Information Sciences 543 (2021) 242–258

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Enhancing artificial bee colony algorithm with multi-elite guidance

Xinyu Zhou [a,*], Jiaxin Lu [a], Junhong Huang [a], Maosheng Zhong [a], Mingwen Wang [a]

[a] *School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China*

ARTICLE INFO

Article history:
Received 29 February 2020
Received in revised form 28 June 2020
Accepted 12 July 2020
Available online 25 July 2020

Keywords:
Artificial bee colony
Solution search equation
Multi-elite guidance
Neighborhood search

ABSTRACT

Artificial bee colony (ABC) algorithm is a relatively new paradigm of swarm intelligence based optimization technique, which has attracted a lot of attention for its simple structure and good performance. For some complex optimization problems, however, the performance of ABC is challenged due to its solution search equation that has strong explorative ability but poor exploitative ability. To solve this defect, in this work, we propose an improved ABC algorithm by using multi-elite guidance, which has the benefits of utilizing valuable information from elite individuals to guide search while without losing population diversity. First, we construct an elite group by selecting some elite individuals, and then introduce two improved solution search equations into the employed bee phase and onlooker bee phase based on the elite group, respectively. Last, we develop a modified neighborhood search operator by utilizing the elite group as well, which aims to achieve a better tradeoff between explorative and exploitative abilities. To verify our approach, 50 well-known test functions and one real-world optimization problem are used in the experiments, including 22 scalable basic test functions and 28 complex CEC2013 test functions. Seven different well-established ABC variants are involved in the comparison and the results show that our approach can achieve better or at least comparable performance on most of the test functions.

1. Introduction

In recent years, as an effective and efficient type of optimization tools, evolutionary algorithms (EAs) have been becoming popular for solving real-world optimization problems in the fields of science research and engineering applications. Because many real-world optimization problems are often characterized as non-convexity, discontinuity, or even non-differentiability, so the traditional optimization methods are hard to deal with these kinds of problems, such as the gradient-based methods. Being different from the traditional optimization methods, however, EAs have few requirements on the mathematical properties of optimization problems, and they can even be used as a black-box optimization tool when optimization problems have no explicit analytic expression [1]. In fact, as a type of optimization algorithms, EAs have many different paradigms, such as genetic algorithm (GA) [2], ant colony optimization (ACO) [3], differential evolution (DE) [4–6], particle swarm optimization (PSO) [7,8], firefly algorithm (FA) [9,10], and artificial bee colony (ABC) algorithm [11].

* Corresponding author.
E-mail addresses: xyzhou@jxnu.edu.cn (X. Zhou), 15070967547@163.com (J. Lu), hugh9527@jxnu.edu.cn (J. Huang), zhongmaosheng@sina.com (M. Zhong), mwwang@jxnu.edu.cn (M. Wang).

https://doi.org/10.1016/j.ins.2020.07.037
0020-0255/© 2020 Elsevier Inc. All rights reserved.

In this work, we concentrate on studying one relatively new paradigm of EAs – the artificial bee colony (ABC) algorithm. The basic ABC algorithm, first proposed by Karaboga et al. in 2005 [12], mimics the intelligent foraging behavior of honeybee swarms, and the process of maximizing the nectar amount corresponds to searching the global best solution. In comparison with other popular EAs, such as DE and PSO, ABC has shown better or at least comparable performance on many well-known test functions [13]. Therefore, ABC has attracted a lot of attention for its simple structure and good performance, and it has been successfully applied to solve many real-world optimization problems [14,15], such as robot path planning [16], portfolio selection [17], job-shop scheduling problem [18], and filter design [19].

However, being similar with other EAs, ABC also suffers from the problems of premature convergence or slow convergence speed in solving complex optimization problems [20–23]. The main reason behind this is that the solution search equation in the basic ABC does well in exploration but poorly in exploitation, which has been verified by many researchers in the field of ABC community [24–27]. Hence, how to improve the performance of ABC by enhancing the exploitation has become an active research topic. As is well known, the exploration and exploitation are antagonistic, and keeping a good balance between them is critical important for EAs. So, in fact, how to enhance the exploitation while without sacrificing the exploration is challenging for improving the performance of ABC.

In the last few years, many improved ABC variants have been proposed, and most of them focus on designing new solution search equations [28–31]. For instances, being inspired by the PSO, Zhu and Kwong [28] proposed a gbest-guided ABC (GABC) by incorporating the global best solution into the solution search equation, which aims to utilize the valuable information from the global best solution to enhance the exploitation. Akey and Karaboga [29] pointed out that the slow convergence speed of the basic ABC is caused by changing only one parameter of the parent solution in the solution search equation, so they modified the solution search equation by introducing two new parameters to control the frequency and magnitude of the perturbation, respectively. Based on the global best solution, Zhou et al. [30] designed a Gaussian barebones search equation to replace the basic solution search equation, which generates a new solution in the search space formed by the current solution and the global best solution.

It is not difficult to observe that these improved ABC variants of modifying the solution search equation mainly pay attention to utilizing the global best solution or some elite solutions to enhance the exploitation. Nevertheless, it should be aware that the global best solution or elite solutions can be a double-edged sword that should be used properly. On the positive side, the exploitation can be enhanced by using the global best solution or elite solutions; but on the negative side, the global best solution or elite solutions may result in premature convergence by weakening the exploration or population diversity. So, it is rather important to elaborately utilize the global best solution or elite solutions.

Being motivated by these observations, in this work, we propose an improved ABC variant based on multi-elite guidance, which is abbreviated as MGABC. In the MGABC, we construct a group of elite solutions to make two modifications. The first modification is to introduce two new solution search equations into the employed bee phase and onlooker bee phase, respectively, while the second modification is to develop a modified neighborhood search operator. Both of these two modifications are based on the elite group with the aim of utilizing the valuable information to enhance the exploitation but without sacrificing the exploration. To verify the performance of our approach, extensive experiments are carried out on 50 well-known test functions and one real-world optimization problem, including 22 scalable basic test functions and 28 complex CEC2013 test functions. The experimental results are compared with other seven well-established ABC variants, and the comparison results show that our approach can achieve better or at least comparable performance on most of the test functions.

The rest of this paper is organized as follows. In the Section 2, the basic ABC and some related work will be briefly reviewed. We will describe the proposed MGABC in detail in the Section 3, and the experimental results and corresponding analysis will be given in the Section 4. Finally, the conclusion will be drawn in the Section 5.

2. Related work

2.1. The basic ABC algorithm

As a representative swarm intelligence based optimization algorithm, ABC implements the purpose of optimization through modeling the collective foraging behavior of honeybee swarms. In the basic ABC, the honeybee swarm consists of three kinds of bees: employed bee, onlooker bee and scout bee, and each kind of bee has its own responsibility. As for the employed bees, they are responsible for searching new food sources in the entire search space, and they can be considered to play the role of exploration. After the employ bees finish their searching, they will share the nectar information with the onlooker bees, such as the nectar amount and position. Based on the received information, the onlooker bees will search new food sources within the neighborhood of some selected food sources. More nectar amount a food source owns, higher probability of being selected it has. To some extent, the onlooker bees play the role of exploitation. Note that the number of employed bees equals to the number of onlooker bees, and the number of employed bees is the same with the number of food sources. If a food source cannot be improved for more than consecutive *limit* times, it will be considered to be exhausted. Correspondingly, the employed bee associated with the exhausted food source will turn into a scout bee, and then start to search a new food source in the entire search space. Note that a food source corresponds to a candidate solution to the optimization problem.

Being similar with other EAs, ABC starts its optimization process with an initialization phase. After that, the remaining process can be divided into three phases according to the number of kinds of bees: employed bee phase, onlooker bee phase and scout bee phase. These three phases will be iteratively executed until the terminal condition is meet. Let $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D})$ be the ith food source (candidate solution), where D denotes the dimension size of the optimization problem. The above four phases of the ABC optimization process are introduced as follows.

(1) Initialization phase

In this phase, the population of SN food sources will be randomly generated by using the Eq. (1). Note that the number of food sources equals to the number of employed bees or onlooker bees.

$$x_{i,j} = x_j^{min} + rand(0,1) \cdot (x_j^{max} - x_j^{min}) \tag{1}$$

where $i \in \{1, 2, \cdots, SN\}$ and $j \in \{1, 2, \cdots, D\}$. x_j^{min} and x_j^{max} represents the lower bound and the upper bound of the jth dimension, respectively.

(2) Employed bee phase

After the initialization phase, employed bees will search new food sources in the entire search space by using the solution search equation listed in the Eq. (2).

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j}) \tag{2}$$

where $V_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,D})$ is the new food source. X_k is a randomly selected food source from the population, and it has to be different from X_i. $\phi_{i,j}$ is a uniformly random number within the range $[-1, 1]$, and $j \in \{1, 2, \cdots, D\}$ is a randomly selected dimension. Note that only one dimension of X_i is changed to generate V_i. If $v_{i,j}$ exceeds the bound, it will be reset by using the Eq. (1). If the new food source V_i is better than its parent X_i, then X_i will be replaced with V_i, and the associated counter for X_i will be reset to 0, otherwise the counter will be increased by 1. Note that each food source has its own counter $limit_i$ to record how many times it has not been improved consecutively.

(3) Onlooker bee phase

In this phase, the onlooker bees will continue to search new food sources. Being different from the search pattern of the employed bees, however, the onlooker bees only search within the neighborhood of some selected food sources. The probability of being selected for a food source depends on its fitness value, which can be calculated by the Eq. (3).

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

where p_i is the selection probability for the food source X_i, and fit_i is the corresponding fitness value which can be calculated by the Eq. (4) as follows.

$$fit_i = \begin{cases} \frac{1}{1+f(X_i)} & \text{if } f(X_i) \geqslant 0 \\ 1 + abs(f(X_i)) & \text{otherwise} \end{cases} \tag{4}$$

where $f(X_i)$ is the objective function value of X_i. Being similar with the employed bee phase, if the new food source is better than its parent, then the new food source will be retained and the associated counter will be reset to 0.

(4) Scout bee phase

In this phase, the associated counter for each food source will be checked. If the counter with the highest value is larger than the predefined parameter *limit*, then the corresponding food source will be considered to be exhausted and the associated employed bee will turn into a scout bee to regenerate a new food source through the Eq. (1). Note that the control parameter *limit* is the only parameter to be preset apart from other common parameters, such as population size.

2.2. Improved ABC variants

Although the basic ABC has shown attractive features, its performance is still challenged in solving complex optimization problems. The main problems include the slow convergence speed and premature convergence, which are caused by the solution search equation that has good exploration but poor exploitation. Accordingly, most of the improved ABC variants focus on modifying the solution search equation by utilizing the global best solution or some elite solutions. In this subsection, some related representative work are reviewed as follows.

Being inspired by PSO, Zhu and Kwong [28] proposed a gbest guided ABC variant (GABC) based on the global best solution, in which the solution search equation is modified by incorporating the valuable information from the global best solution. Similarly, based on the global best solution, Gao and Liu [32] designed an improved solution search equation ABC/best/1 in their modified ABC variant (MABC), and this modification derives from the DE/best/1 mutation strategy. Zhou et al. [30] proposed a Gaussian bare-bones ABC variant (GBABC) in which a Gaussian bare-bones search equation is designed to replace the basic solution search equation, and the new designed search equation generates a new solution in the search space formed by the current solution and the global best solution. Yu et al. [31] propose an adaptive ABC variant (AABC) which is characterized by a novel greedy position update strategy based on the top *t* solutions, and an adaptive control scheme is designed to adjust the size of top solutions for controlling the greediness degree.

In addition to utilizing the global best solution or some elite solutions, the ensemble of multiple solution search equations is another hot research topic. For instances, Wang et al. [33] proposed a multi-strategy ensemble ABC variant (MEABC), in which a pool of three distinct solution search strategies coexists throughout the whole search process and these strategies compete with each other to produce offsprings. Kiran et al. [34] proposed the integration of multiple solution update rules with ABC (ABCVSS). In ABCVSS, five search strategies are used to update the solutions, and the artificial agents will learn which update rule is more appropriate based on the characteristics of the problem to find better solutions. Recently, based on three different solution search equations, i.e., ABC/best/1, ABC/best/2, and GABC, Xue et al. [26] proposed a self-adaptive ABC based on the global best solution (SABC-GB). In SABC-GB, these three solution search equations are used adaptively according to their previous performance of generating promising solutions.

In fact, there are some other relevant ABC variants besides the ones of utilizing the global best solution or employing multiple solution search equations. For examples, Akay and Karaboga [29] pointed out that the problem of slow convergence speed is due to only one dimension of the parent solution is changed in the solution search equation, so they introduced a new control parameter, called modification rate (*MR*), to control the frequency of perturbation. Moreover, another new parameter called scaling factor (*SF*) is introduced into the solution search equation to control the magnitude of perturbation. Cui et al. [35] proposed a depth-first search framework in their novel ABC variant (DFSABC_elite) with the aim of allocating more computing resources to the food sources with better quality and easier to be improved for evolution. Recently, based on the MABC [32], Zhou et al. [36] incorporated a neighborhood search operator into MABC in their proposed ABC variant (MABC-NS) to perform fine search in the neighborhood of the global best solution for improving the performance of ABC.

3. Our approach

In this section, we will describe the proposed MGABC algorithm in detail. First, the motivation of MGABC will be given. Second, two modified solution search equations and a modified neighborhood search operator will be introduced. Last, the complete procedure of MGABC will be shown. It is necessarily to point out that although the proposed MGABC has been introduced in our preliminary work [37], this paper presents a version with significant enhancement in comparison with the preliminary work.

3.1. Motivation

In recent years, the ABC algorithm has attracted a lot of attention for its simplicity yet good performance. But, for some complex optimization problems, such as multimodal problems, the performance of ABC is still not satisfactory, including slow convergence speed and premature convergence. The main reason behind the unsatisfactory performance is that the solution search equation does well in exploration but poorly in exploitation. From the description of the optimization process of the basic ABC shown in the Section 2, we can conclude that the performance of ABC mainly depends on the solution search equation, since the solution search equation is almost the only single way to generate new offsprings.

To enhance the performance of ABC, many improved ABC variants have been proposed, and most of them focus on modifying the solution search equation with the global best solution or some elite solutions. Without question, the valuable information from the global best solution or some elite solutions is beneficial to enhance exploitation and accelerate convergence speed. However, it should be aware that if the information is not used properly, the algorithm would become greedy for too strong exploitation. Hence, these observations motivate us to think: (1) how to design a novel solution search equation to utilize the information without sacrificing exploration, and (2) how to design an additional search strategy, as a complementary operator for the solution search equation, to generate new offsprings.

In addition, Akay and Karaboga [29] pointed out that the problem of slow convergence speed is due to only one dimension of the parent solution is changed in the solution search equation, so they introduced a new control parameter, called modification rate (*MR*), to control the frequency of perturbation. In fact, in our opinion, the parameter *MR* controls how many dimensions can be inherited from the parent solution for the new solution, which plays a similar role with the crossover parameter in the genetic algorithm. Based on the work [29], we are inspired to research whether the global best solution or some elite solutions can be combined with the parameter *MR* for better performance.

3.2. Modified solution search equations based on multi-elite guidance

Based on the above motivation, we introduce two modified solution search equations based on multi-elite guidance into the employed bee phase and onlooker bee phase, respectively. In the basic ABC, both of the employed bee phase and onlooker bee phase share the same solution search equation to generate new offsprings. However, as seen from the viewpoint of internal mechanism of ABC, the employed bees and onlooker bees have different responsibilities. So, it would be better to design a different solution search equation for the employed bee phase and onlooker bee phase, respectively.

As for the employed bee phase, its solution search equation should keep relatively strong exploration, since the employed bees are responsible for exploring new solutions in the entire search space. Therefore, the modified solution search equation listed in the Eq. (5) is employed for the employed bee phase, which derives from the CABC algorithm proposed by Gao et al. [38].

$$v_{i,j} = x_{r_1,j} + \phi_{i,j} \cdot (x_{r_1,j} - x_{r_2,j}) \tag{5}$$

where X_{r_1} and X_{r_2} are two different food sources randomly selected from the population, and both of them are different from X_i. $\phi_{i,j}$ is a uniformly distributed random number within the range of $[-1,1]$. In the Eq. (5), the food sources for generating candidate solutions are all selected from the population randomly. Consequently, it has no bias to any search direction, and it is beneficial to keep good exploration.

Being different from the employed bee phase, the onlooker bee phase should concentrate on the exploitation, since the onlooker bees play the role of performing fine search within the neighborhood of good food sources to generate new offsprings. Hence, we design a modified solution search equation based on multiple elite solutions for the onlooker bee phase, which is listed in the following Eq. (6).

$$v_{i,j} = \begin{cases} x_{e,j} + \phi_{i,j} \cdot (x_{e,j} - x_{i,j}) & \text{if } rand(0,1) \leqslant MR \\ x_{i,j} & \text{otherwise} \end{cases} \tag{6}$$

where $X_e = (x_{e,1}, x_{e,2}, \cdots, x_{e,D})$ is a randomly selected elite solution from the elite group that comprises the top $q \cdot SN$ solutions in the current population. The control parameter MR has the same meaning with that in [29], which is to control how many dimensions can be inherited from the elite solution X_e to the new solution V_i. It should be noted that, in the onlooker bee phase of MGABC, the roulette wheel selection mechanism is used to select the good food sources to generate new offsprings.

The parameter $q \in (0,1)$ controls the size of the elite group, which has significant impact on the performance of the Eq. (6). If the value of q is close to 0, X_e will be the global best solution; but if its value trends to be 1, X_e will be a normal solution in the entire population rather than an elite solution. In our approach, we suggest the value of q is set to 0.1 which can maximize the performance, and the experiments of sensitivity test for q setting are conducted in the following Section 4. As for the parameter MR, we set it to 0.5 which is a good tradeoff value. Because, without prior knowledge about the problem, each 50% dimensions from the elite solution and the parent solution may contribute most to a good new solution. In the experiments, the sensitivity test for the MR setting will be given as well.

In conclusion, to replace the original solution search equation, we introduce two modified solution search equations. The first modified solution search equation listed in the Eq. (5) is for the employed bee phase, which is expected to keep good exploration; while the second one is designed for the onlooker bee phase with the aim of enhancing exploitation by utilizing multi-elite guidance. After combining these two modified solution search equations, it is expected to achieve a good balance between the exploration and exploitation, thus the performance of ABC can be maximized.

3.3. Modified neighborhood search operator based on multi-elite guidance

As described in the subSection 3.1 about motivation, it is necessary to design an additional search strategy, as a complementary operator for the solution search equation, to generate new offsprings. In the previous work [36,39], a neighborhood search operator is developed to enhance the performance of swarm intelligence based optimization algorithms, such as the PSO and ABC, and the extensive experiments have verified the effectiveness of the neighborhood search operator. The following Eq. (7) shows the details of this operator.

$$TX_i = r_1 \cdot X_i + r_2 \cdot gbest + r_3 \cdot (X_1 - X_2) \tag{7}$$

where TX_i is the trial solution generated by the neighborhood search operator for the parent solution X_i. gbest is the global best solution of the entire population, and X_1 and X_2 are two randomly selected solutions which have to be different from X_i. r_1, r_2, and r_3 are three non-negative real numbers randomly chosen from the range $(0,1)$, and they have to meet a constraint condition: $r_1 + r_2 + r_3 = 1$. If TX_i is better than X_i, then X_i will be replaced by TX_i, and this implies that the neighborhood search operator has a successful execution.

The core idea behind the neighborhood search operator is that, for complex multimodal problems, there exist many local optima in the fitness landscape and some of them may be near to the global optimum, so if searching the neighborhood of a good solution, it would be helpful to find better solutions or even the global optimum. To some extent, this operator can be used as an additional search strategy to generate new offsprings, since the operator performs particularly good exploitation and has a simple structure, which can be considered as a local search tool after the main procedure of ABC.

Based on the above considerations, the neighborhood search operator is employed in the proposed MGABC algorithm. But, instead of employing the same Eq. (7), we make a small modification by using the multiple elite solutions. The modified neighborhood search operator in our approach is listed in the Eq. (8).

$$TX_i = r_1 \cdot X_i + r_2 \cdot X_{e_1} + r_3 \cdot (X_{e_2} - X_{e_3}) \tag{8}$$

where X_{e_1}, X_{e_2} and X_{e_3} are three randomly selected food sources from the elite group, and they have to be different from X_i. Note that the elite group used in the Eq. (8) is the same as in the Eq. (6). As for other components in the Eq. (8), they have the same meaning with those in the Eq. (7). As seen, the differences among the Eqs. (7) and (8) include two parts: (1) the gbest is replaced by a randomly selected elite solution X_{e_1}, and (2) the two randomly selected solutions X_1 and X_2 are replaced by two randomly selected elite solutions X_{e_2} and X_{e_3}.

The reasons for the modification contain two aspects. First, the gbest component in the Eq. (7) may restrict the available search space within only the neighborhood of gbest, and this would cause inefficiency to the search ability of this operator. However, after using a randomly selected elite solution, the available search space would be extended dramatically. Second, although the two randomly selected solutions X_1 and X_2 can bring more information to the equation, there exists at least 50% chance that both of them are relatively bad solutions. In such condition, the search direction formed by X_1 and X_2 may lead to a poor search space. So, we use two randomly selected elite solutions to construct a more promising search direction. In a word, the modification in the Eq. (8) is expected to have a better balance between the exploration and exploitation. In addition, a control parameter p is introduced to control the probability of using the modified neighborhood search operator. The value of p is set to 0.1 in our approach, which is kept the same with our previous work [36]. The sensitivity test for the p setting is conducted in the experiments.

3.4. Complete procedure of our approach

To improve the performance of the basic ABC, we propose an improved ABC variant, i.e. MGABC, based on the multi-elite guidance. In MGABC, we make two modifications. The first modification is to introduce two modified solution search equations for the employed bee phase and onlooker bee phase, respectively; while the second modification is to develop a modified neighborhood search operator to generate new offsprings. To better clarify the complete procedure of our approach, the pseudo-code of MGABC is described in the Algorithm 1, where *FEs* is the used number of fitness function evaluations, *MaxFEs*, as the terminal condition, is the maximal number of fitness function evaluations.

Algorithm 1 Pseudo-code of MGABC

```
1: Randomly generate $SN$ food sources $\{X_i \mid i = 1, 2, \cdots, SN\}$ as the initial population;
2: $FEs = SN$;
3: while $FEs \leqslant MaxFEs$
4:    /* Employed bee phase */
5:    for $i = 1$ to $SN$
6:       Generate a new solution $V_i$ according to the Eq. (6);
7:       if $f(V_i) \leqslant f(X_i)$
8:          Replace $X_i$ with $V_i$;
9:          $trial_i$=0;
10:      else
11:         $trial_i = trial_i + 1$;
12:      end if
13:      $FEs = FEs + 1$;
14:   end for
15:   /* Onlooker bee phase */
16:   Calculate the probability $p_i$ according to the Eq. (3);
17:   for $i = 1$ to $SN$
18:      Choose a food source $X_j$ by the roulette wheel selection mechanism;
19:      Randomly choose an elite solution $X_e$ from the elite group;
20:      Generate a new candidate solution $V_j$ according to the Eq. (6);
21:      if $f(V_j) \leqslant f(X_j)$
22:         Replace $X_j$ with $V_j$;
23:         $trial_j$=0;
24:      else
25:         $trial_j = trial_j + 1$;
26:      end if
27:      $FEs = FEs + 1$;
28:   end for
29:   /* Scout bee phase */
30:   if max($trial_i$) $\geqslant limit$
31:      $trial_i$=0;
32:      Randomly generate a new solution for $X_i$ according to the Eq. (1);
33:      $FEs = FEs + 1$;
34:   end if
```

(continued on next page)

```
35:   /* Modified neighborhood search operator */
36:   for $i = 1$ to $SN$
37:     if $rand(0,1) \leqslant p$
38:       Generate a trial solution $TX_i$ by the Eq. (8);
39:       $FEs = FEs + 1$;
40:       if $f(TX_i) \leqslant f(X_i)$
41:         Replace $X_i$ with $TX_i$;
42:       end if
43:     end if
44:   end for
45: end while
```

4. Experimental verifications

In this section, we carry out extensive experiments to verify the performance of our approach. First, we will check the effectiveness of the two modifications in our approach. Second, the sensitivity tests for the control parameters: q (the size of the elite group), MR (the modification rate in the modified solution search equation), and p (the probability of triggering the modified neighborhood search operator), are conducted, since these control parameters has significant impact on the performance of our approach. Last, other seven well-established ABC variants are involved in the comparison with our approach.

4.1. Benchmark functions and parameter settings

In the following experiments, two groups of benchmark functions are used. The first group includes the well-known 22 scalable benchmark functions, while the second one contains the more complex 28 CEC2013 benchmark functions. For the first group of benchmark functions, the first 11 functions are unimodal type and the remaining ones are multimodal type. Specifically, however, the Rosenbrock function (F05) is multimodal when $D > 3$, F06 is a step function and has one minimum, and F07 is a noisy quartic function. The global optimum for the first group of benchmark functions is 0, a brief description about the first group of benchmark functions is given in the Table 1. For the second group of benchmark functions, the details and definitions can be found in [40].

As for the parameter settings, the maximal number of fitness function evaluations, i.e., *MaxFEs*, is different for the two groups of benchmark functions. For the first group of benchmark functions, *MaxFEs* is set to $5000 \cdot D$, while it is set to $10000 \cdot D$ for the second group of benchmark functions according to the suggestions in [40]. For the MGABC algorithm, the parameter q of controlling the size of the elite group is set to 0.1, the parameter MR of modification rate in the solution search Eq. (6) is set to 0.5, the parameter p of controlling the probability of using the modified neighborhood search operator is set to 0.1, the parameter *limit* in the scout bee phase is set to 100, and the number of food sources SN is set to 75.

Table 1
Brief description about the first group of benchmark functions.

Functions	Name	Search range
F01	Sphere	$[-100, 100]$
F02	Schwefel 2.22	$[-10, 10]$
F03	Schwefel 1.2	$[-100, 100]$
F04	Schwefel 2.21	$[-100, 100]$
F05	Rosenbrock	$[-30, 30]$
F06	Step	$[-100, 100]$
F07	Quartic with noise	$[-1.28, 1.28]$
F08	Elliptic	$[-100, 100]$
F09	SumSquare	$[-10, 10]$
F10	SumPower	$[-1, 1]$
F11	Exponential	$[-1.28, 1.28]$
F12	Schwefel 2.26	$[-500, 500]$
F13	Rastrigin	$[-5.12, 5.12]$
F14	Ackley	$[-32, 32]$
F15	Griewank	$[-600, 600]$
F16	Generalized Penalized 1	$[-50, 50]$
F17	Generalized Penalized 2	$[-50, 50]$
F18	NCRastrigin	$[-5.12, 5.12]$
F19	Alpine	$[-10, 10]$
F20	Levy	$[-10, 10]$
F21	Bohachevsky_2	$[-100, 100]$
F22	Weierstrass	$[-0.5, 0.5]$

4.2. Effectiveness test for the two modifications in MGABC

In the MGABC, we propose two modifications for enhancing the performance of ABC. The first modification is to introduce two modified solution search equations into the employed bee phase and onlooker bee phase, respectively. The second modification is to design a modified neighborhood search operator to generate new offsprings. To check the effectiveness of these two modifications, we construct two corresponding competitive ABC variants based on the basic ABC, i.e., ABC-elite and ABC-NS. ABC-elite is constructed by combing the first modification with the basic ABC, which means the original solution search equation in the basic ABC is replaced by the two modified solution search equations. For the ABC-NS, the modified neighborhood search operator is added to the basic ABC, and it implies that the modified neighborhood search operator will be triggered after the main procedure of the basic ABC.

In this subsection, the experiment is conducted on the first group of benchmark functions, and the dimension size of the test functions D is set to 30. Moreover, the basic ABC is also included in the comparison as a baseline. Hence, there are four algorithms in this experiment: ABC, ABC-elite, ABC-NS and MGABC. To make a fair comparison, all of these four algorithms share the same parameter settings, such as the number of food sources. Each algorithm is run 30 times per function, and the average best function value is recorded as the final result. The Table 2 gives all of the final results for the four algorithms, and the best results among these four algorithms are marked with **boldface**.

From the Table 2, it can be observed that both of the ABC-elite and ABC-NS can achieve better results than the basic ABC, which verify the effectiveness of the two modifications in MGABC. For the ABC-elite and ABC-NS, they have different performance on the test functions. To be specific, the ABC-elite performs better on the multimodal functions, while ABC-NS obtains better results on the unimodal functions. The possible reason is that the two modified solution search equations in the ABC-elite can provide better balance between the exploration and exploitation, thus the ABC-elite can achieve better results on the multimodal functions which are more difficult to solve than the unimodal functions. But after combing these two modifications, the MGABC can provide the best results among the involved four algorithms. This proves that these two modifications can work together efficiently to maximize the performance of their combination.

4.3. Sensitivity tests for the control parameters

4.3.1. Sensitivity test for the control parameter q

In this subsection, we perform a sensitivity test for the control parameter q which determines the size of the elite group and has important impact on the performance of our approach. From the Eqs. (6) and (8), it can be seen that the randomly selected elite solution will vary with the size of the elite group. If the control parameter q has a large value, the elite group will be close to the entire population, which implies that the randomly selected elite solution would tend to be a normal solution rather than a real elite solution. In this case, the modified solution search equation for the onlooker bee phase and the modified neighborhood search operator will lose their effectiveness for being without the multi-elite guidance. Conversely, if the control parameter q has a small value, the elite group will degrade into the global best solution, which will cause the problem that the Eqs. (6) and (8) would become too greedy.

Table 2
Comparison results of the ABC, ABC-elite, ABC-NS and MGABC algorithms.

Function	ABC	ABC-elite	ABC-NS	MGABC
F01	2.80E−10	4.74E−56	1.38E−97	**3.95E−183**
F02	9.97E−07	2.52E−34	3.89E−49	**5.55E−93**
F03	7.32E+03	2.77E+02	4.17E−93	**1.16E−95**
F04	3.13E+01	1.08E−05	4.23E−09	**3.44E−70**
F05	**7.68E−01**	4.71E+01	2.87E+01	2.40E+01
F06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F07	1.78E−01	8.46E−03	**3.16E−04**	5.55E−04
F08	1.04E−04	1.23E−50	2.71E−94	**7.71E−177**
F09	1.04E−11	1.18E−58	1.44E−98	**2.97E−185**
F10	3.20E−11	9.59E−120	1.30E−35	**2.57E−250**
F11	7.47E−07	7.85E−06	**6.66E−17**	4.22E−16
F12	**3.82E−04**	**3.82E−04**	**3.82E−04**	**3.82E−04**
F13	2.87E−09	**0.00E+00**	**0.00E+00**	**0.00E+00**
F14	3.51E−06	6.72E−15	**5.63E−16**	1.75E−15
F15	5.24E−09	8.22E−04	**0.00E+00**	**0.00E+00**
F16	6.38E−12	**1.57E−32**	4.41E−15	**1.57E−32**
F17	4.14E−10	**1.35E−32**	1.07E−10	**1.35E−32**
F18	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F19	7.91E−05	1.05E−16	9.83E−51	**3.56E−93**
F20	3.30E−08	**1.35E−31**	9.04E−10	**1.35E−31**
F21	5.10E−08	**0.00E+00**	**0.00E+00**	**0.00E+00**
F22	3.23E−04	**0.00E+00**	**0.00E+00**	**0.00E+00**

Based on the above considerations, it is necessary to set a proper value for the control parameter q. In this experiment, based on the first group of benchmark functions with $D = 30$, we test different values for the control parameter q to check its sensitivity. Due to the q value is within the range $(0, 1)$, we test nine different values taken from the range $[0.1, 0.9]$ in steps of 0.1, i.e., {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. The other parameter settings of MGABC are kept the same as in the subSection 3.1. The MGABC with each different value is run 30 times per function, and the average best function value is recorded as the final result. To simplify the results, only half of the values with better results are shown, and the Table 3 shows the final results and the best results are marked with **boldface**.

From the Table 3, it is clear that the q value has little effect on the performance of MGABC on the multimodal functions. But for the unimodal functions, the q value affects the performance, and there is a trend that the MGABC favors a small q value. The possible reason behind this phenomenon is that the modified neighborhood search operator can perform better on the unimodal functions, since a small q value means a small size of the elite group and thus the operator can focus on searching better solutions within a relatively good neighborhood. According to this sensitivity test, we suggest that the q value should be set to 0.1 for maximizing the overall performance of MGABC.

4.3.2. Sensitivity test for the control parameter MR

Based on the inspiration from the work [29], in the modified solution search equation listed in the Eq. (6), we introduce a new control parameter MR to control how many dimensions can be inherited from the elite solution X_e to the new solution V_i. Generally, if MR has a large value, this implies that the new solution V_i can inherit more information from the elite solution X_e, so it would be a high probability that V_i will have good quality, but the population diversity will be sacrificed at the same time; rather, more information from the parent solution X_i will be inherited to V_i, this would lead to that the modified solution search equation will has similar search behavior with the original solution search equation.

So, in this subsection, we carry out a sensitivity test for MR to investigate its affect on the performance of our approach. In this experiment, based on the first group of benchmark functions with $D = 30$, we test different values for the control parameter MR to check its sensitivity. Being similar with the test for the control parameter q, we test nine different values for MR, i.e., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 and 0.9. As for other parameter settings of MGABC, they are set to the same values with the subSection 3.1. The MGABC with a different MR value is run 30 times per function, and the average best function value is recorded as the final result. To simplify the results, only five representative MR values are picked out, and the Table 4 shows the final results and the best results are marked with **boldface**.

As seen in the Table 4, for the unimodal problems, there is a trend that the MGABC has better performance when MR has a larger value. The possible reason is that a unimodal problem has relatively simple structure of the fitness landscape which favors strong exploitative search ability, so it would be better to inherit more information from elite solutions. As for the multimodal problems, although the five different MR values have similar overall performance, there still exist differences on some specified test functions, such as the F14 and F20. From these two test functions, it can be seen that both of the values 0.1 and 0.9 not always achieve the best performance, and this implies that their performance seems to be unstable on the multimodal problems. After considering these, it would be better to set MR to 0.5, because the intermediate value has good and stable overall performance. Especially for solving an unknown problem without any prior knowledge, 0.5 is a good tradeoff value to generate a new solution.

Table 3
Final results of the MGABC with different q values.

Function	$q = 0.1$	$q = 0.2$	$q = 0.3$	$q = 0.4$	$q = 0.5$
F01	**3.95E−183**	5.74E−130	7.43E−109	7.84E−93	1.68E−82
F02	**5.55E−93**	2.04E−67	3.52E−56	7.89E−48	7.96E−43
F03	**1.16E−95**	5.19E−70	7.52E−57	1.41E−44	4.53E−33
F04	**3.44E−70**	5.46E−49	8.78E−40	2.12E−32	3.59E−28
F05	**2.40E+01**	2.47E+01	2.52E+01	2.57E+01	2.61E+01
F06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F07	**5.55E−04**	8.91E−04	1.14E−03	1.39E−03	1.67E−03
F08	**7.71E−177**	7.98E−128	1.61E−104	7.88E−89	1.43E−78
F09	**2.97E−185**	6.04E−133	7.18E−111	1.40E−94	1.22E−84
F10	**2.57E−250**	5.58E−209	1.34E−183	3.31E−164	3.62E−151
F11	**4.22E−16**	1.83E−13	4.33E−12	8.08E−11	3.35E−10
F12	**3.82E−04**	**3.82E−04**	**3.82E−04**	**3.82E−04**	**3.82E−04**
F13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F14	**1.75E−15**	3.05E−15	2.93E−15	3.64E−15	3.88E−15
F15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F16	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**
F17	**1.35E−32**	**1.35E−32**	**1.35E−32**	**1.35E−32**	**1.35E−32**
F18	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F19	**3.56E−93**	1.01E−66	6.32E−27	8.70E−23	1.73E−21
F20	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**
F21	**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**

Table 4
Final results of the MGABC with different *MR* values.

Function	$MR = 0.1$	$MR = 0.3$	$MR = 0.5$	$MR = 0.7$	$MR = 0.9$
F01	3.24E−122	1.65E−154	3.95E−183	1.17E−213	**7.84E−251**
F02	2.92E−62	5.06E−79	5.55E−93	1.26E−107	**9.60E−127**
F03	3.58E−89	9.21E−87	1.16E−95	1.25E−110	**3.71E−135**
F04	2.56E−45	1.33E−56	3.44E−70	2.87E−85	**1.81E−104**
F05	2.78E+01	2.58E+01	2.40E+01	2.23E+01	**2.07E+01**
F06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F07	**4.60E−04**	6.24E−04	5.55E−04	5.21E−04	5.00E−04
F08	8.16E−119	4.38E−153	7.71E−177	2.12E−208	**1.10E−244**
F09	1.04E−122	1.79E−156	2.97E−185	2.21E−212	**7.69E−252**
F10	1.59E−117	8.26E−191	2.57E−250	9.47E−305	**0.00E+00**
F11	1.92E−16	6.22E−16	4.22E−16	2.52E−16	**1.41E−16**
F12	**3.82E−04**	**3.82E−04**	**3.82E−04**	**3.82E−04**	**3.82E−04**
F13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F14	**7.99E−16**	2.22E−15	1.75E−15	2.81E−15	3.05E−15
F15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F16	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**
F17	3.57E−26	**1.35E−32**	**1.35E−32**	**1.35E−32**	1.37E−32
F18	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F19	2.08E−62	2.80E−79	3.56E−93	1.47E−108	**9.68E−127**
F20	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**	2.35E−31
F21	**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**

4.3.3. Sensitivity test for the control parameter p

To control the probability of using the modified neighborhood search operator, we introduce a new parameter p into the MGABC. Before setting the p value, the role of the modified neighborhood search operator should be considered at first. As mentioned in the subSection 3.1 about motivation, the modified neighborhood search operator is designed to generate new offsprings as a complementary operator for the solution search equation. So, it is clear that the solution search equation is the main way of generating new offsprings, while the modified neighborhood search operator is a complementary way. This implies that the probability of triggering the modified neighborhood search operator should not be high for saving computing resources. On the other hand, the modified neighborhood search operator can be treated as a local search tool to some extent, hence it is reasonable to set not a high value for the control parameter p to avoid premature convergence.

In this experiment, in order to check whether the control parameter p deserves a relatively low value, we perform a sensitivity test for it. Based on the first group of benchmark functions with $D = 30$, we test 10 different p values taken from the range [0.1, 1] in steps of 0.1, i.e., {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}. As for other parameter settings of MGABC, they have the same values like in the subSection 3.1. The MGABC with a different p value is run 30 times per function, and the average best function value is recorded as the final result. To simplify the results, the results of five representative p values are presented in the Table 5, and the best results are marked with **boldface**.

It is clear that the best results of the unimodal problems are achieved when the control parameter p is set to 1, which meets our expectation that the MGABC with a higher p value should perform better on the unimodal problems. Because the modified neighborhood search operator has a good local search ability, and this is beneficial to solve the unimodal problems that have relatively plain fitness landscape. However, the multimodal problems prefer a low value for the control parameter p, such as the test functions F12, F17 and F20. Therefore, after considering the role of the modified neighborhood search operator and the results of the sensitivity test, we suggest that the control parameter p should have a low value and thus it is set to 0.1.

4.4. Comparison with other improved ABC variants

In this subsection, we further verify the effectiveness of our approach by comparing it with other seven well-established ABC variants based on two groups of benchmark functions. The involved seven ABC variants are listed as follows.

- GBABC [30]: Gaussian bare-bones ABC
- AABC [31]: ABC with adaptive greedy position update strategy
- iqABC [21]: Improved quick ABC
- MEABC [33]: Multi-strategy ensemble ABC
- ABCVSS [34]: ABC with variable search strategy
- DFSABC-elite [35]: ABC with depth-first search framework and elite-guided search equation
- MABC-NS [36]: Modified ABC with neighborhood search

Table 5
Final results of the MGABC with different p values.

Function	$p = 0.1$	$p = 0.3$	$p = 0.5$	$p = 0.8$	$p = 1$
F01	3.95E−183	5.16E−317	**0.00E+00**	**0.00E+00**	**0.00E+00**
F02	5.55E−93	6.48E−159	1.84E−204	2.49E−252	**2.10E−279**
F03	1.16E−95	1.12E−241	**0.00E+00**	**0.00E+00**	**0.00E+00**
F04	3.44E−70	7.25E−138	1.49E−183	4.57E−234	**2.17E−259**
F05	2.40E+01	**2.38E+01**	2.41E+01	2.47E+01	2.53E+01
F06	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F07	5.55E−04	1.63E−04	7.85E−05	5.68E−05	**4.57E−05**
F08	7.71E−177	1.01E−312	**0.00E+00**	**0.00E+00**	**0.00E+00**
F09	2.97E−185	4.66E−317	**0.00E+00**	**0.00E+00**	**0.00E+00**
F10	2.57E−250	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F11	4.22E−16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F12	**3.82E−04**	**3.82E−04**	3.95E+00	9.80E+01	8.69E+01
F13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F14	1.75E−15	6.81E−16	**4.44E−16**	**4.44E−16**	**4.44E−16**
F15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F16	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**
F17	**1.35E−32**	1.37E−32	2.55E−29	4.77E−11	4.03E−03
F18	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F19	3.56E−93	9.65E−160	5.30E−205	2.57E−252	**2.02E−279**
F20	**1.35E−31**	2.18E−31	1.83E−02	3.66E−03	2.95E−02
F21	**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**

A short introduction about the above seven ABC variant are as follows. In GBABC, Zhou et al. [30] designed a Gaussian bare-bones search equation in which the global best solution is utilized, and the new solution can be generated from the search space formed by the current solution and the global best solution. In AABC, Yu et al. [31] proposed a greedy position update strategy based on the top t solutions, and an adaptive control scheme is designed to adjust the size of top solutions for controlling the greediness degree. In iqABC, Aslan et al. [21] proposed a new exploitation mechanism from the best food source that is managed by the number of evaluations. In MEABC, Wang et al. [33] introduced three distinct solution search strategies which coexist throughout the whole search process and compete with each other to produce offsprings. In ABCVSS, Kiran et al. [34] proposed the integration of multiple solution update rules and used five search strategies to update the solutions. In DFSABC-elite, Cui et al. [35] introduced a depth-first search framework into ABC and proposed two novel solution search equations based on the elite solutions and current best solution, respectively. In MABC-NS, Zhou et al. [36] added the neighborhood search into ABC as an additional search strategy.

The reasons of choosing these seven improved ABC variants are as follows. First, the one modification in the MGABC is about two modified solution search equations, so it is necessary to compare MGABC with other ABC variants which focus on modifying the solution search equation based on the global best solution or elite solutions, thus the GBABC, AABC, and iqABC are chosen. Second, as there are many variants about the ensemble of multiple solution search equations, thereby the MEABC and ABCVSS are included. Last, the other modification in MGABC is the introduction of neighborhood search, so the MABC-NS is selected. Moreover, the neighborhood search operator in the MGABC can be considered as an additional search strategy, as a result the DFSABC-elite is included as well.

In this experiment, the comparisons include two parts: (1) the comparison on the first group of 22 scalable benchmark functions, and (2) the comparison on the second group of 28 CEC2013 benchmark functions. To make a fair comparison, all of the involved algorithms have the same setting for the number of food sources, i.e., $SN = 75$, and the dimension sizes of functions for test include $D = 30$ and $D = 50$. As for other specific parameters of the competitive algorithms, they are kept the same as in the corresponding papers. For a clear comparison, the basic ABC is included as a baseline. Moreover, the Wilcoxon's rank sum test, a nonparametric statistic test for independent samples, is conducted on the final results at $\alpha = 0.05$ significance level. The symbols "+", "=" and "−" denote that the performance of the competitive algorithm is better than, similar to, and worse than that of the MGABC, respectively, according to Wilcoxon's rank test at $\alpha = 0.05$ significance level.

4.4.1. Comparison on the first group of benchmark functions

For the first group of 22 scalable benchmark functions, the Tables 6 and 7 give the final results of the involved nine algorithms for $D = 30$ and $D = 50$, respectively. As for $D = 30$, first, it can be seen from the Table 6 that the MGABC performs best among the nine involved algorithms, and it achieves the best results on 18 out of 22 test functions. Second, as for the unimodal problems, the MGABC performs best on 8 out of 11 test functions, and it is impressive that the final results of MGABC are much better than the other algorithms on seven functions, such as F01, F08, F09, and F10. For the two test functions, i.e., F05 and F07, in comparison with the best results, the results of the MGABC are not too much worse. Last, in regard to the multimodal problems, the MGABC can perform best on 10 out 11 functions, and it only loses on the F14. For the test function F14, however, the MGABC actually gets the second place, but its result is comparable to the best one.

Table 6
Comparison with other improved ABC variants on the first group of benchmark functions at $D = 30$.

Function	ABC	GBABC	AABC	iqABC	MEABC	ABCVSS	DFSABC_elite	MABC-NS	MGABC
F01	2.80E−10−	2.05E−26−	2.72E−24−	3.39E−09−	6.46E−26−	9.54E−23−	1.57E−57−	1.63E−81−	**3.95E−183**
F02	9.97E−07−	3.48E−16−	3.33E−13−	3.16E−06−	3.08E−14−	6.67E−13−	7.95E−30−	6.25E−42−	**5.55E−93**
F03	7.32E+03−	2.84E+03−	9.95E+03−	5.38E+03−	9.34E+03−	9.90E+03−	4.46E+03−	3.72E−59−	**1.16E−95**
F04	3.13E+01−	6.70E−01−	2.18E+01−	8.45E−01−	9.56E+00−	7.68E+00−	9.53E−05−	1.03E−30−	**3.44E−70**
F05	**7.68E−01+**	2.34E+01=	9.91E−01+	1.38E+00+	1.55E+00+	1.01E+00+	3.61E+00+	2.78E+01−	2.40E+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**
F07	1.78E−01−	2.32E−02−	9.31E−02−	6.01E−02−	3.79E−02−	3.50E−02−	1.30E−02−	**1.41E−04+**	5.55E−04
F08	1.04E−04−	1.64E−18−	1.17E−20−	4.01E−06−	5.74E−23−	2.57E−15−	8.63E−54−	3.47E−78−	**7.71E−177**
F09	1.04E−11−	3.90E−25−	2.76E−25−	3.49E−10−	6.40E−27−	3.81E−24−	1.38E−58−	2.29E−82−	**2.97E−185**
F10	3.20E−11−	3.76E−50−	4.77E−20−	8.51E−12−	2.07E−32−	9.10E−20−	2.85E−41−	8.74E−72−	**2.57E−250**
F11	7.47E−07−	1.52E−06−	1.55E−06−	5.10E−07−	1.24E−06−	1.18E−06−	3.27E−06−	**0.00E+00+**	4.22E−16
F12	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04=**	**3.82E−04**
F13	2.87E−09−	2.50E−03−	**0.00E+00=**	2.22E−08−	**0.00E+00=**	**0.00E+00=**	3.32E−02=	**0.00E+00=**	**0.00E+00**
F14	3.51E−06−	1.03E−10−	1.32E−12−	1.19E−0−5	4.08E−13−	1.79E−12−	7.55E−15−	**4.44E−16+**	1.75E−15
F15	5.24E−09−	8.81E−07−	1.29E−12−	5.27E−08−	**0.00E+00=**	6.38E−13−	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
F16	6.38E−12−	9.03E−26−	2.47E−26−	6.86E−11−	4.34E−28−	2.92E−25−	**1.57E−32=**	4.95E−31−	**1.57E−32**
F17	4.14E−10−	3.33E−24−	5.98E−25−	2.76E−09−	1.38E−26−	2.24E−23−	**1.35E−32=**	1.63E−29−	**1.35E−32**
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**
F19	7.91E−05−	1.43E−07−	2.86E−10−	1.13E−04−	6.28E−14−	8.03E−10−	7.40E−18−	6.85E−43−	**3.56E−93**
F20	3.30E−08−	7.04E−23−	8.14E−17−	3.52E−07−	1.30E−25−	1.70E−16−	**1.35E−31=**	5.44E−28−	**1.35E−31**
F21	5.10E−08−	**0.00E+00=**	**0.00E+00=**	3.91E−07−	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
F22	3.23E−04−	2.15E−13−	3.20E−13−	9.20E−04−	**0.00E+00=**	4.52E−14−	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
+/=/−	1/3/18	0/5/17	1/5/16	1/3/18	1/7/14	1/5/16	1/10/11	3/7/12	–

Table 7
Comparison with other improved ABC variants on the first group of benchmark functions at $D = 50$.

Function	ABC	GBABC	AABC	iqABC	MEABC	ABCVSS	DFSABC_elite	MABC−NS	MGABC
F01	5.58E−10−	9.96E−19−	1.60E−23−	8.02E−09−	2.31E−24−	5.34E−22−	4.01E−57−	4.82E−118−	**3.96E−288**
F02	2.53E−06−	7.68E−15−	9.04E−13−	8.17E−06−	2.37E−13−	1.98E−12−	1.56E−29−	4.20E−60−	**4.43E−145**
F03	2.52E+04−	2.11E+04−	3.21E+04−	1.72E+04−	3.04E+04−	2.93E+04−	1.62E+04−	1.25E−99−	**7.00E−146**
F04	5.86E+01−	6.93E+00−	5.19E+01−	2.91E+00−	2.40E+01−	2.10E+01−	4.06E−03−	1.53E−50−	**1.60E−111**
F05	**8.43E−01+**	7.66E+01−	1.31E+00+	2.52E+00+	1.23E+00++	1.08E+00+	3.02E+00+	4.78E+01−	4.22E+01
F06	3.33E−02=	**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	4.81E−01−	5.94E−02−	2.69E−01−	9.25E−02−	8.99E−02−	8.39E−02−	2.49E−02−	**1.14E−04+**	6.04E−04
F08	1.86E−04−	5.89E−43−	3.79E−20−	6.89E−06−	4.00E−21−	2.51E−14−	4.34E−53−	8.86E−115−	**1.16E−282**
F09	8.65E−11−	5.84E−19−	1.55E−24−	1.91E−09−	5.16E−25−	4.26E−23−	1.17E−57−	1.37E−118−	**4.08E−288**
F10	6.94E−08−	1.14E−51−	4.99E−15−	5.46E−12−	4.18E−32−	6.47E−20−	3.53E−41−	2.74E−99−	**0.00E+00**
F11	4.46E−07−	7.31E−07−	7.85E−07−	2.02E−07−	4.18E−07−	6.17E−07−	1.75E−06−	**0.00E+00+**	1.48E−16
F12	**6.36E−04=**	**6.36E−04=**	**6.36E−04=**	6.37E−04=	**6.36E−04=**	**6.36E−04=**	**6.36E−04=**	**6.36E−04=**	**6.36E−04**
F13	1.43E−01−	1.10E+01−	**0.00E+00=**	1.40E−08−	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
F14	7.02E−06−	5.45E−09−	2.86E−12−	2.23E−05−	1.65E−12−	3.52E−12−	1.29E−14−	**4.44E−16+**	2.10E−15
F15	6.87E−10−	1.51E−12−	3.71E−14−	2.93E−08−	**0.00E+00=**	4.19E−14−	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
F16	1.37E−11−	9.75E−23−	3.48E−26−	1.17E−10−	1.20E−26−	9.65E−25−	**9.42E−33=**	2.73E−29−	**9.42E−33**
F17	2.03E−09−	9.17E−23−	1.54E−24−	5.71E−09−	4.78E−25−	1.36E−22−	**1.35E−32=**	6.19E−28−	**1.35E−32**
F18	2.01E−13=	**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	2.97E−04−	1.79E−05−	9.62E−10−	3.55E−04−	5.16E−13−	3.81E−09−	1.71E−27−	3.90E−61−	**4.38E−146**
F20	3.15E−08−	4.38E−17−	2.37E−16−	5.65E−07−	2.28E−24−	5.98E−17−	**1.35E−31=**	5.77E−26−	**1.35E−31**
F21	1.17E−07−	2.09E−16=	**0.00E+00=**	3.34E−07−	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
F22	7.84E−04−	3.80E−10−	2.27E−13−	1.78E−03−	3.13E−14−	3.45E−13−	**0.00E+00=**	**0.00E+00=**	**0.00E+00**
+/=/−	1/3/18	0/4/18	1/5/16	1/3/18	1/6/15	1/5/16	1/10/11	3/7/12	–

The Table 7 shows the comparison results for $D = 50$, and this case is similar with $D = 30$. Although the overall performance of each algorithm deteriorates with the growth of the dimension size of function, the MGABC consistently gets better results than its other eight competitors. This implies that MGABC has good scalability or robustness. To evaluate the overall performance of the involved nine algorithms, we conduct the Friedman test to obtain the average rankings. The Table 8 gives the average rankings for both $D = 30$ and $D = 50$. As seen, the MGABC achieves the best average ranking among the involved nine algorithms, and the best average ranking is marked with **boldface**.

In addition to the comparison of final result accuracy, we further compare the convergence speed of the involved nine algorithms on the first group of benchmark functions. In the Fig. 1, the corresponding convergence curves on six representative test functions are depicted for $D = 30$, including three unimodal functions (F02, F04 and F07) and three multimodal functions (F12, F17 and F20). As seen, for the test functions F02 and F04, the convergence speed of the MGABC is clearly faster than its other eight competitors. As for the test function F12, although the iqABC is faster than the MGABC before 50,000 FEs, the MGABC also shows comparable fast convergence speed after 50,000 FEs. In regard to the test functions F17 and F20,

Table 8
Average rankings of the involved nine algorithms tested on the first group of benchmark functions for both $D = 30$ and $D = 50$.

Algorithms	$D = 30$	$D = 50$
ABC	7.05	7.57
GBABC	5.5	5.93
AABC	5.64	5.61
iqABC	7.27	7.41
MEABC	4.59	4.45
ABCVSS	5.77	5.52
DFSABC_elite	3.84	3.43
MABC-NS	2.91	2.77
MGABC	**2.43**	**2.3**

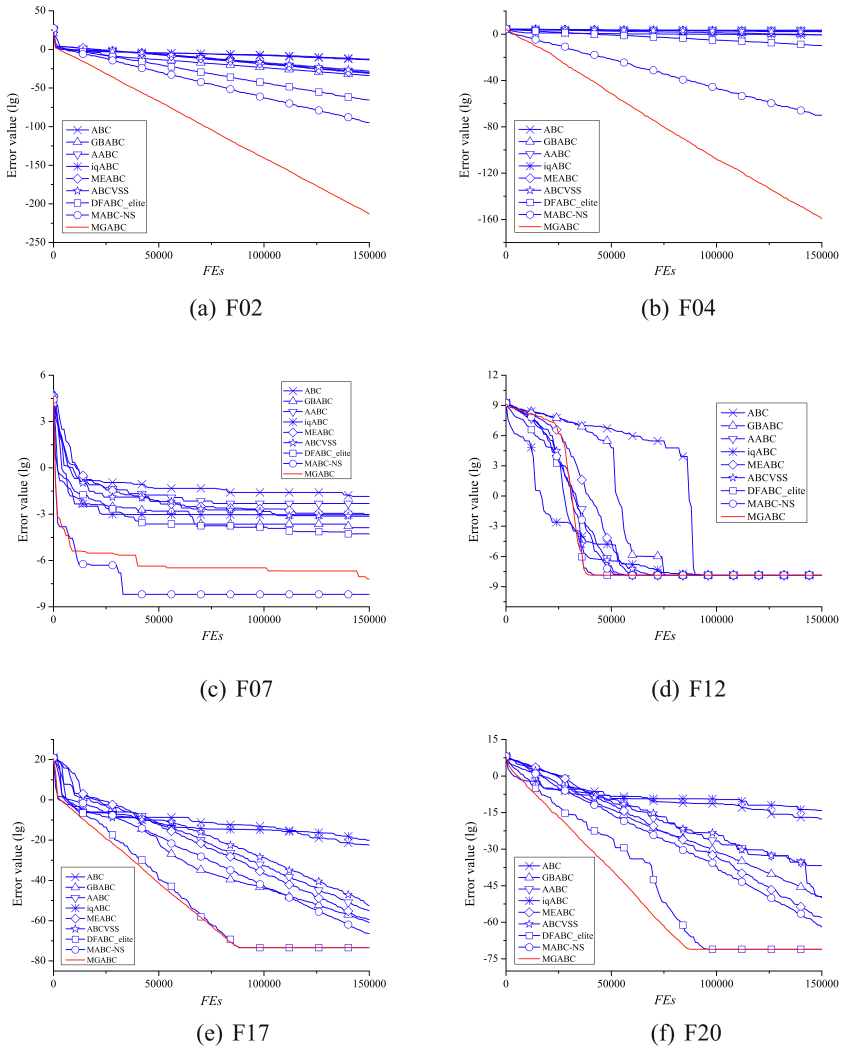

Fig. 1. Convergence curves of the involved nine algorithms on six representative functions of the first group of benchmark functions for $D = 30$.

both of the DFABC_elite and MGABC achieve the same result accuracy, but the MGABC performs faster convergence speed. In a word, according to the presented convergence curves, the MGABC can exhibit promising convergence speed.

4.4.2. Comparison on the second group of benchmark functions

To further investigate the effectiveness of the MGABC, we compare it with the above seven improved ABC variants on the second group of 28 CEC2013 benchmark functions. It is well known that the CEC2013 test suite is much more difficult to solve than the basic test functions, so the comparison results will be more convincing. The classic ABC is included in this experiment as well. The parameter settings for the involved nine algorithms are kept the same as in the above subSection 4.4.1. But according to the descriptions of the subSection 4.1, the MaxFEs is set to $10000 \cdot D$ in this experiment. The dimension size of the test functions is set to $D = 30$ and $D = 50$, respectively.

The Table 9 presents the final results for $D = 30$. As seen, although it is more difficult to solve the CEC2013 test suite, the MGABC can still achieve the best performance on most of the test functions. To be specific, the MGABC can obtain better results on 18 out of 28 test functions in comparison with the classic ABC, AABC, and MABC-NS. Compared with the iqABC and ABCVSS, the MGABC can perform better on 17 test functions. As for the MEABC, the MGABC can defeat it on 16 test functions. Regarding the rest of competitive algorithms, i.e., the GBABC and DFSABC_elite, the MGABC can achieve better results on 15 test functions. So, we can conclude that, for $D = 30$, the MGABC can get the best overall performance for achieving the best results on the majority of test functions.

In the Table 10, the final results of the involved nine algorithms for $D = 50$ are presented. As seen, the comparison results between the MGABC and its competitors are similar with the case of $D = 30$. More specially, in comparison with the GBABC and MABC-NS, the MGABC can perform better on 20 and 19 test functions, respectively. As for the iqABC and ABCVSS, the MGABC achieves better results on 17 test functions. For the remaining competitive algorithms, the MGABC defeats them on 16 test functions. So, for $D = 50$, although the dimension size of the test functions increases, the MGABC still shows convincing performance. In addition, we conclude the Friedman test to obtain the average rankings for this experiment. The Table 11 gives the average rankings for both cases $D = 30$ and $D = 50$, and the best average ranking is marked with **boldface**. However, as seen, although the MGABC can show the best performance on the majority of test functions, it is ranked on the second place and the best ranking is obtained by the DFABC_elite. The main reason is that the MGABC does not show comparable performance on the four test functions, i.e., F16, F19, F21, and F22, even the final results on the four test functions are the worst. So, the average ranking of the MGABC is deteriorated by the rankings on these four test functions. But the difference of the average rankings between the MGABC and DFABC_elite is very small which can be accepted. Moreover, in the next subSection 4.5, we apply the algorithms to solve a real-world optimization problem and the comparison results confirm the superiority of the MGABC.

4.5. Application to a real-world optimization problem

In this subsection, we apply the MGABC to solve a real-world optimization problem, i.e., the Spread Spectrum Radar Polyphase Code Design (SSRPCD). The key point of solving the SSRPCD problem is to set an appropriate waveform for pulse modulation. The problem can be modeled as a min–max nonlinear non-convex optimization problem in continuous variables,

Table 9
Comparison with other improved ABC variants on the second group of benchmark functions at $D = 30$.

Function	ABC	GBABC	AABC	iqABC	MEABC	ABCVSS	DFSABC_elite	MABC−NS	MGABC
F01	6.21E−13−	2.50E−13−	2.88E−13−	4.17E−13−	4.55E−13−	4.93E−13−	2.05E−13−	5.38E−13−	**0.00E+00**
F02	1.17E+07−	2.36E+07−	1.88E+07−	8.64E+06−	1.21E+07−	1.12E+07−	7.61E+06−	3.57E+06−	**1.43E+06**
F03	6.85E+08−	**1.47E+08+**	1.64E+09−	3.70E+08−	7.62E+08−	2.67E+08−	2.55E+08−	2.31E+08−	2.11E+08
F04	6.86E+04−	4.86E+04−	7.09E+04−	8.62E+04−	9.06E+04−	8.99E+04−	8.66E+04−	2.68E+04−	**2.12E+04**
F05	1.01E−12−	2.27E−13−	6.06E−13−	1.16E−12−	4.96E−13−	5.08E−13−	1.29E−13−	4.85E−13−	**1.02E−13**
F06	**1.35E+01+**	1.93E+01+	1.80E+01+	1.58E+01+	1.56E+01+	1.78E+01+	1.79E+01+	4.83E+01−	2.80E+01
F07	1.01E+02−	**3.78E+01+**	1.17E+02−	1.06E+02−	1.04E+02−	1.05E+02−	9.41E+01−	6.63E+01−	4.30E+01
F08	**2.09E+01=**	**2.09E+01=**	2.10E+01−	2.10E+01−	**2.09E+01=**	**2.09E+01=**	**2.09E+01=**	**2.09E+01=**	**2.09E+01**
F09	3.01E+01−	2.96E+01−	3.04E+01−	3.00E+01−	2.95E+01−	2.93E+01−	2.83E+01−	2.57E+01−	**1.88E+01**
F10	2.66E+00−	3.28E−01−	3.90E+00−	1.20E+00−	3.40E+00−	2.93E+00−	4.10E−01−	3.65E−01−	**1.99E−01**
F11	1.74E−13−	6.44E−14−	5.68E−14−	1.16E−13−	1.16E−13−	1.14E−13−	5.49E−14−	3.32E−02−	**1.71E−14**
F12	2.40E+02−	1.86E+02−	2.09E+02−	2.40E+02−	1.66E+02−	1.62E+02−	**1.26E+02+**	1.75E+02−	1.31E+02
F13	2.97E+02−	1.92E+02−	2.49E+02−	3.07E+02−	2.27E+02−	2.23E+02−	1.84E+02−	2.41E+02−	**1.67E+02**
F14	4.20E+00−	1.86E+02−	**8.18E−03+**	7.42E−02+	2.35E−01+	3.40E−02+	1.24E+00−	5.08E−01−	3.03E−01
F15	3.96E+03=	4.93E+03+	4.97E+03−	3.71E+03+	3.70E+03+	3.85E+03=	**3.62E+03+**	3.94E+03−	3.86E+03
F16	1.65E+00+	1.87E+00+	1.98E+00+	1.04E+00+	1.18E+00+	1.25E+00+	**1.03E+00+**	1.40E+00+	2.37E+00
F17	3.07E+01−	3.23E+01−	3.04E+01=	**3.00E+01+**	3.04E+01=	3.04E+01=	3.04E+01=	3.04E+01=	3.04E+01
F18	3.00E+01=	3.00E+01=	3.00E+01=	3.00E+01=	3.00E+01=	**2.99E+01+**	3.00E+01=	3.00E+01=	3.00E+01
F19	9.41E−01+	1.07E+00+	5.67E−01+	5.02E−01+	3.68E−01+	3.97E−01+	**3.34E−01+**	1.16E+00+	1.78E+00
F20	1.19E+01−	1.18E+01−	1.23E+01−	1.20E+01−	1.18E+01−	1.18E+01−	1.15E+01−	**1.05E+01=**	1.07E+01
F21	2.11E+02+	2.80E+02+	2.59E+02+	**1.78E+02+**	2.02E+02+	2.02E+02+	3.07E+02+	3.10E+02+	3.33E+02
F22	9.32E+01+	1.61E+02−	9.69E+01+	3.58E+01+	7.53E+01+	**2.23E+01+**	8.46E+01+	9.36E+01+	1.11E+02
F23	5.11E+03−	5.55E+03−	5.78E+03−	4.76E+03−	4.82E+03−	4.59E+03−	4.52E+03−	4.53E+03−	**4.19E+03**
F24	2.84E+02−	2.34E+02−	2.81E+02−	2.88E+02−	2.85E+02−	2.83E+02−	2.79E+02−	2.41E+02−	**2.18E+02**
F25	3.10E+02−	2.72E+02=	3.05E+02−	3.13E+02−	3.03E+02−	3.03E+02−	3.01E+02−	2.78E+02−	**2.69E+02**
F26	2.01E+02−	**2.00E+02=**	2.01E+02−	2.01E+02−	2.01E+02−	2.01E+02−	**2.00E+02=**	**2.00E+02=**	**2.00E+02**
F27	**4.00E+02+**	4.01E+02+	4.04E+02+	4.97E+02+	4.72E+02+	5.17E+02+	5.12E+02+	8.09E+02−	6.40E+02
F28	2.22E+02+	3.00E+02=	2.84E+02+	**1.89E+02+**	2.66E+02+	2.50E+02+	3.00E+02=	2.87E+02=	2.93E+02
+/=/−	7/3/18	8/5/15	8/2/18	10/1/17	9/3/16	8/3/17	8/5/15	4/6/18	−−

Table 10
Comparison with other improved ABC variants on the second group of benchmark functions at $D = 50$.

Function	ABC	GBABC	AABC	iqABC	MEABC	ABCVSS	DFSABC_elite	MABC−NS	MGABC
F01	1.00E−12−	4.77E−13−	8.19E−13−	8.64E−13−	9.78E−13−	9.78E−13−	2.27E−13−	1.11E−12−	**2.20E−13**
F02	2.31E+07−	5.62E+07−	3.97E+07−	1.35E+07−	2.16E+07−	1.76E+07−	1.22E+07−	3.86E+06−	**1.66E+06**
F03	2.37E+09−	6.23E+09−	1.10E+10−	1.59E+09−	5.07E+09−	1.59E+09−	8.40E+08−	4.26E+08−	**1.54E+08**
F04	1.64E+05−	1.05E+05−	1.33E+05−	1.56E+05−	1.53E+05−	1.56E+05−	1.49E+05−	**5.54E+04+**	5.72E+04
F05	1.06E−12−	4.55E−13−	1.47E−12−	2.34E−12−	1.02E−12−	1.06E−12−	2.27E−13−	1.02E−12−	**1.14E−13**
F06	4.40E+01+	4.51E+01+	4.46E+01+	4.37E+01+	4.40E+01+	**4.31E+01+**	4.53E+01+	7.60E+01−	5.84E+01
F07	1.44E+02−	7.04E+01−	1.61E+02−	1.51E+02−	1.46E+02−	1.46E+02−	1.28E+02−	8.82E+01−	**5.95E+01**
F08	**2.11E+01=**	**2.11E+01=**	**2.11E+01=**	**2.11E+01=**	**2.11E+01=**	2.12E+01−	**2.11E+01=**	**2.11E+01=**	**2.11E+01**
F09	5.76E+01−	5.92E+01−	5.98E+01−	5.90E+01−	5.84E+01−	5.86E+01−	5.61E+01−	4.41E+01−	**4.09E+01**
F10	5.64E+00−	3.99E+00−	9.94E+00−	1.19E+00−	9.24E+00−	2.44E+00−	8.10E−01−	9.70E−01−	**1.87E−01**
F11	2.27E−13−	3.14E−09−	1.36E−13−	3.18E−13−	2.39E−13−	2.33E−13−	5.68E−14−	3.98E−01−	**4.74E−14**
F12	5.13E+02−	3.36E+02−	5.87E+02−	6.61E+02−	4.87E+02−	4.27E+02−	3.57E+02−	4.30E+02−	**1.81E+02**
F13	6.29E+02−	3.38E+02−	6.46E+02−	7.45E+02−	5.66E+02−	5.59E+02−	4.55E+02−	5.66E+02−	**2.84E+02**
F14	2.91E−02+	3.25E+02−	**1.75E−02+**	5.69E−02+	3.35E−01+	5.37E−02+	1.91E+00+	3.99E−01+	2.25E+00
F15	7.80E+03+	9.78E+03−	1.01E+04−	7.60E+03+	7.84E+03+	7.83E+03+	**7.40E+03+**	7.96E+03−	7.91E+03
F16	1.65E+00+	2.65E+00+	2.85E+00+	**1.39E+00+**	1.57E+00+	1.59E+00+	1.51E+00+	1.92E+00+	3.13E+00
F17	**5.08E+01=**	5.48E+01−	**5.08E+01=**	5.14E+01−	**5.08E+01=**	**5.08E+01=**	**5.08E+01=**	**5.08E+01=**	**5.08E+01**
F18	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01=**	**5.02E+01**
F19	6.27E−01+	1.98E+00+	1.25E+00+	1.19E+00+	7.48E−01+	8.99E−01+	**5.29E−01+**	2.62E+00−	2.58E+00
F20	2.11E+01−	2.08E+01−	2.15E+01−	2.12E+01−	2.09E+01−	2.10E+01−	2.05E+01−	**1.91E+01+**	1.98E+01
F21	2.63E+02+	3.10E+02+	2.98E+02+	**2.09E+02+**	2.77E+02+	2.79E+02+	2.90E+02+	3.50E+02−	3.47E+02
F22	1.49E+01+	8.22E+01−	1.09E+01+	3.06E+01−	1.32E+01+	**1.05E+01+**	2.48E+01+	4.04E+01−	2.92E+01
F23	1.00E+04−	1.14E+04−	1.18E+04−	9.86E+03−	1.01E+04−	9.58E+03−	9.08E+03−	9.20E+03−	**8.12E+03**
F24	3.70E+02−	2.93E+02−	3.62E+02−	3.74E+02−	3.66E+02−	3.63E+02−	3.62E+02−	3.05E+02−	**2.58E+02**
F25	4.12E+02−	3.50E+02−	4.12E+02−	4.33E+02−	4.13E+02−	4.07E+02−	3.97E+02−	3.89E+02−	**3.38E+02**
F26	2.02E+02+	2.02E+02+	2.04E+02+	2.02E+02+	2.02E+02+	2.03E+02+	2.01E+02+	**2.00E+02+**	2.29E+02
F27	1.44E+03−	**4.00E+02+**	7.16E+02+	1.04E+03+	1.47E+03−	1.30E+03−	1.26E+03−	1.50E+03−	1.16E+03
F28	**4.00E+02+**	**4.00E+02+**	**4.00E+02+**	**4.00E+02+**	**4.00E+02+**	**4.00E+02+**	**4.00E+02+**	1.18E+03−	7.10E+02
+/=/ −	9/3/16	7/2/19	9/3/16	9/2/17	9/3/16	9/2/17	9/3/16	5/3/20	− −

Table 11
Average rankings of the involved nine algorithms tested on the second group of benchmark functions for both $D = 30$ and $D = 50$.

Algorithms	$D = 30$	$D = 50$
ABC	6.20	5.30
GBABC	5.07	5.39
AABC	6.55	6.16
iqABC	5.32	5.57
MEABC	5.07	5.34
ABCVSS	4.52	4.91
DFSABC_elite	**3.63**	**3.50**
MABC-NS	4.88	5.09
MGABC	3.77	3.73

which has plenty of local optima [41]. More importantly, the problem is NP-hard and has no polynomial time solution, hence the EAs are suitable to solve it. The following formulas are used to express the problem.

$$\text{globalmin}_{x \in X} f(x) = \max\{\phi_1(x), \ldots, \phi_{2m}(x)\} \tag{9}$$

where $X = \{(x_1, \ldots, x_D) \in R^D \mid 0 \leqslant x_j \leqslant 2\pi, j = 1, \ldots, n\}, m = 2D - 1$ and

$$\phi_{2i-1}(x) = \sum_{j=i}^{D} \cos\left(\sum_{k=|2i-j-1|+1}^{j} x_k\right), \quad i = 1, \ldots, D \tag{10}$$

$$\phi_{2i}(x) = 0.5 + \sum_{j=i+1}^{D} \cos\left(\sum_{k=|2i-j|+1}^{j} x_k\right), \quad i = 1, \ldots, D - 1 \tag{11}$$

$$\phi_{m+i}(x) = -\phi_i(x), \quad i = 1, \ldots, m \tag{12}$$

The above established model for this problem is characterized by the fact that the objective function is piecewise smooth.

In the subsection, we consider to solve the two most difficult instances of the SSRPCD problem [42], i.e., the dimension sizes include two cases: $D = 19$ and $D = 20$. The corresponding MaxFEs is set to $5000 \cdot D$, and the parameters of the MGABC

Table 12
The final results of the involved nine algorithms on the SSRPCD problem for both $D = 19$ and $D = 20$.

Algorithms	$D = 19$	$D = 20$
ABC	1.14E+00	1.19E+00
GBABC	1.16E+00	1.14E+00
AABC	1.22E+00	1.25E+00
iqABC	1.11E+00	1.19E+00
MEABC	1.13E+00	1.18E+00
ABCVSS	1.14E+00	1.19E+00
DFSABC_elite	9.58E−01	9.61E−01
MABC-NS	1.10E+00	1.15E+00
MGABC	**7.75E−01**	**7.03E−01**

are set to the same values as in the above subsections. What's more, the seven improved ABC variants showed in the subSection 4.4 are included as well. The basic ABC is involved as a baseline. The Table 12 presents the final average results which are obtained by conducting the algorithms for 30 times, and the best results are marked with **boldface**. From the results, it is clear that the MGABC performs better than other competitive algorithms in terms of the quality of the final solutions, which further verifies the efficiency of the MGABC.

5. Conclusions

As a swarm intelligence based optimization algorithm, ABC has attracted a lot of attention of researchers from various fields for its simplicity yet good performance. However, for some complex optimization problems, the performance of ABC is still unsatisfactory, such as slow convergence speed or premature convergence. The reason for this deficiency is that the solution search equation is good at exploration but poor at exploitation. Hence, in recent years, many improved ABC variants have been proposed which mainly focus on how to modify the solution search equation by utilizing the global best solution or some elite solutions. Although these methods are beneficial to enhance the exploitation, they also run the risk of causing the algorithm too greedy.

In this work, we propose an improved ABC variant based on multi-elite guidance with the aim of elaborately utilizing the valuable information from multiple elite solutions. In the proposed MGABC, we make two modifications based on the elite group to enhance the exploitation without sacrificing the exploration. First, two modified solution search equations are introduced into the employed bee phase and onlooker bee phase, respectively. Second, a modified neighborhood search operator is integrated into the framework of ABC as an additional search strategy. After combining these two modifications, the proposed MGABC can significantly improve the performance of ABC. Extensive experiments are carried out on 50 benchmark functions and one real-world optimization problem, and other seven well-established ABC variants are included in the comparison. The experiments show that our approach can achieve promising results on most of the test functions, which are better or at least comparable to its competitors. In the future, the MGABC can be applied to more real-world problems, such as the software modular clustering problem [43–45].

CRediT authorship contribution statement

Xinyu Zhou: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Jiaxin Lu:** Methodology, Software, Data curation, Writing - review & editing. **Junhong Huang:** Writing - review & editing. **Maosheng Zhong:** Formal analysis, Writing - review & editing. **Mingwen Wang:** Formal analysis, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

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] G. Karafotias, M. Hoogendoorn, Á.E. Eiben, Parameter control in evolutionary algorithms: trends and challenges, IEEE Trans. Evol. Comput. 19 (2) (2014) 167–187.
[2] Z.-J. Wang, Z.-H. Zhan, J. Zhang, Solving the energy efficient coverage problem in wireless sensor networks: a distributed genetic algorithm approach with hierarchical fitness evaluation, Energies 11 (12) (2018) 3526.

[3] Y.-N. Ma, Y.-J. Gong, C.-F. Xiao, Y. Gao, J. Zhang, Path planning for autonomous underwater vehicles: an ant colony algorithm incorporating alarm pheromone, IEEE Trans. Veh. Technol. 68 (1) (2018) 141–154.
[4] Z. Yang, H. Qiu, L. Gao, X. Cai, C. Jiang, L. Chen, Surrogate-assisted classification-collaboration differential evolution for expensive constrained optimization problems, Inf. Sci. 508 (2020) 50–63.
[5] H. Peng, Z. Guo, C. Deng, Z. Wu, Enhancing differential evolution with random neighbors based strategy, J. Comput. Sci. 26 (2018) 501–511.
[6] H. Peng, Z. Wu, Heterozygous differential evolution with taguchi local search, Soft. Comput. 19 (11) (2015) 3273–3291.
[7] W. Liu, Z. Wang, X. Liu, N. Zeng, D. Bell, A novel particle swarm optimization approach for patient clustering from emergency departments, IEEE Trans. Evol. Comput. 23 (4) (2018) 632–644.
[8] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, X.-L. Shen, A hybrid particle swarm optimization algorithm using adaptive learning strategy, Inf. Sci. 436 (2018) 162–177.
[9] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, L. Cui, Firefly algorithm with adaptive control parameters, Soft Comput. 21 (17) (2017) 5091–5102.
[10] H. Wang, W. Wang, X. Zhou, H. Sun, J. Zhao, X. Yu, Z. Cui, Firefly algorithm with neighborhood attraction, Inf. Sci. 382 (2017) 374–387.
[11] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (abc) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57.
[12] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[13] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (1) (2008) 687–697.
[14] F. Wang, Y. Li, A. Zhou, K. Tang, An estimation of distribution algorithm for mixed-variable newsvendor problems, IEEE Trans. Evol. Comput..
[15] F. Wang, Y. Li, H. Zhang, T. Hu, X.-L. Shen, An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization, Swarm Evol. Comput. 49 (2019) 220–233.
[16] F. Xu, H. Li, C.-M. Pun, H. Hu, Y. Li, Y. Song, H. Gao, A new global best guided artificial bee colony algorithm with application in robot path planning, Appl. Soft Comput. (2020) 1–31.
[17] W. Gao, H. Sheng, J. Wang, S. Wang, Artificial bee colony algorithm based on novel mechanism for fuzzy portfolio selection, IEEE Trans. Fuzzy Syst. 27 (5) (2018) 966–978.
[18] S. Sundar, P.N. Suganthan, C.T. Jin, C.T. Xiang, C.C. Soon, A hybrid artificial bee colony algorithm for the job-shop scheduling problem with no-wait constraint, Soft. Comput. 21 (5) (2017) 1193–1202.
[19] D. Bose, S. Biswas, A.V. Vasilakos, S. Laha, Optimal filter design using an improved artificial bee colony algorithm, Inf. Sci. 281 (2014) 443–461.
[20] J. Zhou, X. Yao, F.T. Chan, Y. Lin, H. Jin, L. Gao, X. Wang, An individual dependent multi-colony artificial bee colony algorithm, Inf. Sci. 485 (2019) 114–140.
[21] S. Aslan, H. Badem, D. Karaboga, Improved quick artificial bee colony (iqabc) algorithm for global optimization, Soft. Comput. 23 (24) (2019) 13161–13182.
[22] Q. Lin, M. Zhu, G. Li, W. Wang, L. Cui, J. Chen, J. Lu, A novel artificial bee colony algorithm with local and global information interaction, Appl. Soft Comput. 62 (2018) 702–735.
[23] M. Zhang, N. Tian, V. Palade, Z. Ji, Y. Wang, Cellular artificial bee colony algorithm with gaussian distribution, Inf. Sci. 462 (2018) 374–401.
[24] D. Bajer, B. Zorić, An effective refined artificial bee colony algorithm for numerical optimisation, Inf. Sci. 504 (2019) 221–275.
[25] D. Kumar, K. Mishra, Co-variance guided artificial bee colony, Appl. Soft Comput. 70 (2018) 86–107.
[26] Y. Xue, J. Jiang, B. Zhao, T. Ma, A self-adaptive artificial bee colony algorithm based on global best for global optimization, Soft. Comput. 22 (9) (2018) 2935–2952.
[27] L. Cui, G. Li, Z. Zhu, Q. Lin, Z. Wen, N. Lu, K.-C. Wong, J. Chen, A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization, Inf. Sci. 414 (2017) 53–67.
[28] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (7) (2010) 3166–3173.
[29] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. 192 (2012) 120–142.
[30] X. Zhou, Z. Wu, H. Wang, S. Rahnamayan, Gaussian bare-bones artificial bee colony algorithm, Soft. Comput. 20 (3) (2016) 907–924.
[31] W.-J. Yu, Z.-H. Zhan, J. Zhang, Artificial bee colony algorithm with an adaptive greedy position update strategy, Soft. Comput. 22 (2) (2018) 437–451.
[32] W. Gao, S. Liu, A modified artificial bee colony algorithm, Comput. Oper. Res. 39 (3) (2012) 687–697.
[33] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-S. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. 279 (2014) 587–603.
[34] M.S. Kiran, H. Hakli, M. Gunduz, H. Uguz, Artificial bee colony algorithm with variable search strategy for continuous optimization, Inf. Sci. 300 (2015) 140–157.
[35] L. Cui, G. Li, Q. Lin, Z. Du, W. Gao, J. Chen, N. Lu, A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation, Inf. Sci. 367 (2016) 1012–1044.
[36] X. Zhou, H. Wang, M. Wang, J. Wan, Enhancing the modified artificial bee colony algorithm with neighborhood search, Soft. Comput. 21 (10) (2017) 2733–2743.
[37] J. Lu, X. Zhou, Y. Ma, M. Wang, An elite group guided artificial bee colony algorithm with a modified neighborhood search, in, in: The 15th Pacific Rim International Conference on Artificial Intelligence, Springer, 2018, pp. 387–394.
[38] W. Gao, S. Liu, L. Huang, A novel artificial bee colony algorithm based on modified search equation and orthogonal learning, IEEE Trans. Cybern. 43 (3) (2013) 1011–1024.
[39] H. Wang, H. Sun, C. Li, S. Rahnamayan, J.-S. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[40] J. Liang, B. Qu, P. Suganthan, A.G. Hernández-Díaz, Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization, Tech. Rep. 34, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report, 2013..
[41] S. Das, P. Suganthan, Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems (Tech. rep.), Jadavpur University and Nanyang Technological University, 2011.
[42] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[43] W. Pan, B. Li, J. Liu, Y. Ma, B. Hu, Analyzing the structure of java software systems by weighted k-core decomposition, Future Gen. Comput. Syst. 83 (2018) 431–444.
[44] W. Pan, H. Ming, C. Chang, Z. Yang, D.-K. Kim, Elementrank: ranking java software classes and packages using a multilayer complex network-based approach, IEEE Trans. Software Eng..
[45] W. Pan, B. Song, K. Li, K. Zhang, Identifying key classes in object-oriented software using generalized k-core decomposition, Future Gen. Comput. Syst. 81 (2018) 188–202.