Information Sciences 527 (2020) 227–240

Contents lists available at ScienceDirect

Information Sciences

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

Improving artificial Bee colony algorithm using a new neighborhood selection mechanism

Hui Wang[a,*], Wenjun Wang[b,*], Songyi Xiao[a], Zhihua Cui[c], Minyang Xu[a], Xinyu Zhou[d]

[a] *Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang Institute of Technology, Nanchang 330099, China*
[b] *School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China*
[c] *School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China*
[d] *College of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China*

A R T I C L E   I N F O

Article history:
Received 31 October 2019
Revised 31 January 2020
Accepted 21 March 2020
Available online 12 April 2020

Keywords:
Artificial bee colony (ABC)
Neighborhood selection
Probability selection
Ring topology
Opposition-based learning
Optimization

A B S T R A C T

Artificial bee colony (ABC) and its most modifications use a probability method to select good food sources (called solutions) in the onlooker bee search phase. However, the probability selection does not work with increasing of iterations, because the fitness values cannot be used to distinguish two different solutions. In order to tackle this problem, this paper proposes a new ABC (called NSABC), in which a new selection method based on neighborhood radius is used. Unlike the probability selection in the original ABC, NSABC chooses the best solution in the neighborhood radius to generate offspring. Based on the neighborhood radius, two new solution search strategies are modified. The scout bee search phase is improved by using opposition-based learning and the neighborhood radius. To evaluate the search ability of NSABC, there are 22 benchmark problems used in the experiments. Performance comparison shows NSABC achieves better results than five other ABC algorithms.

1. Introduction

Optimization problems arise in many engineering problems. To solve these optimization problems, people always try to design some efficient optimization algorithms/technqieus. Especially for complex and difficult optimization problems, strong optimization algorithms are needed. Then, various bio-inspired optimization approaches were proposed/designed, such as ant colony optimization (ACO) [3,26], particle swarm optimization (PSO) [37,40], artificial bee colony (ABC) [2,20], firefly algorithm (FA) [41,42], and cuckoo search (CS) [47]. Compared with pure mathematical methods, those bio-inspired algorithms have simpler concepts, easier implementations and stronger search abilities [7,8,22,28,36,46].

Among the above bio-inspired algorithms, ABC has two significant advantages: strong exploration and few control parameters. Recently, ABC was successfully applied to many real-world and benchmark problems. However, the original ABC is not perfect. For example, ABC is not good at exploitation and cannot obtain high accuracy solutions. As the iteration increases, the probability selection method in the onlooker bee search phase does not work. Due to the fitness evaluation function, two

[*] Corresponding authors.
E-mail addresses: huiwang@whu.edu.cn (H. Wang), wangwenjun881@126.com (W. Wang), speaknow@126.com (S. Xiao), zhihuacui@gmail.com (Z. Cui), ncxmy2019@163.com (M. Xu), xyzhou@jxnu.edu.cn (X. Zhou).

https://doi.org/10.1016/j.ins.2020.03.064
0020-0255/© 2020 Elsevier Inc. All rights reserved.

solutions with different objective values may have the same fitness value. In order to overcome these drawbacks, this paper proposes a new ABC (called NSABC), in which a new selection mechanism is employed based on a concept of neighborhood radius for onlooker bees. Compared to the probability selection in the original ABC, NSABC chooses the best solution in a neighborhood radius to generate offspring. Based on the neighborhood radius, two new solution search strategies are modified. Morevoer, the scout bee search phase is improved by using opposition-based learning and the neighborhood radius. Experiments are conducted on 22 benchmark problems. Performance of NSABC is compared with ABC and four improved ABC variants on dimensions 30 and 100. Computational results demonstrate NSABC is effective on most test problems.

The reminder of this work is organized as follows. The original ABC is briefly presented in Section 2. Recent advances about ABC are presented in Section 3. The new neighborhood selection, modified solution search strategies, improved scout bee search phase, and framework of NSABC are described in Section 4. Experiments and discussions are shown in Section 5. Finally, the conclusion is given in Section 6.

2. Artificial bee colony algorithm

Problem solving from nature provides important references in optimization community. In view of this, different optimization algorithms inspired by nature were proposed. In 2005, Karaboga [20] designed a new intelligent algorithm called artificial bee colony (ABC), and it was motivated by the intelligent behaviors of bees. Over the years, ABC has become one of the most popular optimization algorithms, just like ACO and PSO.

In ABC, bees always try to search good food sources with many nectar. By the suggestions of [20], bees in the swarm are divided into three classes: employed bees, onlooker bees, and scout bees. For a given problem, each food source indicates a candidate solution in the search space. The nectar amount of the food source is related to the fitness value of the corresponding solution. During the iterations, different bees use three main operations to search good solutions.

The initial swarm has *SN* solutions and $X_i = x_{i,1}, x_{i,2}, \ldots, x_{i,D}$ is the *i*th solution in the swarm, where *SN* is the size of swarm and *D* is dimension size. The detailed operations of ABC are described as below.

Employed bee search phase –At this stage, employed bees search around each solution X_i $(i = 1, 2, \ldots, SN)$, and attempt to find better solutions [20].

$$v^*_{i,jr} = x_{i,jr} + \phi_{i,jr} \cdot (x_{i,jr} - x_{k,jr}), \tag{1}$$

where X_k is randomly taken from the whole swarm ($X_k \neq X_i$), and *jr* is a random integer in the range [1, *D*]. The weight $\phi_{i,jr}$ is randomly generated between -1 and 1. After the above search (Eq. (1)), a new solution V_i is generated as below.

$$v_{i,j} = \begin{cases} v^*_{i,jr}, & \text{if } j = jr \\ x_{i,j}, & \text{otherwise} \end{cases}, \tag{2}$$

where $j = 1, 2, \ldots, D$.

From Eq. 2, a new solution V_i is obtained. To gain a better solution, a greedy selection method is utilized as below.

$$X_i = \begin{cases} V_i, & \text{if } V_i \text{ outperforms } X_i \\ X_i, & \text{otherwise} \end{cases} \tag{3}$$

Onlooker bee search phase –At this stage, the employed bees complete the neighborhood search of all solutions. Then, the onlooker bees get the search information from the employed bees. Unlike the employed bees, only some better solutions are selected to do further search at this stage. To select better solutions, a probability p_i is defined for each solution. The probability p_i is computed by [20]

$$p_i = \frac{fit(X_i)}{\sum_{i=1}^{SN} fit(X_i)}, \tag{4}$$

where *fit*(X_i) is the fitness value of X_i, and it can be calculated by [20]

$$fit(X_i) = \begin{cases} \frac{1}{1+f(X_i)}, & \text{if } f(X_i) \geq 0 \\ 1 + |f(X_i)|, & \text{if } f(X_i) < 0 \end{cases}, \tag{5}$$

where *f*(X_i) is the objective function value of X_i.

A good solution X_i is selected based on p_i for each onlooker bee. Then, an offspring V_i is generated by Eqs. (1) and (2). Based on Eq. (3), a better solution between X_i and V_i is retained.

Scout bee search phase –According to Eq. (3), new solutions compete with their parent solutions. When V_i is better than X_i, it means that the neighborhood search on X_i is successful. When V_i is worse than X_i, it indicates that the neighborhood search fails. A failure counter $trial_i$ is used to monitor the neighborhood search of X_i. For the successful search, $trial_i$ is initialized to 0. For failure search, $trial_i$ is added by 1. The update of $trial_i$ can be described as below.

$$trial_i = \begin{cases} 0, & \text{if } V_i \text{ is better than } X_i \\ trial_i + 1, & \text{otherwise} \end{cases} \tag{6}$$

If $trial_i$ is greater than a control parameter (called *limit*), the corresponding solution X_i is abandoned. To replace the abandoned X_i, a new one is created as below [20].

$$x_{i,j} = low_j + rand(0,1) \cdot (up_j - low_j), \tag{7}$$

where $j = 1, 2, \ldots, D$, $rand(0, 1)$ is randomly generated in [0,1], and $[low_j, up_j]$ is the boundary constraint.

3. Recent work on ABC

In order to enable ABC to handle various problems, lots of modified ABC variants and their applications were proposed. In the following section, the research progress of ABC is briefly presented.

To accelerate the search and improve the exploitation, the global best solution (X_{best}) is usually used to modify the solution search equations [5,29,45,49]. Zhu and Kwong [49] added a difference vector ($X_{best} - X_i$) to the search equation of the original ABC. Banharnsakun et al. [5] designed another search strategy based on X_{best}, in which the fitness value and position vector of X_{best} are utilized. In [45], different strategies based on X_{best} can be self-adaptively chosen. Peng et al. [29] employed the best neighbor to modify the search strategy. And it is also used in the scout bees to maintain the search experiences.

The original ABC uses only one search strategy (Eq. 1) to update the solutions. Recently, two ore more search strategies were introduced into ABC to strengthen the search performance. Wang et al. [43] used three search strategies. At the initial stage, each solution is randomly allocated a strategy. As the iteration increases, the assigned strategy is dynamically updated. Based on the historical search experiences, Gao et al. [14] designed a probability to choose a suitable search strategy. In [21], five search strategies were employed, and each strategy has an independently selection probability. By monitoring the quality of new solutions, the selection probability of each strategy is dynamically changed in the search process. In [17], three search strategies and other modifications were used to enhance the performance. In [32], two search strategies with different characteristics were used. In [33], ten search strategies based on three kinds of search regions were designed.

ABC updates only one dimension. In [1], a modification rate was introduced to update more dimensions. Gaussian sampling was successfully used to generate good solutions in ABC [13,48]. To help ABC run on hardware, the cost or space should be reduced. To tackle this issue, Banitalebi et al. [6] implemented a modified compact ABC. In [9], a depth-first search method was proposed to allocate more computational efforts to better solutions. This may be helpful to improve the efficiency of evolution. Moreover, some elite solutions are used to modify the search strategies. Cui et al. [11] proposed a dynamic method to update the colony size. If the current search exhibits good exploration, the colony size will be decreased to enhance the exploitation; otherwise the colony size will be increased to improve the exploration. Li and Yang [24] introduced a memory to store the previous successful search information. Firstly, a memory matrix M is built. Then, each element in M has two components: the index of selected neighbor (k) and the weight value ($\phi_{i,j}$). In [10], the selection probability p_i for each solution X_i is defined as $p_i = (R_i/SN)^{\alpha}$, where R_i is the ranking value of X_i and α is a predefined nonnegative parameter. Xiao et al. [44] introduced an elite method and dimensional learning into ABC. Better solutions are selected based on the elite method to accelerate the search process. The dimension differences between two different solutions are used to generate new solutions. This may help to strengthen the global search.

Recently, some scholars combined ABC with differential evolution (DE) to improve the search. In [16], ABC/best/1 is designed based on DE mutation operator. In [15], GABC and DE were hybridized. Based on a probability, GABC and DE are executed alternately for each solution in the swarm. In [25], the employed bees used DE operators, and the onlooker bees used another different operator. Solutions in the swarm are sorted. Then, the swarm is divided into m groups and each group contains n solutions ($m = SN/n$). Each group has the same rank. For example, solutions in the first group have the rank 1. For the last group, the rank is m. Based on the group rank, the selection probability for onlooker bees is defined.

Unlike single objective optimization, multi-objective optimization problems have more than 2 objectives [35,38]. Saad et al. [31] designed a multi-objective ABC for the network topology of computer communication, in which five objectives including reliability, availability, mean link utilization, cost, and delay are considered. Ozturk et al. [27] presented a new binary ABC based on genetic operators. At the initial stage, each component of a solution is randomly assigned 0 or 1. Two modified crossover and swap operators are designed. The proposed algorithm was tested on three different problems including image clustering, Knapsack problem and numerical optimization. Localization is an important technique in wireless sensor network [23]. In [34], a localization method based on ABC was proposed and results showed good performance. By using some top solutions in the swarm, Bajer and Zorić [4] proposed a modified the search strategy, which is similar to GABC. By combining X_{best} and two randomly chosen solutions, a new solution is produced to replace the abandoned one. A quick ABC programming approach was used for symbolic regression [19]. Garg et al. [18] designed a new ensemble ABC for cloud environment.

4. Proposed approach

In this section, a modified ABC variant based on neighborhood selection (called NSABC) is proposed. The NSBAC employs three modifications. Firstly, a new neighborhood selection mechanism is designed based on neighborhood radius in a ring topology. The probability selection method is replaced by the neighborhood selection. Secondly, two modified search strategies are constructed based on the neighborhood radius. Finally, the scout bee search phase is improved by using the neighborhood radius and opposition-based learning.

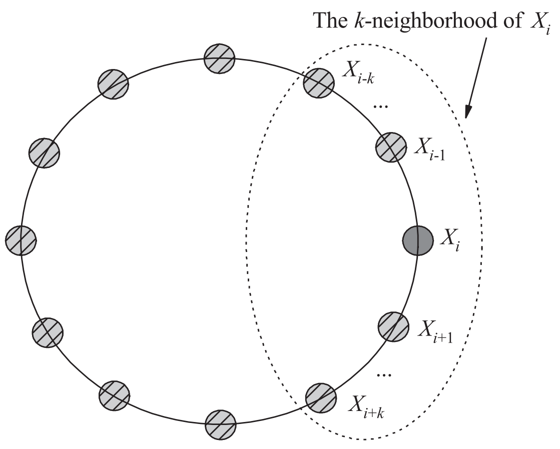


Fig. 1. The k-neighborhood of X_i in the ring topology.

4.1. Neighborhood selection

The objective function value of a solution is converted into fitness value in terms of Eq. (5). For a given solution X_i, when its objective function value $f(X_i)$ is very small (e.g. less than 1.0E–20) and satisfies $f(X_i) \geq 0$, the fitness value $fit(X_i)$ will be rounded up to 1, because the small value $f(X_i)$ is ignored by the fitness function [5]. For two different solutions X_1 and X_2, their objective function values are 1.0E–20 and 1.0E-30, respectively. Then, their fitness values are 1/(1+1.0E–20)=1 and 1/(1+1.0E–30)=1, respectively. It is obvious that we cannot use the fitness value to distinguish X_1 and X_2, because their fitness values are the same. If we use fitness value to measure the quality of solutions, more accurate solutions cannot be achieved according to the greedy selection (see Eq. (3)). Finally, the search is stagnant.

To tackle this issue, the objective function value is directly used to compare and select better solutions [5]. For the selection probability p, the literature [5] still uses Eq. (4) to select good solutions. The probability p is defined based on fitness value. Then, there is a strange phenomenon. As the iteration increases, the swarm is converging. When the objective function values of all solutions in the swarm are less than 1.0E–20 and greater than 0, the fitness values of all solutions are 1. So, all solutions have the same selection probability $1/SN$. Better solutions will not have higher selection probability. To improve this case, various strategies were proposed, such as elite set [9], population ranking [10,17], and normalization for objective function value [11].

In this paper, a new neighborhood selection method is proposed to replace the original probability selection. Firstly, all solutions in the colony are assumed to form a ring topology in terms of their indices [12]. For instance, X_1 connects with X_2 and X_{SN}. Secondly, a concept of k-neighborhood is used [12], where the parameter k is the neighborhood radius. For the solution X_i, its k-neighborhood contains $2k+1$ solutions $\{X_{i-k}, \ldots, X_i, \ldots, X_{i+k}\}$. The neighborhood radius k should satisfy $1 \leq k \leq \frac{SN-1}{2}$. Fig. 1 illustrates how to construct the k-neighborhood of a solution.

Based on the concept of k-neighborhood, a novel solution selection mechanism is proposed for the onlooker bee search phase. For each solution X_i in the swarm, the best solution in the k-neighborhood of X_i is chosen as X_{ib}. Then, we use the corresponding search strategy searches around X_{ib}. The k-neighborhood of X_i has $2k+1$ solutions, and X_{ib} is the best one among them. So, some better solutions in the swarm are selected for further search. Compared with most ABC algorithms, our method does not need to calculate the selection probability of each solution.

4.2. Modified search strategies

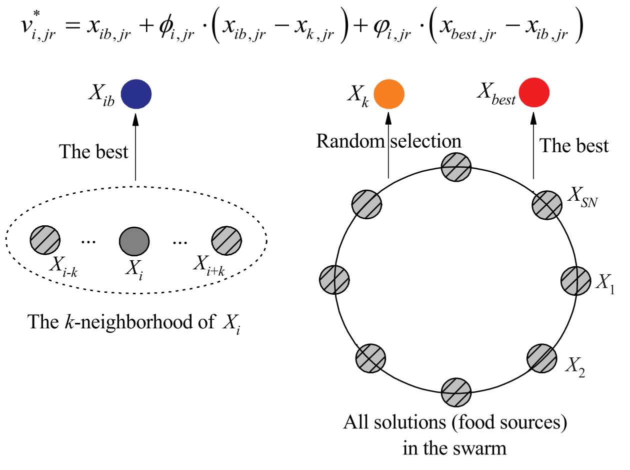
By combining the concept of k-neighborhood and GABC, a modified search strategy for the employed bees is proposed, where X_i is replaced by X_{ib} in the original search of GABC. Then, the employed bees do not search the neighborhood of all solutions X_i, and they only search around X_{ib}. The detailed modification is given as follows.

$$v^*_{i,jr} = x_{ib,jr} + \phi_{i,jr} \cdot (x_{ib,jr} - x_{k,jr}) + \varphi_{i,jr} \cdot (x_{best,jr} - x_{ib,jr}), \tag{8}$$

where X_{ib} is the best solution chosen from the k-neighborhood of X_i. The weight factors $\phi_{i,jr} \in [-1,1]$ and $\varphi_{i,jr} \in [0,C]$ are two random values, where C is equal to 1.5 by the suggestions of [49]. Fig. 2 shows the solution selection of the modified search strategy in the employed bee search phase. Based on Eqs. (2) and (8), a new solution V_i is produced. According to Eq. (3), the better solution between V_i and X_i is chosen as the new X_i.

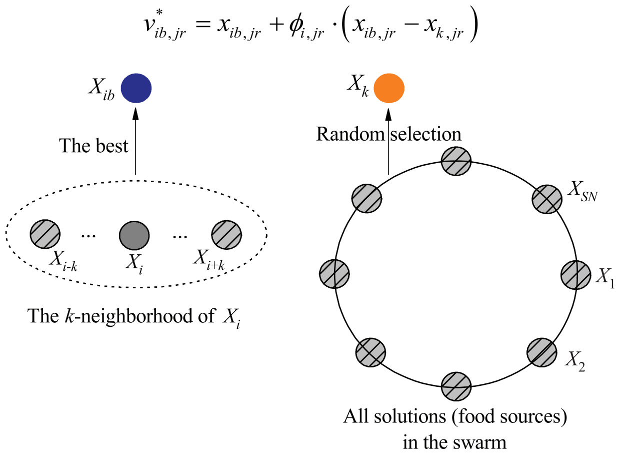
For the ith onlooker bee, X_{ib} is chosen from the k-neighborhood of X_i. Inspired by Eq. (6) in [43], an improved search strategy is proposed as below.

$$v^*_{ib,jr} = x_{ib,jr} + \phi_{i,jr} \cdot (x_{ib,jr} - x_{k,jr}), \tag{9}$$



$v^*_{i,jr} = x_{ib,jr} + \phi_{i,jr} \cdot (x_{ib,jr} - x_{k,jr}) + \varphi_{i,jr} \cdot (x_{best,jr} - x_{ib,jr})$
X_{ib}
The best
X_{i-k}
X_i
X_{i+k}
The k-neighborhood of X_i
X_k
Random selection
X_{best}
The best
X_{SN}
X_1
X_2
All solutions (food sources) in the swarm

Fig. 2. The modified search equation based on k-neighborhood for employed bees.



$v^*_{ib,jr} = x_{ib,jr} + \phi_{i,jr} \cdot (x_{ib,jr} - x_{k,jr})$
X_{ib}
The best
X_{i-k}
X_i
X_{i+k}
The k-neighborhood of X_i
X_k
Random selection
X_{SN}
X_1
X_2
All solutions (food sources) in the swarm

Fig. 3. The modified search equation based on k-neighborhood for onlooker bees.

$$v_{ib,j} = \begin{cases} v^*_{ib,jr}, & \text{if } j = jr \\ x_{ib,j}, & \text{otherwise} \end{cases} \tag{10}$$

where $j = 1, 2, \ldots, D$. Fig. 3 shows the solution selection of the modified search strategy in the onlooker bee search phase. For V_{ib} and X_{ib}, the better one is chosen according to Eq. 11.

$$X_{ib} = \begin{cases} V_{ib}, & \text{if } f(V_{ib}) < f(X_{ib}) \\ X_{ib}, & \text{otherwise} \end{cases} \tag{11}$$

4.3. Modified scout bee search phase

For the solution X_i, the failure counter $trial_i$ monitors its search status. A large $trial_i$ means that X_i is stagnant in many iterations. When $trial_i \geq limit$, X_i is abandoned. Then, a new X_i is randomly generated to substitute for the abandoned one. As the iteration increases, the swarm is converging. The search region of the swarm becomes small. When adding a random solution in the swarm, the search region will enlarge. Finally, the convergence is slowed down.

To tackle the above issue, we use a new method to update the abandoned solution. When X_i is abandoned, three solutions U_1, U_2, and U_3 are produced. Then, the best solution among U_1, U_2, and U_3 is selected to replace the abandoned X_i. Like the original ABC, U_1 is randomly generated according to Eq. (7).

For U_2, the best solution X_{ib} is chosen in the k-neighborhood of X_i. Then, U_2 is produced around X_{ib} as follows.

$$u_{2,j} = x_{ib,j} + rand(0,1) \cdot (x_{r1,j} - x_{r2,j}), \tag{12}$$

where $j = 1, 2, \ldots, D$, X_{r1} and X_{r2} are two different solutions randomly selected from the swarm, and $r1 \neq r2 \neq ib$.

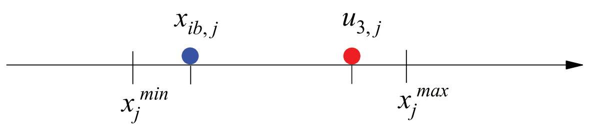


Fig. 4. The current solution (X_{ib}) and its opposite solution (U_3) in one dimension.

Opposition-based learning (OBL) [30] is an effective method, which can enhance the probability of find better candidate solutions. Then, the third solution U_3 is generated by the OBL. The best neighbor X_{ib} is determined in the k-neighborhood of the abandoned solution X_i. Based on X_{ib}, an opposite solution U_3 is generated as follows.

$$u_{3,j} = x_j^{min} + x_j^{max} - x_{ib,j}, \tag{13}$$

$$x_j^{min} = \min\{x_{i,j}\}, \quad x_j^{max} = \max\{x_{i,j}\}, \tag{14}$$

$$i = 1, 2, \dots, SN, \quad j = 1, 2, \dots, D,$$

where $[x_j^{\min}, x_j^{\max}]$ is the boundary of the current swarm, and it is dynamically updated by Eq. (14). To have clear illustration, Fig. (4) shows X_{ib} and U_3 in one dimension. As seen, $| x_{ib,j} - x_j^{\min} |$ and $| x_j^{\max} - u_{3,j} |$ are equal for each dimension. When the current X_{ib} is stagnant, we try to check its opposite side U_3. This can help to increase the probability of finding better candidate solutions [30].

When all three new solutions U_1, U_2, and U_3 are generated, a simple elite method is used. The best solution among U_1, U_2, and U_3 is selected to replace the abandoned solution X_i.

4.4. Framework of NSABC

The proposed NSABC has three modifications: 1) a new solution selection method based on k-neighborhood; 2) two modified search strategies based on k-neighborhood for employed and onlooker bees; and 3) an improved scout bee search phase. The framework of NSABC is given in Algorithm 1, where *FEs* is the number of function evaluations, and *MaxFEs* is the maximum value of *FEs*.

The quantity of solutions, employed bees, and onlooker bees are the same (*SN*). For *SN* employed bees, we can get *SN* offspring. Similarly, another *SN* offspring are generated by onlooker bees. In the scout bee search phase, three different solutions are produced for each abandoned solution. The first solution U_1 is randomly distributed in the whole search range. When the search of current swarm is stagnant, the random solution many help it jump out of the local minimum. The second solution U_2 is generated around the best solution X_{ib} in the k-neighborhood of the abandoned solution X_i. Though X_i is trapped, searching around X_{ib} may find a better solution. By exploring the opposition of X_{ib}, we get the third solution U_3. Compared to random search, opposition-based learning can improve the probability of find better solutions [30]. By comparing U_1, U_2, and U_3, the best one is selected to replace the abandoned X_i.

Compared with the original ABC, the proposed strategies do not increase the computational time complexity. The complexity of ABC is affected by three main factors: population size (*SN*), complexity of objective function (*O(f)*), and the maximum number of iterations ($T_{\max}$). Then, the complexity of the original ABC is $O(T_{\max} \cdot (SN \cdot f + SN \cdot f + f)) = O(T_{\max} \cdot SN \cdot f)$. For our approach NSABC, its complexity is $O(T_{\max} \cdot (SN \cdot f + SN \cdot f + 3 \cdot f)) = O(T_{\max} \cdot SN \cdot f)$. So, both NSABC and ABC have the same complexity.

5. Experimental study

5.1. Test problems

In order to evaluate the performance of NSABC, 22 benchmark problems are utilized. A simple descriptions of these problems are described in Table 1. The detailed mathematical definitions can be found in [9,39]. In this paper, problems with $D = 30$ and 100 are considered. The global minimum of each problem is given in the last column of Table 1.

5.2. Effects of the neighborhood radius k

The proposed NSABC is inspired by the concept of k-neighborhood. There are $2k+1$ solutions ($1 \le k \le \frac{SN-1}{2}$) in the k-neighborhood of X_i. Among $2k+1$ solutions, the best one is selected as X_{ib}. When the value of k tends to $\frac{SN-1}{2}$, the size of the k-neighborhood approaches to the whole swarm and each X_{ib} is close to X_{best}. So, the neighborhood radius k may affect the performance of NSABC.

To evaluate the effects of the neighborhood radius k, different values of k are tested in this section. The swarm size *SN* is equal to 50 in many ABC references, and the parameter k should satisfy $1 \le k \le 24$. In order to choose the best k, NSABC

Algorithm 1: Proposed Approach (NSABC).

```
1  Randomly produce $SN$ initial solutions and compute their objectivefunction values;
2  Set $FEs = SN$, and $trial_i = 0$, $i = 1, 2, \ldots, SN$;
3  while $FEs \le MaxFEs$ do
       /* Employed bee search phase */
4      for $i = 1$ to $SN$ do
5          Choose the best solution $X_{ib}$ in the $k$-neighborhood of $X_i$;
6          Generate $V_i$ by Eqs. 2 and 8;
7          Calculate the objective function value of $V_i$, and $FEs$++;
8          Update $X_i$ according to Eq. 3;
9          Update $trial_i$ according to Eq. 6;
10     end
       /* Onlooker bee search phase */
11     for $i = 1$ to $SN$ do
12         Choose the best solution $X_{ib}$ in the $k$-neighborhood of $X_i$;
13         Generate $V_{ib}$ according to Eqs. 9and 10;
14         Compute the objective function value of $V_{ib}$, and $FEs$++;
15         Update $X_{ib}$ according to Eq. 11;
16         Update $trial_{ib}$ according to Eq. 6;
17     end
       /* Scout bee search phase */
18     if $max\{trial_i\} \le limit$ then
19         Generate $U_1$ according to Eq. 7;
20         Generate $U_2$ according to Eq. 12;
21         Generate $U_3$ according to Eq. 13;
22         Calculate the objective function values of $U_1$, $U_2$, and $U_3$,and $FEs = FEs + 3$;
23         Select the best one among $U_1$, $U_2$, and $U_3$ to replace theabandoned $X_i$;
24     end
25 end
26 Output the best solution;
```

with different k is tested. In the following experiment, k is set to 1, 5, 10, 15, and 24, respectively. For other parameters, $SN = 50$, $D = 30$, $MaxFEs = 3000 \cdot D$, $C = 1.5$, and $limit = 100$ are used. For each problem, NSABC with each k is run 30 trials.

Table 2 gives the comparison results of NSABC with different neighborhood radius k, where “Mean”represents the mean best objective function value. For unimodal problems $f_1 - f_9$ except for f_6, all five different k values can help NSABC achieve good solutions. On f_6, all NSABC instances fall into local minimum. For f_7, the obtained solutions are the global optimum. A larger k is better for problems $f_1 - f_5$. NSABC with $k \ge 10$ achieves more accurate solutions than other cases. For problem f_1, $k = 15$ performs better than other k values. $k = 24$ obtains the best results on three problems (f_2, f_3, and f_5), and $k = 10$ is the best on two problems (f_4 and f_6). The parameter k cannot affect the performance of NSABC on f_7 and f_8. For multimodal problems $f_{10} - f_{22}$, NSABC with all k values converge to the global optimum on four problems (f_{11}, f_{12}, f_{20}, and f_{21}). Among five k values, only $k = 24$ fails to help NSABC find the global optimum on f_{13} and f_{14}. NSABC with all k values obtain the same results on f_{16}, f_{17}, and f_{19}. For f_{15}, The performance of all cases is similar.

From the above analysis, all five k values can help ABC achieve good solutions on most test problems. A large or a small k value is still effective. To choose the best setting of the parameter k, Friedman test is utilized to compute the mean ranking of NSABC with each k value [43]. Table 3 lists the mean ranking values of different NSABCs. From the results, $k = 10$ wins the best ranking. It demonstrates that the neighborhood radius $k = 10$ can help NSABC obtain better performance than other k values. $k = 15$ is slightly worse than $k = 10$. For two extreme values $k = 1$ and $k = 24$, their mean ranking values are the worst, but $k = 24$ is a little better. Therefore, $k = 10$ is used in this paper.

5.3. Comparison of NSABC with other ABCs

To further evaluate the effectiveness of NSABC, several other ABCs are used for comparisons. The related algorithms are presented as below.

- The original ABC [20].
- The global best guided ABC (GABC) [49].
- Modified ABC (MABC) [16].
- Multi-strategy ensemble of ABC (MEABC) [43].

Table 1
Test problems.

Problem name	Search range	Global minimum
Sphere (f_1)	$[-100, 100]$	0
Elliptic (f_2)	$[-100, 100]$	0
SumSquare (f_3)	$[-10, 10]$	0
SumPower (f_4)	$[-1, 1]$	0
Schwefel 2.22 (f_5)	$[-10, 10]$	0
Schwefel 2.21 (f_6)	$[-100, 100]$	0
Step (f_7)	$[-100, 100]$	0
Exponential (f_8)	$[-10, 10]$	0
Quartic (f_9)	$[-1.28, 1.28]$	0
Rosenbrock (f_{10})	$[-5, 10]$	0
Rastrigin (f_{11})	$[-5.12, 5.12]$	0
NCRastrigin (f_{12})	$[-5.12, 5.12]$	0
Griewank (f_{13})	$[-600, 600]$	0
Schwefel 2.26 (f_{14})	$[-500, 500]$	$-418.98289 \cdot D$
Ackley (f_{15})	$[-50, 50]$	0
Penalized 1 (f_{16})	$[-100, 100]$	0
Penalized 2 (f_{17})	$[-100, 100]$	0
Alpine (f_{18})	$[-10, 10]$	0
Levy (f_{19})	$[-10, 10]$	0
Weierstrass (f_{20})	$[-1, 1]$	0
Himmelblau (f_{21})	$[-5, 5]$	−78.33
Michalewicz (f_{22})	$[0, \pi]$	$-D$

Table 2
Comparison results of NSABC with different neighborhood radius k.

Problem	$k = 1$	$k = 5$	$k = 10$	$k = 15$	$k = 24$
	Mean	Mean	Mean	Mean	Mean
f_1	6.64E−51	5.34E−69	1.44E−80	**4.03E−82**	1.82E−80
f_2	5.66E−46	9.76E−66	1.73E−74	8.07E−77	**1.22E−80**
f_3	1.98E−52	1.80E−70	3.42E−79	3.21E−83	**1.87E−86**
f_4	1.77E−75	1.66E−88	**1.19E−94**	5.63E−92	2.23E−93
f_5	6.68E−27	1.61E−36	5.81E−42	2.33E−42	**2.63E−44**
f_6	3.18E+00	1.73E+00	**1.39E+00**	1.46E+00	1.68E+00
f_7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_8	**7.18E−66**	**7.18E−66**	**7.18E−66**	**7.18E−66**	**7.18E−66**
f_9	3.53E−02	3.28E−02	**2.37E−02**	2.87E−02	3.71E−02
f_{10}	**2.25E−02**	5.09E−02	7.14E−02	2.57E−01	7.82E−01
f_{11}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{13}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.66E−13
f_{14}	**−12569.5**	**−12569.5**	**−12569.5**	**−12569.5**	−12451.9
f_{15}	2.62E−14	2.70E−14	1.72E−14	**1.48E−14**	1.84E−14
f_{16}	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**
f_{17}	**1.35E−32**	**1.35E−32**	**1.35E−32**	**1.35E−32**	**1.35E−32**
f_{18}	6.42E−17	**1.40E−20**	9.43E−16	1.39E−15	5.61E−15
f_{19}	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**
f_{20}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{21}	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**
f_{22}	**−2.96E+01**	**−2.96E+01**	**−2.96E+01**	**−2.96E+01**	−2.94E+01

Table 3
Mean ranking values of SNABC with different neighborhood radius k.

Algorithm	Mean ranking
$k = 1$	3.43
$k = 5$	3.16
$k = 10$	**2.57**
$k = 15$	2.61
$k = 24$	3.23

Table 4
Comparison results of six ABC algorithms when $D = 30$. The best results are shown in bold.

Problem	ABC Mean	GABC Mean	MABC Mean	MEABC Mean	BABC Mean	NSABC Mean
f_1	9.49E−16	4.45E−16	3.75E−26	3.12E−40	2.48E−56	**1.44E−80**
f_2	3.41E−08	5.32E−16	5.61E−23	3.93E−37	1.59E−56	**1.73E−74**
f_3	7.22E−16	5.97E−16	3.71E−41	5.26E−41	1.88E−55	**3.42E−79**
f_4	3.38E−14	6.63E−17	1.52E−52	3.30E−90	5.89E−144	**1.19E−94**
f_5	1.58E−10	1.61E−15	4.29E−14	1.62E−21	2.71E−30	**5.81E−42**
f_6	3.72E+01	1.45E+01	1.07E+01	4.57E+00	1.76E+01	**1.39E+00**
f_7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_8	1.36E−22	9.08E−24	**7.18E−66**	**7.18E−66**	**7.18E−66**	**7.18E−66**
f_9	1.86E−01	1.03E−01	4.81E−02	3.75E−02	5.48E−02	**2.37E−02**
f_{10}	1.46E−01	**6.84E−02**	6.11E−01	3.01E−01	1.41E+00	7.14E−02
f_{11}	4.09E−14	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	9.63E−03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{13}	7.97E−14	2.22E−16	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{14}	−12533.8	**−12569.5**	**−12569.5**	**−12569.5**	**−12569.5**	**−12569.5**
f_{15}	7.21E−05	2.32E−10	4.54E−11	3.26E−14	2.89E−14	**1.72E−14**
f_{16}	7.69E−16	5.53E−16	2.05E−27	**1.57E−32**	**1.57E−32**	**1.57E−32**
f_{17}	3.39E−15	5.47E−16	3.25E−26	**1.35E−32**	**1.35E−32**	**1.35E−32**
f_{18}	1.46E−06	4.55E−07	4.31E−13	**1.09E−21**	6.11E−16	9.43E−16
f_{19}	2.16E−13	9.69E−16	4.73E−26	**1.35E−31**	**1.35E−31**	**1.35E−31**
f_{20}	2.58E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{21}	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**
f_{22}	−2.93E+01	−2.95E+01	**−2.96E+01**	**−2.96E+01**	**−2.96E+01**	**−2.96E+01**
w/t/l	20/2/0	15/6/1	13/9/0	9/12/1	8/12/2	–

- Bare bones ABC (BABC) [13].
- Our approach NSABC.

For the above six ABC algorithms, they are run on 22 problems with $D = 30$ and 100. The stopping criterion $MaxFEs = 5000 \cdot D$ is used in all ABCs [9,43]. For ABC and NSABC, SN and $limit$ are equal to 50 and 100, respectively. The parameter C and the neighborhood radius k used in NSABC are set to 1.5 and 10, respectively. Other parameters in GABC, MABC, MEABC, and BABC are set according to the corresponding literature [13,16,43,49]. For each test problem, all ABC algorithms are run 30 trials.

Table 4 presents the mean best objective function values of six ABCs on problems with $D = 30$. In the last row, $w/t/l$ gives the overall comparison results between NSABC and other algorithms, and the meanings of the symbols w, t, and l are described in the following. NSABC outperforms the compared algorithm on w problems. NSABC and the compared algorithm obtain the same results on t problems. The compare algorithm is better than NSABC on l problems. From the results, NSABC surpasses the original ABC on all test problems except for f_7 and f_{21}. On these two problems, all six ABCs can converge to the global optima. GABC is slightly better than NSABC on f_{10}. The performance of NSABC and GABC are similar on six problems. For the rest of 15 problems, NSABC outperforms GABC. NSABC achieve better solutions than MABC on 13 problems, and they gain the same results on the rest of nine problems. NSABC gains more accurate solutions than MEABC on nine problems, while MEABC outperforms NSABC on only one problem f_{18}. For the rest of 12 problems, the performance of NSABC, MEABC, and BABC is similar. For problems f_4 and f_{18}, BABC is better than NSABC, but NSABC outperforms BABC on eight problems. Among all 22 problems, NSABC can find optimal or sub-optimal solutions on 18 problems. For problems f_6 and f_{22}, all six ABCs are trapped, and they only obtain promising solutions on f_9 and f_{10}.

Table 5 lists the results of six ABCs on $D = 100$. When the dimensional size increases from 30 to 100, the overall performance summarized by $w/t/l$ is not seriously affected. NSABC obtains better solutions than ABC on all problems except for f_{21}. With increasing of dimension, ABC is trapped on f_7. Both NSABC and GABC obtain the same results on three problems, while NSABC is better than GABC on the rest of 19 problems. For f_{11}, f_{12}, and f_{14}, GABC no longer converges to the global minimum on $D = 100$. NSABC is worse than MABC on f_{22}, while NSABC outperforms MABC on 13 problems. Compared to $D = 30$, MEABC surpasses NSABC on f_4 and f_{22}. BABC still performs better than NSABC on two problems, but NSABC surpasses BABC on nine problems.

To compare the overall performance of all six ABC variants, Friedman and Wilcoxon tests are used [43]. Based on the Friedman test, the mean ranking value of each algorithm is calculated. A smaller mean ranking value represents that the corresponding algorithm obtains a better overall performance. Table 6 gives the mean ranking values of all six ABCs. As shown, NSABC achieves the smallest ranking values for both $D = 30$ and 100. It demonstrates NSABC obtains the best overall performance among six ABCs. BABC and MEABC have the same performance for $D = 30$, while BABC is slightly worse than MEABC for $D = 100$. By using the Wilcoxon test, the significance between NSABC and its competitor is identified. Table 7 lists the p-values between NSABC and other algorithms. NSABC significantly outperforms BABC, MEABC, MEABC, GABC, and ABC. Compared with MEABC, NSABC achieves significantly better results on $D = 30$. For $D = 100$, NSABC is not significantly better than MEABC.

Table 5
Comparison results of six ABC algorithms when $D = 100$.

Problem	ABC Mean	GABC Mean	MABC Mean	MEABC Mean	BABC Mean	NSABC Mean
f_1	8.15E−15	3.24E−15	1.63E−22	2.52E−36	4.04E−50	**2.69E−80**
f_2	2.75E−06	3.07E−15	4.48E−19	4.48E−33	1.21E−51	**3.78E−75**
f_3	5.24E−15	3.23E−15	5.86E−23	7.84E−37	2.88E−50	**3.72E−77**
f_4	1.22E−08	4.75E−11	2.33E−52	4.03E−87	**5.79E−129**	2.84E−72
f_5	1.29E−09	6.82E−15	3.62E−12	2.44E−19	8.59E−28	**6.06E−40**
f_6	8.82E+01	8.55E+01	5.16E+01	3.98E+01	8.56E+01	**1.16E+01**
f_7	1.28E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_8	5.14E−40	6.33E−45	**7.12E−218**	**7.12E−218**	**7.12E−218**	**7.12E−218**
f_9	1.51E+00	7.37E−01	2.20E−01	1.83E−01	7.41E−01	**9.31E−02**
f_{10}	1.74E−01	3.18E−01	4.65E+00	1.13E+00	4.14E+01	**1.62E−01**
f_{11}	1.43E−11	7.11E−15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{12}	9.75E+00	1.93E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{13}	2.48E−14	3.18E−15	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{14}	−41164.8	−41856.9	**−41898.3**	**−41898.3**	**−41898.3**	**−41898.3**
f_{15}	5.24E−05	2.75E−07	5.27E−07	3.16E−13	1.57E−13	**4.20E−14**
f_{16}	7.57E−15	3.23E−15	1.76E−24	**4.71E−33**	**4.71E−33**	**4.71E−33**
f_{17}	1.42E−13	3.31E−15	6.86E−23	**1.35E−32**	**1.35E−32**	**1.35E−32**
f_{18}	7.65E−04	2.96E−05	1.52E−10	**2.04E−16**	1.80E−15	1.18E−15
f_{19}	9.67E−12	1.22E−14	1.43E−23	**1.35E−31**	**1.35E−31**	**1.35E−31**
f_{20}	2.92E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{21}	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**	**−7.83E+01**
f_{22}	−9.72E+01	−9.82E+01	−9.94E+01	**−9.95E+01**	**−9.95E+01**	−9.92E+01
w/t/l	21/1/0	18/4/0	13/8/1	8/11/3	9/11/2	–

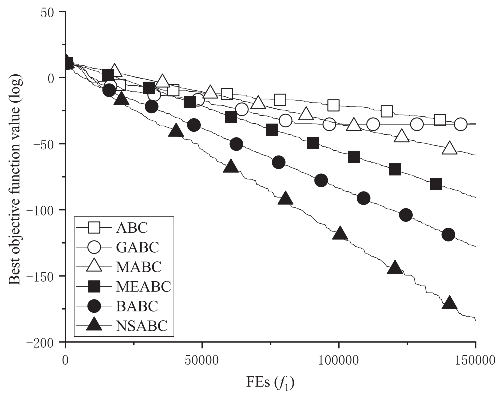
Table 6
Mean ranking values of six ABC algorithms. The best rank for each dimensional size is shown in bold.

Algorithm	$D = 30$ Mean ranking	$D = 100$ Mean ranking
ABC	5.64	5.70
GABC	4.23	4.48
MABC	3.52	3.55
MEABC	2.75	2.52
BABC	2.75	2.70
NSABC	**2.11**	**2.05**

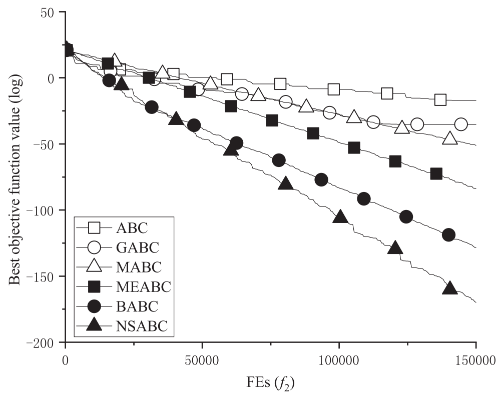
Table 7
The p-value achieved by Wilcoxon test between NSABC and five other ABCs. The p-values below the 0.05 significance level are shown in bold.

NSABC vs.	$D = 30$ p-value	$D = 100$ p-value
ABC	**8.86E-05**	**5.96E-05**
GABC	**4.46E-03**	**1.32E-04**
MABC	**1.47E-03**	**1.10E-02**
MEABC	**2.84E-02**	1.31E-01
BABC	**3.67E-02**	**3.29E-02**

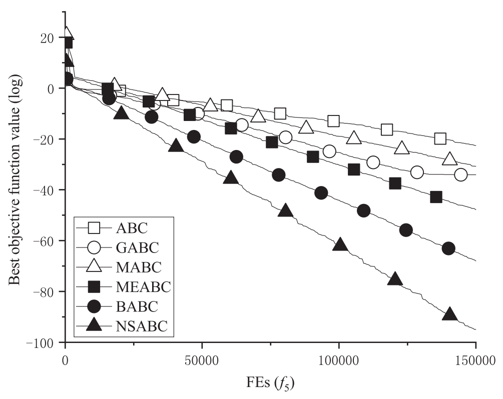
Fig. 5 gives the convergence curves of six ABCs on six problems (f_1, f_2, f_5, f_6, f_9, and f_{10}) when $D = 30$. NSABC shows the fastest convergence speed among six ABCs on f_1, f_2, f_5 and f_6. For f_9, BABC faster than other algorithms at the early search stage. With the increase of iterations, NSABC converges faster than other ABC algorithms at last two search stages. For f_{10}, NSABC shows faster convergence than other ABCs at the early and middle search stages, while GABC shows the fastest convergence at the last search stage. Fig. 6 presents the convergence curves of six ABCs on four problems ($f_{15} - f_{18}$) when $D = 30$. For problem $f_{15} - f_{17}$, NSABC is the fastest algorithm among six ABCs, and BABC achieves the second place. Though MEABC is slower than BABC and NSABC on f_{18} at the early and middle search stages, it shows the fastest convergence at the last search stage.



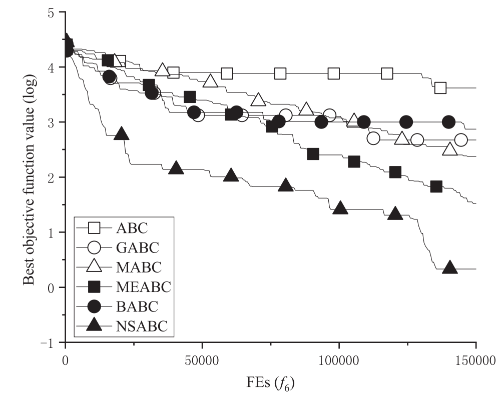
(a) Sphere (f_1)



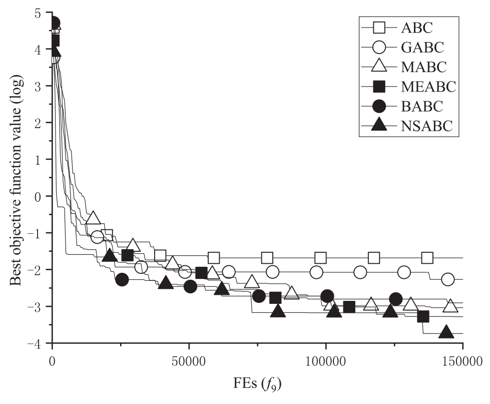
(b) Elliptic (f_2)



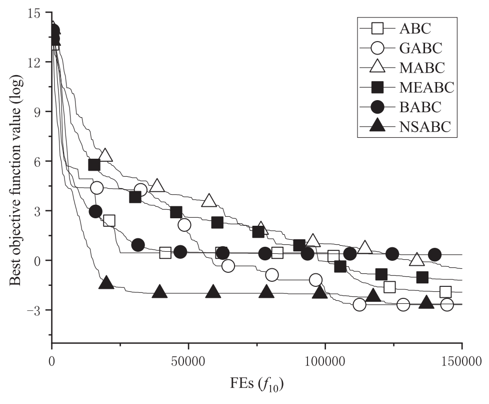
(c) Schwefel 2.22 (f_5)



(d) Schwefel 2.21 (f_6)

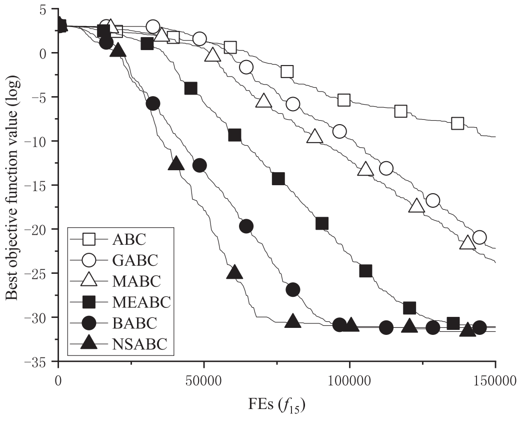


(e) Quartic (f_9)

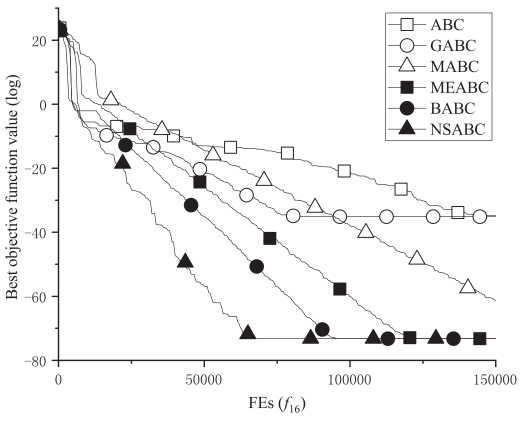


(f) Rosenbrock (f_{10})

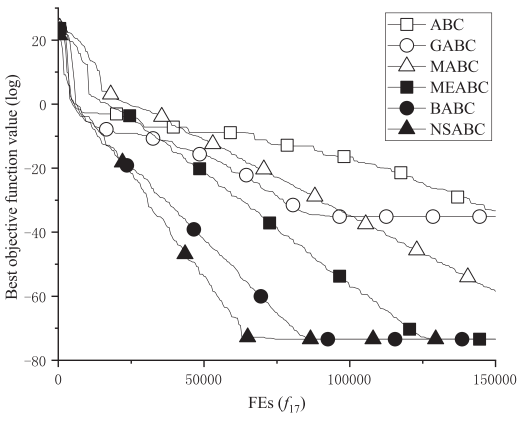
Fig. 5. Convergence curves of six ABC algorithms on six problems (f_1, f_2, f_5, f_6, f_9, and f_{10}) when $D = 30$.



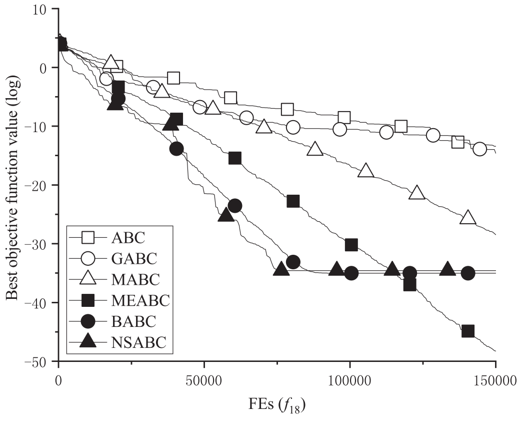
(a) Ackley (f_{15})



(b) Penalized 1 (f_{16})



(c) Penalized 2 (f_{17})



(d) Alpine (f_{18})

Fig. 6. Convergence curves of six ABC algorithms on four problems ($f_{15} - f_{18}$) when $D = 30$.

6. Conclusion

To overcome the drawbacks of the original ABC, this paper presents a novel ABC based on neighborhood selection (NSABC). The new approach employed three strategies. Firstly, a new selection mechanism based on the concept of k-neighborhood is proposed. The best solution X_{ib} is selected from the k-neighborhood of X_i, and an onlooker bee generates offspring based on X_{ib}. Compared with the original ABC, NSABC uses the objective function value to choose better solutions, and it does not need to calculate the selection probability. Secondly, two new search strategies are constructed based on the k-neighborhood. Finally, the scout bee search phase is improved by using the k-neighborhood and opposition-based learning. To evaluate the performance of NSABC, 22 benchmark problems with $D = 30$ and 100 are tested. Performance of NSABC is compared with five other ABC variants.

The parameter k defines the neighborhood radius, and it affects the quality of the local best solution X_{ib}. Results show that NSABC with different neighborhood radius k can obtain much better solutions than the original ABC. $k = 10$ is the relatively best setting among five k values.

Results show that NSABC outperforms ABC, GABC, MABC, MEABC, and BABC on most test problems. For $D = 30$ and 100, the performance of NSABC is not seriously affected. From the convergence characteristics, NSABC converges faster than five other algorithms on most problems.

The concept of *k*-neighborhood successfully helps ABC achieve better performance, and it may be applied to other ABC variants. The neighborhood radius *k* is fixed to 10 in the experiments. By dynamically updating the *k*, the performance of NSABC may be further improved. These will be possible research directions in the future study.

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.

CRediT authorship contribution statement

Hui Wang: Conceptualization, Methodology, Writing - original draft, Writing - review & editing. **Wenjun Wang:** Writing - review & editing. **Songyi Xiao:** Software, Data curation. **Zhihua Cui:** Writing - review & editing, Formal analysis. **Minyang Xu:** Software, Validation. **Xinyu Zhou:** Writing - review & editing.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. 61663028, 61966019, and 61603163).

References

[1] B. Akay, D. Karaboga, A modified artificial bee colony algorithm for real-parameter optimization, Inf. Sci. 192 (2012) 120–142.
[2] E. Amiri, M.N. Dehkordi, Dynamic data clustering by combining improved discrete artificial bee colony algorithm with fuzzy logic, Int. J. Bio-Inspir. Comput. 12 (3) (2018) 164–172.
[3] S. Asghari, N.J. Navimipour, Cloud service composition using an inverted ant colony optimisation algorithm, Int. J. Bio.-Inspir. Comput. 13 (4) (2019) 257–268.
[4] D. Bajer, B. Zorić, An effective refined artificial bee colony algorithm for numerical optimisation, Inf. Sci. 504 (2019) 221–275.
[5] A. Banharnsakun, T. Achalakul, B. Sirinaovakul, The best-so-far selection in artificial bee colony algorithm, Appl. Soft. Comput. 11 (2) (2011) 2888–2901.
[6] A. Banitalebi, M.I.A. Aziz, A. Bahar, Z.A. Aziz, Enhanced compact artificial bee colony, Inf. Sci. 298 (2015) 491–511.
[7] Z.H. Cui, Y. Cao, X.J. Cai, J.H. Cai, J.J. Chen, Optimal LEACH protocol with modified bat algorithm for big data sensing systems in internet of things, J. Parallel. Distrib. Comput. 132 (2019) 217–229.
[8] Z.H. Cui, Y. Chang, J.J. Zhang, X.J. Cai, W.S. Zhang, Improved NSGA-III with selection-and-elimination operator, Swarm. Evol. Comput. 49 (2019) 23–33.
[9] L.Z. Cui, G.H. Li, Q.Z. Lin, Z.H. Du, W.F. Gao, J.Y. 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.
[10] L.Z. Cui, G.H. Li, X.Z. Wang, Q.Z. Lin, J.Y. Chen, N. Lu, J. Lu, A ranking-based adaptive artificial bee colony algorithm for global numerical optimization, Inf. Sci. 417 (2017) 169–185.
[11] L.Z. Cui, G.H. Li, Z.X. Zhu, Q.Z. Lin, Z.K. Wen, N. Lu, K.C. Wong, J.Y. Chen, A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization, Inf. Sci. 414 (2017) 53–67.
[12] S. Das, A. Abraham, U. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (3) (2009) 526–553.
[13] W.F. Gao, F.T.S. Chan, L.L. Huang, S.Y. Liu, Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood, Inf. Sci. 316 (2015) 180–200.
[14] W.F. Gao, L.L. Huang, S.Y. Liu, F.T.S. Chan, C. Dai, X. Shan, Artificial bee colony algorithm with multiple search strategies, Appl. Math. Comput. 271 (2015) 269–287.
[15] W.F. Gao, L.L. Huang, J. Wang, S.Y. Liu, C.D. Qin, Enhanced artificial bee colony algorithm through differential evolution, Appl. Soft Comput. 48 (2016) 137–150.
[16] W.F. Gao, S.Y. Liu, A modified artificial bee colony algorithm, Comput. Oper. Res. 39 (2012) 687–697.
[17] W.F. Gao, Z. Wei, Y. Luo, J. Cao, Artificial bee colony algorithm based on parzen window method, Appl. Soft. Comput. 74 (2019) 679–692.
[18] S. Garg, K. Kaur, S. Batra, G.S. Aujla, G. Morgan, N. Kumar, A.Y. Zomaya, R. Ranjan, En-ABC: an ensemble artificial bee colony based anomaly detection scheme for cloud environment, J. Parallel Distrib. Comput. 135 (2020) 219–233.
[19] B. Gorkemli, D. Karaboga, A quick semantic artificial bee colony programming (qsABCP) for symbolic regression, Inf. Sci. 502 (2019) 346–362.
[20] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Erciyes University, Engineering Faculty, Computer engineering Department, 2005 Technical report-tr06.
[21] 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.
[22] K.S. Li, Y. Chen, W. Li, J. He, Y. Xue, Improved gene expression programming to solve the inverse problem for ordinary differential equations, Swarm. Evol. Comput. 38 (2018) 231–239.
[23] K.S. Li, H. Wang, A mobile node localization algorithm based on an overlapping self-adjustment mechanism, Inf. Sci. 481 (2019) 635–649.
[24] X. Li, G. Yang, Artificial bee colony algorithm with memory, Appl. Soft Comput. 41 (2016) 362–372.
[25] Z.P. Liang, K.F. Hu, Q.X. Zhu, Z.X. Zhu, An enhanced artificial bee colony algorithm with adaptive differential operators, Appl. Soft Comput. 58 (2017) 480–494.
[26] R. Mohammadi, R. Javidan, M. Keshtgari, An intelligent traffic engineering method for video surveillance systems over software defined networks using ant colony optimisation, Int. J. Bio-Inspired. Comput. 12 (3) (2018) 173–185.
[27] C. Ozturk, E. Hancer, D. Karaboga, A novel binary artificial bee colony algorithm based on genetic operators, Inf. Sci. 297 (2015) 154–170.
[28] J.S. Pan, P. Hu, S.C. Chu, Novel parallel heterogeneous meta-heuristic and its communication strategies for the prediction of wind power, Processes 7 (11) (2019) 845.
[29] H. Peng, C. Deng, Z. Wu, Best neighbor guided artificial bee colony algorithm for continuous optimization problems, Soft Comput. 23 (18) (2019) 8723–8740.
[30] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution, IEEE Trans. Evol. Comput. 12 (2008) 64–79.
[31] A. Saad, S.A. Khan, A. Mahmood, A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design, Swarm Evol. Comput. 38 (2018) 187–201.
[32] X. Song, M. Zhao, Q. Yan, S. Xing, A high-efficiency adaptive artificial bee colony algorithm using two strategies for continuous optimization, Swarm. Evol. Comput. 50 (2019) 100549.
[33] H.S. Tsai, Artificial bee colony directive for continuous optimization, Appl. Soft Comput. 87 (2020) 105982, doi:10.1016/j.asoc.2019.105982.

[34] C.L. Tseng, C.S. Cheng, C.W. Chen, S.Y. Wang, R.J. Wen, A Localization Method Using the Bee Colony Algorithm for Mobile Wireless Sensor Networks, in: Proceedings of International Automatic Control Conference (CACS), 2016, pp. 194–199.
[35] F. Wang, Y.X. 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.
[36] F. Wang, Y.X. Li, A.M. Zhou, K. Tang, An estimation of distribution algorithm for mixed-variable newsvendor problems, IEEE Transactions on Evolutionary Computation (2019), doi:10.1109/TEVC.2019.2932624. To be published.
[37] F. Wang, H. Zhang, K.S. Li, Z.Y. Lin, J. Yang, X.L. Shen, A hybrid particle swarm optimization algorithm using adaptive learning strategy, Inf. Sci. 436/437 (2018) 162–177.
[38] F. Wang, H. Zhang, Y.X. Li, Y.Y. Zhao, Q. Rao, External archive matching strategy for MOEA/d, Soft. Comput. 22 (23) (2018) 7833–7846.
[39] H. Wang, S. Rahnamayan, H. Sun, M.G.H. Omran, Gaussian bare-bones differential evolution, IEEE Trans. Cybern. 43 (2) (2013) 634–647.
[40] 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.
[41] H. Wang, W.J. Wang, Z.H. Cui, X.Y. Zhou, J. Zhao, Y. Li, A new dynamic firefly algorithm for demand estimation of water resources, Inf. Sci. 438 (2018) 95–106.
[42] H. Wang, W.J. Wang, H. Sun, S. Rahnamayan, Firefly algorithm with random attraction, Int. J. Bio-Inspir. Comput. 8 (1) (2016) 33–41.
[43] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J. Pan, Multi-strategy ensemble artificial bee colony algorithm, Inf. Sci. 279 (2014) 587–603.
[44] S.Y. Xiao, W.J. Wang, H. Wang, D.K. Tan, Y. Wang, X. Yu, R.X. Wu, An improved artificial bee colony algorithm based on elite strategy and dimension learning, Mathematics 7 (3) (2019) 289.
[45] 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.
[46] Y. Xue, B. Xue, M. Zhang, Self-adaptive particle swarm optimization for large-scale feature selection in classification, ACM Trans. Knowl. Discov. Data 13 (5) (2019) 1–28.
[47] M. Zhang, H. Wang, Z.H. Cui, J.J. Chen, Hybrid multi-objective cuckoo search with dynamical local search, Memetic Comput. 10 (2) (2018) 199–208.
[48] X.Y. Zhou, Z.J. Wu, H. Wang, S. Rahnamayan, Gaussian bare-bones artificial bee colony algorithm, Soft Comput. 20 (3) (2016) 907–924.
[49] G. Zhu, S. Kwong, Gbest-guided artificial bee colony algorithm for numerical function optimization, Appl. Math. Comput. 217 (2010) 3166–3173.