. 68, pp. 507–524, 2018.

[33] H. Wang, W. Wang, X. Zhou et al., "Firefly algorithm with neighborhood attraction," *Information Sciences*, vol. 382-383, pp. 374–387, 2017.

[34] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[35] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.