Enhancing Artificial Bee Colony Algorithm with Dynamic Best Neighbor-guided Search Strategy

Qiyu Cai[1], Xinyu Zhou[1, 2] (Corresponding Author), *Member*, *IEEE*, Anquan Jie[1], Maosheng Zhong[1], Mingwen Wang[1], Hui Wang[3], Hu Peng[4]
[1]School of Computer and Information Engineering,
Jiangxi Normal University,
Nanchang 330022, China.
[2]Department of Computer Science,
University of Surrey,
Guildford GU2 7XH, UK.
[3]School of Information Engineering,
Nanchang Institute of Technology,
Nanchang 330099, China.
[4]School of Information Science and Technology,
Jiujiang University,
Jiujiang 332005, China.
Corresponding Email: xyzhou@jxnu.edu.cn

***Abstract*—Artificial bee colony (ABC) algorithm is a relatively new bio-inspired optimization technique which has attracted a lot of attention for its competitive performance. However, in the basic ABC, the solution search equation performs well in exploration but poorly in exploitation, which may cause the problem of slow convergence rate. To tackle this issue, we propose a dynamic best neighbor-guided search strategy to enhance the performance of ABC. In the proposed strategy, a dynamic neighborhood with variable size is first constructed with respect to different evolution stage of the algorithm. After that, the best food source of the neighborhood is selected to guide search instead of only using the global best food source or some elite food sources, which aims to achieve a better balance between the exploration and exploitation. In addition, we design an improved global neighborhood search operator with better robustness to further enhance the performance of ABC. In the experiments, 50 different benchmark functions are used to verify our approach, including the CEC2013 benchmark test suite. The experimental results compared with other four recently well-established ABC variants show that our approach can significantly enhance the performance of ABC.**

Keywords—artificial bee colony, solution search equation, exploration and exploitation, neighborhood search

I. INTRODUCTION

Nowadays, there exist many optimization problems which are often characterized as nonlinear, multimodal or even non-convex in the fields of science research and engineering applications, but unfortunately these problems are hard to solve by some traditional gradient-based optimization methods. Therefore, many researchers have been concentrating on the investigation of bio-inspired optimization methods [1-2], such as genetic algorithm (GA) [3], particle swarm optimization algorithm (PSO) [4-5], differential evolution algorithm (DE) [6], ant colony optimization algorithm (ACO) [7] and artificial bee colony optimization algorithm (ABC) [8]. Compared with other bio-inspired optimization algorithms, in recent years, ABC has received extensive attention from researchers due to its simple structure, easy implementation, yet good performance.

Although ABC has been widely used and showed good performance in many cases, it still has a deficiency concerning on its solution search equation which is good at exploration but poor at exploitation. To date, therefore, a number of improved ABC variants have been proposed. These ABC variants can be roughly divided into two groups, i.e., 1) how to modify the solution search equation, and 2) how to hybridize ABC with other search techniques. More details about the related work will be introduced briefly in the section 2.2. From most of the ABC variants, we observe that they mainly focus on how to utilize the information of the global best food source or some elite food sources. Indeed, the global best food source and elite food sources can enhance the exploitative ability and the convergence rate can be speeded up accordingly. But it should be aware that the ABC algorithm would easily fall into local optimums or become too greedy if these kinds of food sources are not used properly.

To overcome the above potential problem, in this work, we propose a dynamic best neighbor-guided search strategy. In the proposed strategy, we first construct a dynamic neighborhood of which size is adjusted with respect to different evolution stage. At the early evolution stage, the neighborhood size is set to a small value, but its size will gradually increase as the evolution stage proceeds. After that, we select the best food source from the neighborhood as the best neighbor to guide search of new food sources. Note that the best neighbor is dynamic for the variable neighborhood, and it is beneficial to enhance the exploration at the early evolution stage while encourage the exploitation at the late evolution stage. In addition, as the second contribution to this work, we design an improved global neighborhood search operator based on our previous work [9]. In the improved global neighborhood search operator, we put more weight on the randomly selected food sources to increase the robustness of the operator. To verify the performance of our approach, 50 different benchmark functions are used in the experiments, including the 22 basic benchmark functions and the 28 CEC2013 benchmark functions. We compare our approach with four recently well-established ABC variants and the comparative results show that our approach can significantly perform better on the majority of the test functions.

The rest of this paper is organized as follows. In the section 2, the basic ABC algorithm and some related work are briefly introduced. In the section 3, our approach is described in detail. After that, the experiments and analysis are discussed in the section 4. Finally, the conclusion is drawn in the section 5.

II. Basic ABC and Related Work

A. Basic ABC

The ABC algorithm is a bio-inspired optimization method which mimics the intelligent foraging behavior of bee colonies. Its optimization process is divided into four phases, i.e., the initialization phase, the employed bee phase, the onlooker bee phase and the scout bee phase. Similar with other bio-inspired optimization methods, the initialization phase aims to generate an initial population for evolution. In ABC, the population is comprised of food sources which correspond to the candidate solutions to the problem. After the initialization phase, the optimization process enters into the loop of the employed bee phase, the onlooker bee phase and the scout bee phase until the termination condition is satisfied. The details of each phase are described as follows.

Initialization phase: As the first phase of the optimization process, *SN* food sources will be randomly generated according to the following Eq. (1), and *SN* denotes the population size.

$$x_{i,j} = x_j^L + rand_j \cdot \left(x_j^U - x_j^L\right) \tag{1}$$

where $X_i = \left(x_{i,1}, x_{i,2}, \cdots, x_{i,D}\right)$ represents the *i*th food source, and *D* denotes the problem dimension size. x_j^L and x_j^U are the lower and upper bounds for the *j*th dimension, respectively. $rand_j$ is an uniformly distributed random number within the range of [0, 1].

Employed bee phase: In this phase, the employed bees start to search for new food sources by exploring the existing food sources. But note that an existing food source can be explored by one and only one employed bee. Let $V_i = \left(v_{i,1}, v_{i,2}, \cdots, v_{i,D}\right)$ be a new food source, and it can be generated based on the solution search equation listed in the Eq. (2). If the fitness value of V_i is better than its parent food source X_i, then X_i will be replaced by V_i.

$$v_{i,j} = x_{i,j} + \Phi_{i.j} \cdot \left(x_{i,j} - x_{k,j}\right) \tag{2}$$

where X_k is a randomly selected food source from the population and has to be different from X_i. $j \in \{1,2,\cdots,D\}$ is a randomly selected dimension, and $\Phi_{i,j}$ is an uniformly distributed random number within the range of [-1, 1].

Onlooker bee phase: After the employed bees finish their search, they will share the information about the food sources with the onlooker bees. After that, the onlooker bees will continue to search for new food sources by using the same solution search equation shown in the Eq. (2). However, being different from the search pattern of the employed bees, an existing food source can be exploited by more than one onlooker bee. The probability of being exploited for the existing food source X_i, it is related to the nectar amount and can be calculated by the following Eq. (3).

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{3}$$

where p_i is the probability and fit_i denotes the fitness value of X_i. As seen, if a food source has good fitness value, the probability of being selected would be high. This also implies that a food source with poor nectar amount may never be exploited by any onlooker bee. Being similar with the employed bee phase, the one with better fitness value can survive between the existing food source and the new food source.

Scout bee phase: If a food source cannot be consecutively improved for at least *limit* times, it will be considered to be exhausted. As a result, the corresponding employed bee will transform into a scout bee, and the Eq. (1) is used to reset the exhausted food source. Note that the control parameter *limit* is the single parameter needed to preset except for the population size.

B. Related work

Although ABC has shown good performance in many cases, its performance is still challenged for some complicated problems, such as the kind of multimodal problems. In the last few years, therefore, many improved ABC variants have been proposed which can be roughly classified into two groups. The first group focuses on the modification of solution search equation, while the second one pays close attention to hybridize ABC with other search techniques. Both of these two groups are briefly introduced as follows.

The solution search equation in the basic ABC has strong exploration but weak exploitation due to the randomly selected food source. Thus many modified solution search equations include the global best food source or some elite food sources to enhance the exploitation. For instances, being inspired by the PSO algorithm, Zhu and Kwong [10] developed a gbest-guided ABC (GABC) in which the global best food source is added into the solution search equation as a guiding term. Based on the classic mutation strategy of DE, Gao and Liu [11] proposed a modified ABC (MABC) by designing a new solution search equation ABC/best/1. Gao et al. [12] presented a novel solution search equation in the their proposed CABC which likes the crossover operation of GA and has no bias to any search direction. Recently, Zhou et al. [13] proposed a Gaussian bare-bones ABC (GBABC) which is characterized by the new Gaussian bare-bones solution search equation. In this new search equation, the global best food source is utilized as well, and the reported experimental results showed the high efficiency of GBABC. To overcome the issue of slow convergence rate of ABC, Cui et al. [14] introduced a depth-first search framework and proposed an elite-guided solution search equation by utilizing some elite food sources. With the concept of neighborhood, Peng et al. [15] proposed a best neighbor-guided solution search strategy (NABC) in which the best food source out of five neighbors is utilized.

Hybridization of ABC with other search techniques is another hot research topic in the research field of ABC community. For examples, Kang et al. [16] proposed a Rosenbrock ABC in which the rotational direction method is used to complement the exploitation as a local search tool. Xiang et al. [17] used the chaotic map technique in both of the initialization phase and scout bee phase to enhance the effectiveness of ABC. Based on the technique of orthogonal experimental design, Gao et al. [12] developed an orthogonal learning strategy for ABC to discover more useful information from the search experience. Zhou et al. [9] proposed a modified ABC by integrating a global neighborhood search operator within the search framework of MABC to better balance the exploration and exploitation. Sharma and Pant [18] designed a new method to initialize food sources by using the opposition-based learning technique.

III. Our Approach

A. Dynamic Best Neighbor-guided Search Strategy

Similar to many other bio-inspired optimization algorithms, ABC also suffers from the problem of slow convergence rate for some complicated problems. The main reason is that the solution search equation does well in exploration but poorly in exploitation for the strong robustness of the randomly selected food source. It is well known that keeping a good balance between the exploration and exploitation is the key point for bio-inspired optimization algorithms to have promising performance. Therefore, many improved ABC variants have focused on how to enhance the exploitation for the solution search equation by utilizing the global best food source or elite food sources, which have been briefly reviewed in the section 2 about the related work. Without questions, the exploitation of ABC can be enhanced by utilizing the global best food source and elite food sources, which has also been proven by the extensive experimental results of some related work.

However, it should be aware that the global best food source and elite food sources can not only enhance the exploitation but also easily cause premature convergence. Because for some complicated problems, its fitness landscape is usually rugged with numerous local optima, so the global best food source and elite food sources may locate on the local optima and will misguide the entire population into the same local optima. In such scenario, the population will tend to stop evolving and be stagnant. To avoid this, a strategy of how to properly use the global best food source or elite food sources is required. Following this idea, we propose a dynamic best neighbor-guided search strategy based on the concept of dynamic neighborhood. In the proposed strategy, we first construct a neighborhood of the parent food source with variable neighborhood size. Then we select the best food source from the neighborhood as a guiding food source to generate the new food source, which can be formulated by the Eq. (4) as follows.

$$v_{i,j} = x_{nbest,j} + \Phi_{i.j} \cdot \left(x_{nbest,j} - x_{k,j}\right) \tag{4}$$

where V_i is the new food source that corresponds to its parent food source X_i. X_k is a randomly selected food source from the entire population and has to be different from X_i. X_{nbest} is the best food source of the neighborhood of X_i.

As for the construction of the neighborhood, there exist many ways which can be roughly categorized into two kinds. The first kind is usually based on the individual index, while the second kind often employs distance-based metric, such as the Euclidean distance. To keep the proposed strategy simple but efficient, we follow the first kind and randomly select *M* food sources from the entire population as the neighbors of X_i. The parameter *M* is the neighborhood size which represents the number of neighbors that X_i has, and it is defined as follows.

$$M = \left\lceil \left(\frac{FEs}{MaxFEs}\right) \cdot SN \right\rceil \tag{5}$$

where $\lceil \cdot \rceil$ represents the ceiling function, *FEs* is the used number of fitness function evaluations, *MaxFEs* denotes the maximum number of fitness function evaluations, and *SN* is the number of food sources. It is necessary to point out that the way of constructing the neighborhood is similar with that of the NABC [15] which has been briefly introduced in the section 2. But it is different from the NABC that the neighborhood size is dynamic in our approach. To some extent, our approach can be considered as an improved version of the NABC.

From the Eq. (5), it can be seen that at the early stage of the evolution process the parameter *M* has a small value, but its value will gradually increases as the evolution process continues. This implies that the number of the neighbors will increase as the evolution process proceeds, and the best food source of the neighborhood tends to be the global best food source of the entire population at the late evolution stage. Therefore, the modified solution search equation in the proposed strategy will has strong exploration at the early evolution stage, but it will transform into strong exploitation at the late evolution stage. In fact, the search behavior of the modified solution search equation meets the requirements for the bio-inspired optimization algorithms. The requirements often expect strong exploration to discover more new areas in the search space at the beginning of evolution process, but as the evolution process reaches the end, the strong exploitation is expected to speed up the convergence rate.

B. Improved Global Neighborhood Search Operator

To further enhance the performance of ABC, we design an improved global neighborhood search operator. The motivation is that if an individual unfortunately trapped into a local optimum, it is hard for the individual to jump out the local optimum. But on the bright side, a local optimum usually implies that it has a relatively good fitness value and a better solution or even the global optimum solution may exist within or near the neighborhood of the individual. So, if the neighborhood can be searched with small steps, it would increase the probability of finding a better solution or the global optimum solution. Following this idea, we design an improved global neighborhood search operator based on our previous work [9]. In the work [9], we introduced a global neighborhood search operator to enhance the performance of ABC (MABC-NS) as a local search technique which can be expressed as follows.

$$TX = r_1 \cdot X_i + r_2 \cdot gbest + r_3 \cdot (X_a - X_b) \tag{6}$$

TABLE I. THE 22 BASIC BENCHMARK FUNCTIONS

Function	Name	Search range	Global optimum
F01	Sphere	[-100, 100]	0
F02	Schwefel2.22	[-10, 10]	0
F03	Schwefel1.2	[-100, 100]	0
F04	Schwefel2.21	[-100, 100]	0
F05	Rosenbrock	[-5, 10]	0
F06	Step	[-100, 100]	0
F07	Quartic with noise	[-1.28, 1.28]	0
F08	Elliptic	[-100, 100]	0
F09	SumSquare	[-10, 10]	0
F10	SumPower	[-1, 1]	0
F11	Exponential	[-10, 10]	$-418.98\cdot D$
F12	Schwefel2.26	[-500, 500]	0
F13	Rastrigin	[-5.12, 5.12]	0
F14	Ackley	[-50, 50]	0
F15	Griewank	[-600, 600]	0
F16	Penalized1	[-100, 100]	0
F17	Penalized2	[-100, 100]	0
F18	NCRastrigin	[-5.12, 5.12]	0
F19	Alpine	[-10, 10]	0
F20	Levy	[-10, 10]	0
F21	Bohachevsky2	[-100, 100]	0
F22	Weierstrass	[-1, 1]	0

TABLE II. PARAMETER SETTINGS OF THE FOUR COMPARED ABC VARIANTS

Algorithm	Parameter settings
CABC	$limit = SN\cdot D$
DFSABC_elite	$limit = SN\cdot D$, $p = 0.1$, $r = 1/p$
MABC-NS	$limit = 100$, $p = 0.1$
NABC	$limit = 50$, $N = 5$

where X_i is the current food source, TX is the trial solution for X_i generated by the global neighborhood search operator. *gbest* is the global best food source of the entire population. X_a and X_b are two randomly selected food sources from the entire population and they have to be different from X_i. r_1, r_2 and r_3 are three mutually exclusive numbers randomly chosen from the range (0, 1), and they have to meet the condition: $r_2 + r_2 + r_3 = 1$.

Although the global neighborhood search operator has shown competitive performance for the MABC-NS, it seems to be too greedy for our approach due to the utilization of *gbest* based on our preliminary experiments. To tackle this problem, we design an improved version by making a small modification based on the randomly selected food sources, and the improved version is listed as follows.

$$TX = r_1 \cdot \frac{X_i + X_a}{2} + r_2 \cdot \frac{gbest + X_b}{2} + r_3 \cdot (X_a - X_b) \quad (7)$$

where the variables are the same with those of the Eq. (6). As seen, the improved global neighborhood search operator almost has the same structure with the original one, but it puts more weight on the two randomly selected food sources that can effectively enhance the robustness or diversity.

C. *Complete Procedure of Our Approach*

To enhance the performance of ABC, we propose two strategies, i.e., 1) the dynamic best neighbor-guided search strategy (can be considered the main contribution to this work), and 2) the improved global neighborhood search operator. Being different from the algorithmic structure of classic ABC, we employ a more brief structure to combine the two proposed strategies to maximize the effectiveness. In this structure, the employed bee phase, the onlooker bee phase and the scout bee phase are merged into one phase, and the dynamic best neighbor-guided search strategy is used in this phase. After the execution of this phase, the improved global neighborhood search operator is trigged with a predefined probability 0.1, and this is kept the same with the MABC-NS. Much more details can be referred to the Algorithm 1, in there, *FEs* is the used number of fitness function evaluations, *MaxFEs* denotes the maximum number of fitness function evaluations, and *SN* is the number of food sources.

Algorithm 1. The complete procedure of our approach

```
Randomly generate SN food sources as an initial population according
to the Eq. (1) and calculate their fitness values, FEs=SN;
while FEs ≤ MaxFEs do
    for i=1 to SN do
        Generate a new food source V_i according to the Eq. (4);
        Evaluate V_i and set FEs = FEs + 1
        if f(V_i) ≤ f(X_i) do
            Replace X_i with V_i
        end if
    end for
  // The improved global neighborhood search operator
    for i=1 to SN do
        Generate a random number rand∈[0, 1];
            if rand ≤ 0.1 then
                Generate a trial solution TX according to the Eq. (7);
                Evaluate TX and set FEs = FEs + 1
                if f(TX) ≤ f(X_i) then
                    Replace X_i with TX
                end if
            end if
    end for
end while
```

IV. EXPERIMENTAL VERIFICATIONS

A. *Benchmark Functions and Parameter Settings*

To verify the performance of our approach, abbreviated as DABC, 50 different benchmark functions are used which include the 22 basic benchmark functions and the 28 CEC2013 benchmark functions. As for the 22 basic benchmark functions listed in the table 1, they are widely used to verify different bio-inspired optimization algorithms [9], [11], [13], [18], and the first 11 ones are unimodal types while the remaining ones are multimodal types. For the 28 CEC2013 benchmark functions, they are almost rotated and composition types and much more difficult to solve than the 22 basic benchmark functions. Due to the limited paper space, the definition details of the 28 CEC2013 benchmark functions are not given here, and they can be referred to the work [19].

TABLE III. COMPARISONS WITH OTHER FOUR ABC VARIANTS ON THE 22 BASIC BENCHMARK FUNCTIONS FOR *D*=30

Function	CABC	DFSABC_elite	MABC-NS	NABC	DABC
F01	1.95E−50±2.35E−50+	3.98E−83±8.90E−83+	9.76E−110±2.29E−109+	8.69E−39±2.30E−38+	**1.22E−137±2.29E−137**
F02	8.96E−27±4.49E−27+	4.11E−43±3.91E−43+	7.45E−56±7.92E−56+	1.35E−20±5.55E−21+	**8.87E−70±2.03E−69**
F03	1.24E+04±1.60E+03+	4.89E+03±2.20E+03+	2.84E−85±3.43E−85+	4.06E+03±1.24E+03+	**6.72E−115±1.06E−114**
F04	1.87E+01±4.20E+00+	1.34E+01±1.48E+00+	2.47E−42±2.02E−42+	3.43E−01±8.10E−02+	**9.14E−57±1.23E−56**
F05	2.95E+00±3.18E+00−	7.73E+00±2.20E+01−	2.83E+01±1.68E−01=	**2.66E−01±8.79E−01−**	2.86E+01±1.44E−01
F06	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F07	5.96E−02±8.82E−03+	3.81E−02±9.47E−03+	2.58E−04±1.41E−04+	2.56E−03±9.49E−04+	**1.77E−04±5.81E−05**
F08	3.69E−42±7.57E−42+	7.53E−80±1.05E−79+	1.31E−105±3.71E−105+	1.89E−32±2.36E−32+	**1.78E−133±2.79E−133**
F09	1.05E−51±1.71E−51+	8.59E−85±1.37E−84+	7.18E−111±1.01E−110+	6.05E−41±7.11E−41+	**4.21E−137±1.19E−136**
F10	1.46E−29±2.73E−29+	8.19E−46±2.44E−45+	3.54E−106±7.68E−106+	4.34E−38±8.58E−38+	**6.33E−141±1.68E−140**
F11	4.59E−06±5.92E−06+	2.44E−06±2.54E−06+	**0.00E+00±0.00E+00=**	8.04E−13±7.95E−13+	**0.00E+00±0.00E+00**
F12	**3.82E−04±8.34E−13=**	**3.82E−04±0.00E+00=**	2.37E+01±4.74E+01+	**3.82E−04±8.68E−13=**	**3.82E−04±0.00E+00**
F13	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F14	3.06E−14±3.28E−15+	2.82E−14±2.66E−15+	7.99E−16±1.07E−15+	5.95E−15±1.77E−15+	**4.44E−16±0.00E+00**
F15	7.60E−13±2.11E−12+	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F16	**1.57E−32±2.74E−48=**	**1.57E−32±2.74E−48=**	**1.57E−32±2.74E−48=**	**1.57E−32±2.74E−48=**	**1.57E−32±2.74E−48**
F17	**1.35E−32±2.74E−48=**	**1.35E−32±2.74E−48=**	**1.35E−32±2.74E−48=**	**1.35E−32±2.74E−48=**	**1.35E−32±2.74E−48**
F18	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F19	8.73E−28±9.35E−28+	2.72E−16±5.70E−16+	2.66E−55±7.75E−55+	3.61E−09±7.62E−09+	**1.00E−70±1.50E−70**
F20	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00**
F21	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F22	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
+/−/=	12/1/9	11/1/10	11/0/11	11/1/10	−−

TABLE IV. COMPARISONS WITH OTHER FOUR ABC VARIANTS ON THE 22 BASIC BENCHMARK FUNCTIONS FOR *D*=50

Function	CABC	DFSABC_elite	MABC-NS	NABC	DABC
F01	6.30E−50±8.70E−50+	5.25E−83±4.73E−83+	2.16E−160±5.43E−160+	3.99E−38±5.02E−38+	**3.44E−206±0.00E+00**
F02	3.68E−26±2.00E−26+	7.42E−43±2.97E−43+	6.18E−82±4.77E−82+	1.04E−19±3.35E−20+	**2.42E−105±4.81E−105**
F03	4.14E+04±4.33E+03+	2.15E+04±6.30E+03+	3.38E−142±6.88E−142+	7.30E+03±2.97E+03+	**7.52E−192±0.00E+00**
F04	4.93E+01±2.26E+00+	4.23E+01±3.61E+00+	1.74E−70±1.08E−70+	3.57E−02±1.29E−02+	**5.48E−94±8.71E−94**
F05	4.07E+00±4.34E+00−	8.35E+00±2.42E+01−	4.85E+01±1.56E−01=	**1.12E+00±2.21E+00−**	4.87E+01±2.56E−03
F06	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F07	1.80E−01±2.75E−02+	1.23E−01±2.72E−02+	1.41E−04±6.98E−05+	1.35E−03±4.67E−04+	**8.56E−05±5.96E−05**
F08	2.62E−42±6.33E−42+	1.65E−79±3.45E−79+	3.67E−156±1.09E−155+	1.09E−30±1.42E−30+	**4.67E−203±0.00E+00**
F09	1.19E−50±1.87E−50+	1.26E−83±1.38E−83+	2.47E−161±6.52E−161+	4.75E−39±1.03E−38+	**7.91E−207±0.00E+00**
F10	4.69E−25±9.31E−25+	9.82E−32±1.77E−31+	8.77E−147±1.22E−146+	9.18E−38±3.54E−37+	**8.47E−196±0.00E+00**
F11	2.45E−06±2.62E−06+	2.51E−06±1.89E−06+	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F12	**6.36E−04±7.71E−12=**	**6.36E−04±0.00E+00=**	**6.36E−04±1.09E−12=**	**6.36E−04±1.59E−12=**	**6.36E−04±0.00E+00**
F13	3.55E−16±1.07E−15+	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F14	6.65E−14±4.26E−15+	5.76E−14±5.61E−15+	**4.44E−16±0.00E+00=**	4.71E−15±1.42E−15+	**4.44E−16±0.00E+00**
F15	2.61E−12±5.86E−12+	8.11E−14±2.43E−13+	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F16	**9.42E−33±1.37E−48=**	**9.42E−33±1.37E−48=**	**9.42E−33±1.37E−48=**	**9.42E−33±2.74E−48=**	**9.42E−33±1.37E−48**
F17	**1.35E−32±2.74E−48−**	**1.35E−32±2.74E−48−**	2.20E−03±4.39E−03=	**1.35E−32±2.74E−48−**	6.59E−03±1.32E−02
F18	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F19	3.92E−27±6.68E−27+	2.66E−16±4.60E−16+	1.53E−82±3.38E−82+	9.00E−08±1.23E−07+	**2.04E−105±4.05E−105**
F20	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00=**	**1.35E−31±0.00E+00**
F21	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
F22	1.42E−14±1.27E−14+	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00=**	**0.00E+00±0.00E+00**
+/−/=	14/2/6	12/2/8	9/0/13	10/2/10	−−

TABLE V. AVERAGE RANKINGS OF THE FIVE INVOLVED ABC VARIANTS ON THE 22 BASIC BENCHMARK FUNCTIONS FOR *D*=30 AND 50.

Algorithm	*D*=30	*D*=50
CABC	3.78	3.86
DFSABC_elite	3.20	3.32
MABC-NS	2.57	2.48
NABC	3.34	3.18
DABC	**2.16**	**2.21**

TABLE VI. COMPARISONS WITH OTHER FOUR ABC VARIANTS ON THE 28 CEC2013 BENCHMARK FUNCTIONS FOR *D*=30

Function	CABC	DFSABC_elite	MABC-NS	NABC	DABC
F01	**2.27E−13±0.00E+00−**	2.50E−13±6.82E−14−	5.00E−13±9.09E−14+	4.32E−13±6.82E−14=	4.77E−13±6.82E−14
F02	2.41E+07±4.62E+06+	1.35E+07±3.40E+06+	5.21E+06±7.44E+05+	6.99E+06±1.36E+06+	**4.83E+06±1.26E+06**
F03	2.11E+09±1.04E+09+	2.30E+08±1.50E+08−	9.73E+08±7.91E+08+	**1.07E+08±6.12E+07−**	9.16E+08±6.81E+08
F04	6.74E+04±7.24E+03+	6.76E+04±8.10E+03+	4.85E+04±4.14E+03+	**4.67E+04±5.51E+03=**	**4.63E+04±5.05E+03**
F05	4.77E−13±8.51E−14=	5.41E−13±5.08E−14+	4.89E−13±5.21E−14+	5.91E−13±6.82E−14+	**4.57E−13±4.57E−14**
F06	2.89E+01±6.49E+00−	**1.66E+01±2.34E+00−**	4.97E+01±2.72E+01=	2.69E+01±1.04E+01−	4.72E+01±2.81E+00
F07	1.01E+02±1.59E+01+	1.05E+02±1.02E+01+	8.90E+01±1.13E+01=	**7.90E+01±8.47E+00−**	9.25E+01±1.32E+01
F08	**2.10E+01±3.44E−02=**	**2.09E+01±3.61E−02=**	**2.09E+01±5.63E−02=**	**2.10E+01±3.01E−02=**	**2.09E+01±3.25E−02**
F09	3.03E+01±1.48E+00+	2.88E+01±1.72E+00+	**2.54E+01±2.36E+00=**	2.88E+01±2.16E+00+	**2.55E+01±3.17E+00**
F10	9.38E+00±1.41E+00+	1.34E+00±2.19E−01+	**2.94E−01±1.59E−01=**	8.05E−01±2.56E−01+	**3.08E−01±1.32E−01**
F11	**4.55E−14±2.27E−14−**	5.12E−14±1.71E−14−	3.98E−01±6.60E−01−	5.68E−14±0.00E+00−	1.79E+00±1.39E+00
F12	1.75E+02±7.67E+00+	1.22E+02±2.23E+01+	1.97E+02±3.87E+01+	1.19E+02±2.17E+01+	**1.15E+02±2.77E+01**
F13	2.02E+02±1.31E+01−	1.70E+02±2.05E+01−	2.69E+02±4.25E+01−	**1.64E+02±2.02E+01−**	3.44E+02±3.03E+01
F14	8.19E−01±6.40E−01+	1.26E+00±1.05E+00+	1.32E+01±3.64E+01+	2.31E−01±4.33E−02+	**8.33E−02±1.21E−02**
F15	5.67E+03±1.22E+02+	4.61E+03±3.79E+02+	3.69E+03±4.55E+02+	3.74E+03±3.16E+02+	**3.16E+03±3.93E+02**
F16	2.22E+00±2.65E−01+	2.09E+00±1.81E−01+	1.17E+00±2.27E−01+	1.15E+00±1.03E−01+	**1.12E+00±2.05E−01**
F17	**3.05E+01±1.80E−02=**	**3.05E+01±2.44E−02=**	**3.05E+01±5.04E−02=**	**3.04E+01±3.75E−03=**	**3.05E+01±5.46E−02**
F18	**3.00E+01±2.79E−14=**	**3.00E+01±2.61E−14=**	**3.02E+01±3.23E−01=**	**3.00E+01±2.79E−14=**	**3.01E+01±2.42E−01**
F19	2.31E−02±2.26E−02−	**1.21E−01±1.28E−01−**	1.45E+00±4.43E−01=	2.85E−01±1.01E−01−	1.47E+00±8.13E−01
F20	1.23E+01±2.22E−01+	1.19E+01±4.44E−01+	**1.04E+01±5.69E−01=**	1.17E+01±3.71E−01+	**1.05E+01±2.42E−01**
F21	**3.00E+02±1.25E−13=**	**3.00E+02±6.32E+01=**	**3.00E+02±4.90E+01=**	**3.00E+02±3.00E+01=**	**3.00E+02±4.00E+01**
F22	1.13E+02±2.08E+00+	1.09E+02±4.57E+01+	9.10E+01±5.29E+01+	1.03E+02±1.80E+01+	**8.04E+01±1.43E+00**
F23	6.55E+03±2.40E+02+	5.11E+03±3.72E+02=	**4.34E+03±6.66E+02−**	5.10E+03±3.95E+02=	5.12E+03±3.95E+02
F24	2.79E+02±5.20E+00+	2.78E+02±4.94E+00+	**2.56E+02±5.01E+00=**	**2.54E+02±1.48E+01=**	**2.55E+02±9.88E+00**
F25	2.99E+02±4.62E+00+	3.00E+02±5.92E+00+	**2.80E+02±1.44E+01=**	2.98E+02±9.06E+00+	**2.78E+02±1.68E+01**
F26	2.02E+02±3.52E−01+	2.01E+02±1.18E−01+	**2.00E+02±2.99E−02=**	**2.00E+02±5.18E−02=**	**2.00E+02±2.86E−02**
F27	4.17E+02±1.03E+01−	**4.00E+02±2.66E−01−**	8.76E+02±5.66E+01=	5.28E+02±2.56E+02−	8.99E+02±6.55E+01
F28	3.00E+02±1.55E−13−	**2.97E+02±9.97E+00−**	2.34E+03±7.84E+02=	3.00E+02±1.37E−13−	2.26E+03±7.89E+02
+/−/=	16/7/5	15/8/5	10/3/15	11/8/9	−−

The following experiments comprise two parts, i.e., 1) test on the 22 basic benchmark functions, and 2) test on the 28 CEC2013 benchmark functions. All of the benchmark functions are tested on two kinds of dimension sizes: *D*=30 and *D*=50. The corresponding maximum number of fitness function evaluations *MaxFEs* is set to $5000 \cdot D$. Four recently well-established ABC variants are compared with our approach, i.e., 1) CABC [12], 2) DFSABC_elite [13], 3) MABC-NS [9], and 4) NABC [15]. All of these four ABC variants have been briefly introduced in the previous section 2. To make a fair comparison, all of the involved algorithms have the same setting for the number of food sources *SN*=50. As for other particular parameter settings, they are kept the same with the original paper, which are also listed in the table 2. After each algorithm has run 30 times per function, we record the average best fitness values and standard deviations as the final results.

B. Test on the 22 Basic Benchmark Functions

The final results on the 22 basic benchmark functions are shown in the tables 3 and 4 for *D*=30 and *D*=50, respectively. Furthermore, to make the comparison more reasonable, the Wilcoxon signed-ranked test at α=0.05 is employed, and the markers “+”, “−”, and “=” indicate our approach is significantly better than, worse than, and equal to the corresponding competitive ABC variant. The summarized comparison results are listed in the last row of the table.

As seen, from the table 3 for *D*=30, the proposed DABC is superior to the other four ABC variants on the majority of test functions. To be specific, DABC performs better than CABC, DFSABC_elite, MABC_NS and NABC on 12, 11, 11 and 11 test functions, respectively. It is impressive that DABC can obtain the best results on 21 out of the 22 test functions, and only lose one on the Rosenbrock function (F05). As for *D*=50, it can be seen from the table 4 that DABC still achieves the best performance among the five involved ABC variants, and this shows that the DABC has good robustness. To make the comparison results more intuitive, we take the Friedman test to further compare the involved algorithms, the average rankings of the five involved ABC variants for both D=30 and D=50 are shown in the table 5, and the best rankings are marked in **bold** face. As seen, DABC is ranked in the first order for D=30 and D=50.

C. Test on the 28 CEC2013 Benchmark Functions

Compared with the 22 basic benchmark functions, the 28 CEC2013 benchmark functions are much more complicated which include the rotated and composition functions. So, the test on the 28 CEC2013 benchmark functions can make the comparison more convincing. According to the guidelines of using the CEC2013 test suite [19], the maximum number of fitness function evaluations *MaxFEs* is set to $10000 \cdot D$, while the other parameter settings are kept the same with the section 4.2.

The final results are given in the tables 6 and 7 for D=30 and 50, respectively. As seen, for D=30 in the table 6, DABC performs best among the five involved algorithms. Concretely, DABC obtains better results than CABC, DFSABC_elite, MABC-NS and NABC on 16, 15, 10 and 11 test functions, respectively. But it is also worse than the corresponding competitor on only 7, 8, 3 and 8 test functions, respectively. As the dimension size increases to 50, we can get the similar results that DABC is significantly better than its competitors on the majority of test functions. The results of the Friedman test for the 28 CEC2013 benchmark functions are shown in the table 8, it can be seen that DABC achieves the first order for both D=30 and 50.

TABLE VII. COMPARISONS WITH OTHER FOUR ABC VARIANTS ON THE 28 CEC2013 BENCHMARK FUNCTIONS FOR D=50

Function	CABC	DFSABC_elite	MABC-NS	NABC	DABC
F01	4.09E−13±9.09E−14+	6.59E−13±6.82E−14+	**2.69E−13±1.36E−13=**	9.32E−13±1.22E−13+	**2.66E−13±1.22E−13**
F02	5.13E+07±5.70E+06+	3.22E+07±8.70E+06+	5.40E+06±1.30E+06+	8.16E+06±2.42E+06+	**5.21E+06±1.31E+06**
F03	1.52E+10±2.85E+09+	2.16E+09±8.31E+08+	1.12E+09±1.01E+09=	**5.95E+08±3.66E+08−**	1.01E+09±1.09E+09
F04	1.29E+05±9.85E+03+	1.24E+05±1.37E+04+	6.70E+04±6.90E+03+	7.23E+04±5.76E+03+	**5.90E+04±7.44E+03**
F05	1.25E−12±1.02E−13+	9.66E−13±9.17E−14+	1.15E−12±1.07E−13+	1.13E−12±9.44E−14+	**1.01E−12±1.29E−13**
F06	4.65E+01±3.28E−01−	**4.45E+01±1.64E+00−**	8.98E+01±3.53E+01+	7.61E+01±6.76E−01+	6.92E+01±2.30E+01
F07	1.56E+02±8.32E+00+	1.44E+02±1.04E+01+	1.06E+02±8.42E+00−	**9.03E+01±9.36E+00−**	1.22E+02±1.10E+01
F08	**2.11E+01±3.04E−02=**	**2.11E+01±2.41E−02=**	**2.11E+01±2.66E−02=**	**2.11E+01±4.63E−02=**	**2.11E+01±2.98E−02**
F09	5.89E+01±1.58E+00+	5.83E+01±1.46E+00+	4.91E+01±1.89E+00+	5.83E+01±4.14E+00+	**4.65E+01±4.29E+00**
F10	3.32E+01±7.91E+00+	2.20E+00±2.65E−01+	9.04E−01±1.80E−01+	1.33E+00±1.53E−01+	**7.12E−01±4.38E−01**
F11	**8.51E−14±3.14E−14−**	**8.53E−14±2.84E−14−**	1.89E+00±1.37E+00−	1.02E−13±2.27E−14−	5.87E+00±5.38E+00
F12	5.07E+02±4.04E+01−	3.98E+02±5.88E+01−	5.50E+02±6.17E+01−	**2.69E+02±3.64E+01−**	6.15E+02±5.87E+01
F13	5.57E+02±3.68E+01−	5.11E+02±3.18E+01−	6.38E+02±5.61E+01−	**3.52E+02±4.80E+01−**	7.67E+02±5.04E+01
F14	3.22E+00±1.29E+00+	7.44E+00±4.96E+00+	2.55E+00±1.84E+00+	**1.13E+00±3.31E−01=**	**1.13E+00±8.43E−01**
F15	1.11E+04±5.08E+02+	9.57E+03±5.21E+02+	7.74E+03±5.58E+02+	7.67E+03±6.71E+02+	**6.96E+03±6.46E+02**
F16	2.96E+00±3.00E−01+	2.88E+00±1.64E−01+	1.74E+00±3.56E−01+	1.61E+00±1.47E−01+	**1.45E+00±3.45E−01**
F17	**5.09E+01±5.31E−02=**	**5.09E+01±3.58E−02=**	**5.09E+01±1.34E−01=**	**5.08E+01±1.09E−02=**	**5.09E+01±5.98E−02**
F18	**5.02E+01±3.77E−14=**	**5.02E+01±4.55E−14=**	**5.02E+01±5.52E−01=**	**5.02E+01±3.42E−14=**	**5.02E+01±7.04E−01**
F19	4.35E−02±3.14E−02−	**2.28E−01±1.74E−01−**	3.74E+00±1.03E+00=	5.93E−01±1.60E−01−	3.72E+00±1.52E+00
F20	2.17E+01±1.64E−01+	2.13E+01±3.17E−01+	1.94E+01±5.41E−01+	2.08E+01±3.31E−01+	**1.88E+01±6.28E−01**
F21	**3.00E+02±2.52E−13−**	3.07E+02±3.28E+01−	3.40E+02±4.90E+01=	**3.00E+02±1.14E−13−**	3.40E+02±4.90E+01
F22	1.80E+01±2.18E+00+	4.87E+01±7.05E+01+	7.39E+01±8.75E+01+	1.40E+01±7.81E−01+	**9.47E+00±3.49E+00**
F23	1.29E+04±4.33E+02+	1.11E+04±1.50E+03+	8.76E+03±7.90E+02+	9.66E+03±8.06E+02+	**8.53E+03±6.21E+02**
F24	3.60E+02±3.79E+00+	3.60E+02±4.72E+00+	**3.28E+02±1.51E+01=**	**3.27E+02±1.64E+01=**	**3.27E+02±1.24E+01**
F25	**3.99E+02±6.49E+00=**	**4.00E+02±5.27E+00=**	**3.93E+02±1.51E+01=**	**3.98E+02±8.95E+00=**	**3.99E+02±1.14E+01**
F26	**2.04E+02±7.78E−01+**	**2.03E+02±6.01E−01+**	**2.01E+02±1.87E−01=**	**2.01E+02±1.70E−01=**	**2.01E+02±8.35E−02**
F27	5.52E+02±5.78E+01−	**5.51E+02±4.35E+02−**	1.73E+03±1.20E+02+	1.57E+03±5.90E+02+	1.22E+03±1.43E+02
F28	**4.00E+02±2.05E−13−**	**4.00E+02±3.52E−13−**	4.92E+03±1.80E+03+	**4.00E+02±3.29E−13−**	4.57E+03±4.80E+02
+/−/=	16/8/4	16/8/4	14/4/10	13/8/7	−−

TABLE VIII. AVERAGE RANKINGS OF THE FIVE INVOLVED ABC VARIANTS ON THE 28 CEC2013 BENCHMARK FUNCTIONS FOR *D*=30 AND 50.

Algorithm	*D*=30	*D*=50
CABC	3.64	3.68
DFSABC_elite	3.02	3.20
MABC-NS	3.11	3.17
NABC	2.74	2.67
DABC	**2.47**	**2.38**

V. CONCLUSIONS

As a relatively new bio-inspired optimization algorithm, ABC has attracted a lot of attention for its simple structure, easy implementation, yet good performance. For some complicated optimization problems, however, its performance is also challenged. The main issue limiting its performance is that the solution search equation is good at exploration but poor at exploitation. To solve this problem, in this work, we propose two strategies, i.e., 1) the dynamic best neighbor-guided search strategy, and 2) the improved global neighborhood search operator.

As for the dynamic best neighbor-guided search strategy, we first construct a dynamic neighborhood with variable size, and then select the best food source from the neighborhood to guide the search of new food sources. Furthermore, the neighborhood size is adjusted dynamically according to different evolution phase of the algorithm. But being different from utilizing the global best food source or some elite food sources, only the best food source of the neighborhood is utilized in the dynamic best neighbor-guided search strategy, which is beneficial to keep a better balance between the exploration and exploitation.

In the improved global neighborhood search operator, we modify a neighborhood search operator by putting more weight on the randomly selected food sources, which is good to enhance the robustness. To verify our approach, 50 different benchmark functions are used in the experiments, including the 22 basic benchmark functions and 28 CEC2013 benchmark functions. Four recently well-established ABC variants are compared with our approach, and the comparative results show that our approach can significantly perform better on the majority of the benchmark functions.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Nos. 61966019, 61603163, 61877031 and 61876074), the Science and Technology Foundation of Jiangxi Province (No. 20192BAB207030).

REFERENCES

[1] F. Wang, Y. Li, A. Zhou, and K. Tang, "An estimation of distribution algorithm for mixed-variable newsvendor problems," IEEE Transactions on Evolutionary Computation, 2019.

[2] F. Wang, Y. Li, H. Zhang, T. Hu, and X. Shen, "An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization," Swarm and Evolutionary Computation, vol. 49, pp. 220-233, 2019.

[3] S. T. K, F. M. K, K. S, and H. Q, "Genetic algorithms and their applications," IEEE Signal Processing Magazine, vol. 13, pp. 22-37, 1996.

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

[5] K. J and E. R, "Particle swarm optimization," in Proceedings of the International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4.

[6] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341-359, 1997.

[7] D. M. and D. C. G., "Ant colony optimization: a new meta-heuristic," in Proceedings of the IEEE Congress on Evolutionary Computation, 1999, pp. 1470-1477 vol. 2.

[8] D. Karaboga, "An idea based on honey bee swarm for numerical optimization,", Erciyes University, Kayseri, Trkiye, Tech Technical report-tr06, 2005.

[9] X. Zhou, H. Wang, M. Wang, and J. Wan, "Enhancing the modified artificial bee colony algorithm with neighborhood search," Soft Computing, vol. 21, pp. 2733-2743, 2017.

[10] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," Applied Mathematics & Computation, vol. 217, pp. 3166-3173, 2010.

[11] W. F. Gao and S. Y. Liu, "A modified artificial bee colony algorithm," Computers & Operations Research, vol. 39, pp. 687-697, 2012.

[12] W. F. Gao, S. Y. Liu and L. L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," IEEE Transactions on Cybernetics, vol. 43, pp. 1011-1024, 2012.

[13] L. Cui, G. Li, Q. Lin, Z. Du, W. Gao, J. Chen, and N. Lu, "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.

[14] X. Zhou, Z. Wu, H. Wang, and S. Rahnamayan, "Gaussian bare-bones artificial bee colony algorithm," Soft Computing, vol. 20, pp. 907-924, 2016.

[15] P. Hu, C. Deng, and Z. Wu, "Best neighbor-guided artificial bee colony algorithm for continuous optimization problems," Soft Computing, vol. 23, pp. 8723-8740, 2019.

[16] F. Kang, J. Li and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," Information Sciences, vol. 181, pp. 3508-3531.

[17] W. Xiang and M. An, "An efficient and robust artificial bee colony algorithm for numerical optimization," Computers & Operations Research, vol. 40, pp. 1256-1265, 2013.

[18] T. K. Sharma and M. Pant, "Enhancing the food locations in an artificial bee colony algorithm," Soft Computing, vol. 17, pp. 1939–1965, 2013.

[19] 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 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, 2013.