Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Improving artificial bee colony algorithm using modified nearest neighbor sequence

Kai Li [a], Hui Wang [a,*], Wenjun Wang [b], Feng Wang [c], Zhihua Cui [d]

[a] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China
[b] School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China
[c] School of Computer Science, Wuhan University, Wuhan 430072, China
[d] School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China

ARTICLE INFO

Article history:
Received 23 July 2021
Revised 26 October 2021
Accepted 27 October 2021
Available online xxxx

Keywords:
Artificial bee colony
Swarm intelligence
Nature inspired algorithm
Multi-strategy
Probability selection

ABSTRACT

Nearest neighbor (NN) is a simple machine learning algorithm, which is often used in classification problems. In this paper, a concept of modified nearest neighbor (MNN) is proposed to strengthen the optimization capability of artificial bee colony (ABC) algorithm. The new approach is called ABC based on modified nearest neighbor sequence (NNSABC). Firstly, MNN is used to construct solution sequences. Unlike the original roulette selection, NNSABC randomly chooses a solution from the corresponding nearest neighbor sequence to generate offspring. Then, two novel search strategies based on the modified nearest neighbor sequence are employed to build a strategy pool. In the optimization process, different search strategies are dynamically chosen from the strategy pool according to the current search status. In order to study the optimization capability of NNSABC, two benchmark sets including 22 classical problems and 28 CEC 2013 complex problems are tested. Experimental results show NNSABC obtains competitive performance when compared with twenty-three other ABCs and evolutionary algorithms.

1. Introduction

In recent years, various evolutionary algorithms (EAs) have been proposed to deal with optimization problems in real world (Liu et al., 2020; Wang et al., 2020; Cai et al., 2020; Cui et al., 2019). There are some representative EAs, such as particle swarm optimization (PSO) (Wang et al., 2021), differential evolution (DE) (Wang et al., 2013), genetic algorithm (GA) (Zhang et al., 2020), artificial bee colony (ABC) (Karaboga, 2005), ant colony optimization (ACO) (Asghari and Navimipour, 2019), and estimation of distribution algorithm (EDA) (Wang et al., 2020). Among these algorithms, ABC has attracted great attention because of its excellent optimization performance, simple concept, and easy implementation.

There are two types of optimization problems including continuous optimization problems and discrete optimization problems. From the computational complexity point of view, continuous optimization problems can be considered easy to solve, but they are still challenging. For example, the interactions between decision variables and the growth of the number of decision variables can significantly increase the complexity and difficulty of problems. Especially for numeric function optimization, the characteristics of functions greatly affect the difficulty of functions. Although ABC has shown good performance on many continuous optimization problems, it has some shortcomings because of its intrinsic randomness. The original ABC is very efficient for basic multimodal problems, while it easily suffers from premature convergence on some complex problems (composite and non-separable) (Akay and Karaboga, 2012). Relying on the behavior of swarm bees, an employed or onlooker bee could alter one feature (dimension) only during exploitation. However, changing only one dimension of the parent solution results in a slow convergence rate. Then, the exploitation capability is weakened. In Banharnsakun et al. (2011), Banharnsakun et al. reported a strange phenomenon. When the function values are very small but larger than the zero, their corresponding fitness values are the same (rounded up to be 1.0). A further analysis shows that the roulette selection in the onlooker bee phase does not work because all solutions have the same selection probability.

* Corresponding author.
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2021.10.012
1319-1578/

Please cite this article as: K. Li, H. Wang, W. Wang et al., Improving artificial bee colony algorithm using modified nearest neighbor sequence, Journal of King Saud University – Computer and Information Sciences, https://doi.org/10.1016/j.jksuci.2021.10.012

In order to overcome the above issues, an improved ABC algorithm based on modified nearest neighbor sequence (called NNSABC) is proposed in this paper. The main contributions of this work can be summarized as follows.

- This paper aims to enhance the optimization capability of ABC by using a simple machine learning technique (nearest neighbor).
- A modified nearest neighbor (MNN) is proposed to construct solution sequences.
- Two effective search strategies based on the modified nearest neighbor sequence are designed to establish a strategy pool. During the search process, an appropriate search strategy is selected from the strategy pool in terms of the quality of offspring.
- The roulette selection used in the original ABC may not work at the last search stages. In NNSABC, a new selection mechanism is proposed based on the modified nearest neighbor sequence.

The remaining part of the paper is arranged as below. The original ABC and its recent advances are briefly introduced in Section 2. In Section 3, ABC based on modified nearest neighbor sequence (NNSABC) is proposed. Experimental results and discussions are given in Section 4. Finally, the work is summarized in Section 5.

2. Background review and related work

2.1. Artificial bee colony

ABC is a popular optimization method proposed by Karaboga (2005). It is inspired by the cooperation of swarm bees. Bees in the swarm are divided into three categories: employed bees, onlooker bees and scout bees. The task of the swarm bees is to search for profitable food sources. Each food source represents a potential solution in the search space. The employed bees try to find better food sources by searching around the current food sources. They share the quality information of the food sources to the onlooker bees. Then, the onlooker bees choose good food sources found to further search for better food sources. When a food source is abandoned, the related employed bee will become a scout bee and find a new random food source.

Initialization – There are SN initial food sources (solutions) in the swarm, where SN stands for the number of food sources and it is equal to the number of employed or onlooker bees. Each food source $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$ is randomly generated as below.

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

where $i = 1,2,\ldots,SN, j = 1,2,\ldots,D, D$ is the problem dimension, $rand(0,1) \in [0,1]$ is a random value, and $[x_{max,j}, x_{min,j}]$ is the interval constraint.

Employed bee search phase – They search around each food source X_i to generate a new one $V_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,D})$. The search strategy is described below.

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

where $i = 1,2,\ldots,SN, j$ is a random index between 1 and D, k is a random integer between 1 and SN ($i \neq k$), and $\phi_{i,j} \in [-1,1]$ is a random number within $[-1,1]$. If V_i is better than X_i, X_i will be replaced by V_i (Karaboga, 2005).

Onlooker bee search phase – They use a probability selection method to determine which solution is chosen for further search. The probability p_i of a food source X_i is computed by

$$p_i = \frac{fit(i)}{\sum_{i=1}^{SN} fit(i)}, \tag{3}$$

where $fit(i)$ is the fitness value of the i-th food source. The fitness value $fit(i)$ is described as below.

$$fit(i) = \begin{cases} \frac{1}{1+f_i}, & \text{if} f_i \geqslant 0 \\ 1 + abs(f_i), & \text{if} f_i < 0 \end{cases}, \tag{4}$$

where f_i is the objective function value of the i-th food source.

When the probability p_i is satisfied, the corresponding food source X_i will be selected. Like the employed bees, the onlooker bees use the same strategy Eq. (2) to produce a new food source V_i by searching around X_i. The greedy selection is also used to compare the quality of V_i and X_i, and the better one between them is used as the new X_i (Karaboga, 2005).

Scout bee search phase – In ABC, when the current X_i still cannot be improved after a preset number ($limit$) of greedy selections, the solution will be abandoned. Then, a new solution is randomly generated on the basis of Eq. (1) and it is used to replace the abandoned one.

2.2. Recent work on ABC

Since the strong search ability and simple concept, many scholars have paid attention to the research of ABC. Different variants of ABC were proposed to deal with practical and benchmark optimization problems. In this section, recent work on ABC is briefly introduced.

To speed up the search and improve the exploitation, the search equation/strategy is usually combined with the global best solution X_{best} (Banharnsakun et al., 2011; Peng et al., 2019; Zhu and Kwong, 2010). Banharnsakun et al. (2011) modified the search strategy based on X_{best}, and proposed the best-so-far ABC. Peng et al. (2019) used the best neighbor to guide the search and designed a new search strategy. In Zhu and Kwong (2010), the difference $(X_{best} - X_i)$ is added to the search equation of the original ABC. Inspired by DE, a new ABC variant (called ABC/best/1) was proposed in Gao and Liu (2011).

It is known that ABC is not good at exploitation. For the sake of improving this case, some researchers proposed various improved strategies. In Cui et al. (2016), a depth-first search framework was proposed to make better solutions have more computing resources. The elite solution set is used to improve the search strategy. Cui et al. (2017) presented a dynamical approach to adjust the swarm size. If the current search strategy does not achieve good performance, the swarm size will be expanded to improve the exploration; otherwise the swarm size will be reduced to enhance the exploitation. Wang et al. (2020) proposed an improved ABC based on neighborhood selection (NSABC), in which the best neighbor of each solution is chosen to participate in the search of onlooker bees. In Karaboga and Gorkemli (2014), Karaboga et al. pointed out that the onlooker bees choose their food sources in a different way from the employed bees, and the onlookers' forging behavior should be modeled by a different search equation from that of the employed bees. In Zhou et al. (2021), three kinds of neighborhood topologies are used for different solutions to perform diverse abilities of disseminating search information, contributing to a good balance between exploration and exploitation. Therefore, a new search equation is proposed for the onlooker bees, which exploits the best solution among neighbors. Wang and Yi (2017) presented a hybrid approach based on ABC and krill herd (KH). ABC and KH can share the global best solutions through an information exchange strategy in the search process.

Most ABCs use a single search strategy (like Eq. (2))) to produce offspring. To strengthen the search ability, many ABC variants use two or more search strategies. Wang et al. (2014) used three types of search strategies in ABC (MEABC). In the swarm, every solution has an independent search strategy, which is dynamically updated in each iteration. In Gao et al. (2015), the probability of different strategies being selected is based on historical search experience.

In Kiran et al. (2015), five search strategies are used in ABC. In the beginning, each strategy has an independent probability of being selected. The selection probability of each strategy will be dynamically updated at the end of each iteration.

Recently, ABC was applied to various optimization problems. To deal with constrained optimization problems, Liu et al. (2018) proposed a new ABC based on dynamic penalty and lévy flight, in which the dynamic penalty method is used to handle the constraints. Yeh and Hsieh (2011) designed a constrained ABC to solve mixed-integer reliability problems. Results show the effectiveness of the new ABC. Manoj and Elias (2012) used an improved ABC for filter design. In Yildiz (2013), a hybrid ABC was proposed to solve the optimization problem in the manufacturing industry. A general multi-objective optimization problems has two or three objectives (Wang et al., 2018; Wang et al., 2019). Saad et al. (2018) used a multi-objective ABC to optimize the network topology.

3. ABC based on modified nearest neighbor sequence (NNSABC)

In this work, an improved ABC based on modified nearest neighbor sequence (called NNSABC) is proposed. In NNSABC, there are four main modifications: 1) a concept of modified nearest neighbor (MNN) is defined; 2) the modified nearest neighbor is used to construct solution sequences; 3) a strategy pool consists of two improved search strategies based on the modified nearest neighbor sequence; and 4) a novel selection strategy based on the modified nearest neighbor sequence is designed. The detailed of those modifications are described as follows.

3.1. A concept of modified nearest neighbor (MNN)

In the search process, checking the neighborhoods of solutions is important for the current search (Wang et al., 2020). Because the neighborhood of a solution may cover more accurate solutions. In some cases, it can gain the global optimum by searching the neighborhoods of suboptimal solutions (Wang et al., 2013). Therefore, neighborhood search is an efficient operation for improving the search capabilities of many intelligent optimization algorithms. In the past decades, various neighborhood search strategies have been proposed. In Lin et al. (2017), Lin et al. used a star-shaped neighborhood structure to strengthen the search of employed bees. Banharnsakun et al. (2011) adjusted the neighborhood range to enhance the global search of scout bees. In Das et al. (2009), some efficient neighborhood search strategies were constructed based on ring topology. Inspired by the literature (Wang et al., 2020), Xiao et al. (2021) designed an adaptive method to dynamically change the neighborhood size.

In our recent work, the ring topology was used in ABC to construct new search strategies. However, the ring topology is a hypothetical structure. The adjacent solutions are determined by their indices. For example, X_2 is adjacent to X_3 and X_1. In the real search space, they may be far way from each other and they are no longer neighbors. Therefore, it may be meaningful to use the distance to find true neighbors in the search space.

For the current solution X_i, what kinds of neighbors does it need? First, the neighbors are truly near to X_i. Second, the neighbors have better fitness values than the current X_i. Based on these near and better neighbors, we may conduct some successful neighborhood searches with high probabilities. Based on the above analysis, a concept of modified nearest neighbor (MNN) is proposed in this work.

Definition 1 (*Modified Nearest Neighbor (MNN)*). For the current solution X_i in the population $P(t)$, its modified nearest neighbor X_i^* must satisfy the following two conditions (this work only considers minimization problems):

(1) Establish a set $S_i = \{X | X \in P(t) \bigwedge f(X) < f(X_i)\}$;
(2) If $S_i \neq \varnothing$, then $\exists X_i^* \in S_i, \forall X \in S_i, dist(X_i^*, X_i) \leqslant dist(X, X_i)$.

Lemma 1. *If X_i does not have a modified nearest neighbor, X_i is the best solution in the population.*

Proof 1. If X_i does not have a modified nearest neighbor, the set S_i is $\varnothing$. It demonstrates that the rest of solutions in the population are worse than X_i. Therefore, X_i is the best solution in the population. □

Fig. 1 presents an example of finding a modified nearest neighbor of X_3. As shown in Fig. 1(a), there are 10 solutions in the swarm and X_3 is the current solution. In Fig. 1(b), some better solutions than X_3 are chosen to construct the set $S_3 = \{X_1, X_2, X_7, X_{10}\}$. Solutions in S_3 are surround by dotted lines. All solution in S_3 are better than X_3. In Fig. 1(c), the Euclidean distance between X_3 and each solution in S_3 is calculated. Then, we can get the modified nearest neighbor X_3^* from the S_3, and X_3^* is nearest to X_3. □

3.2. Modified nearest neighbor sequence (MNNS)

Based on the concept of MNN, we can easily obtain the modified nearest neighbor sequence (MNNS) of each solution in the population. Supposing X_i is the i-th solution in the current swarm $P(t)$. Let X_i^0 be equal to X_i, and the modified nearest neighbor of X_i^0 is X_i^1. Then, X_i^1 is set as the current solution, and its modified nearest neighbor X_i^2 is obtained. The above operation can be repeated until the corresponding modified nearest neighbor cannot be found. Thus, the modified nearest neighbor sequence of X_i is

$$MS_i = \{X_i^0, X_i^1, X_i^2, \ldots, X_i^{m_i}\}, \tag{5}$$

where $0 \leqslant m_i \leqslant SN - 1$, and X_i^0 is actually X_i. It is clear that the length of MS_i is $m_i + 1$.

It is noted that solutions in the modified nearest neighbor sequence are arranged in descending order by the objective function value. Then, we can get $f(X_i^0) > f(X_i^1) > f(X_i^2) > \cdots > f(X_i^{m_i})$. It can easily deduce that $X_i^{m_i}$ is the best solution in the population. For an extreme case $m_i = 0$, the modified nearest neighbor sequence of X_i has only one solution $MS_i = \{X_i^0\}$. It demonstrates that X_i (or X_i^0) is the best solution in the population.

Fig. 2 gives an example of constructing the modified nearest neighbor sequence of X_3. As seen, for the current X_3, its modified nearest neighbor X_3^1 (X_2) is firstly found. Then, X_3^1 is used as the new current solution and its modified nearest neighbor is X_3^2 (X_7). Next, the modified nearest neighbor of X_3^2 is X_3^3 (X_{10}). Finally, we cannot find the modified nearest neighbor of X_3^3 and the above repeated process is stopped. Thus, the modified nearest neighbor sequence of X_3 is $MS_3 = \{X_3, X_2, X_7, X_{10}\}$.

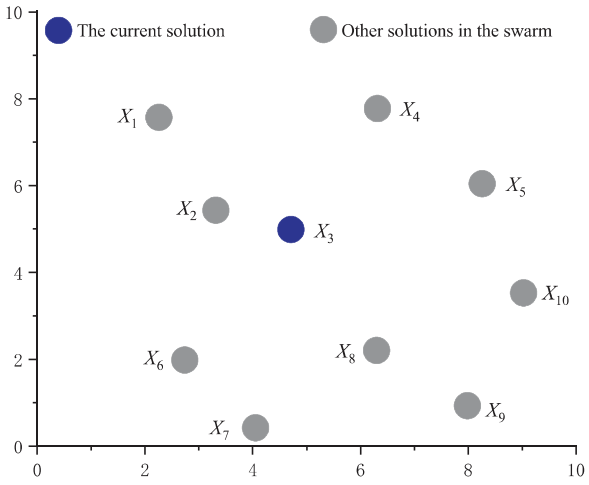
3.3. Improved search strategies based on MNNS

The search strategy in the original ABC shows good exploration, but the exploitation is weak. In Gao and Liu (2012), a new strategy (called ABC/best/1) motivated by the global best guided DE mutation operator (DE/best/1) was proposed to strengthen the exploitation. The ABC/best/1 is defined as below.

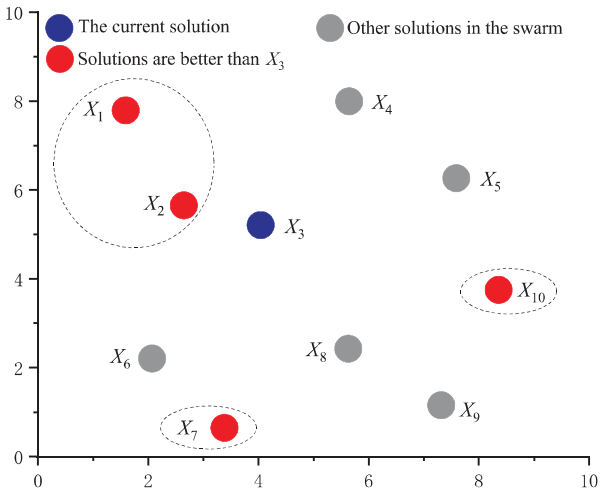
$$v_{i,j} = x_{best,j} + \phi_{i,j} \cdot \left(x_{r1,j} - x_{r2,j}\right), \tag{6}$$

where X_{r1} and X_{r2} are two different solutions randomly chosen from the swarm, and $i \neq r1 \neq r2$.

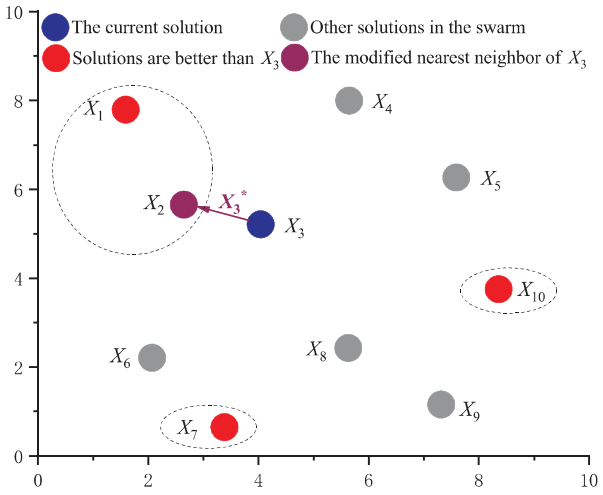
To further improve the exploitation, a modified ABC/best/1 was proposed as follows Wang et al. (2014).



(a) The current population $P(t)$.



(b) Construct the set $S_3 = \{X_1, X_2, X_7, X_{10}\}$.



(c) Find the modified nearest neighbor X_3^* from S_3.

Fig. 1. An clear implementation of the modified nearest neighbor.

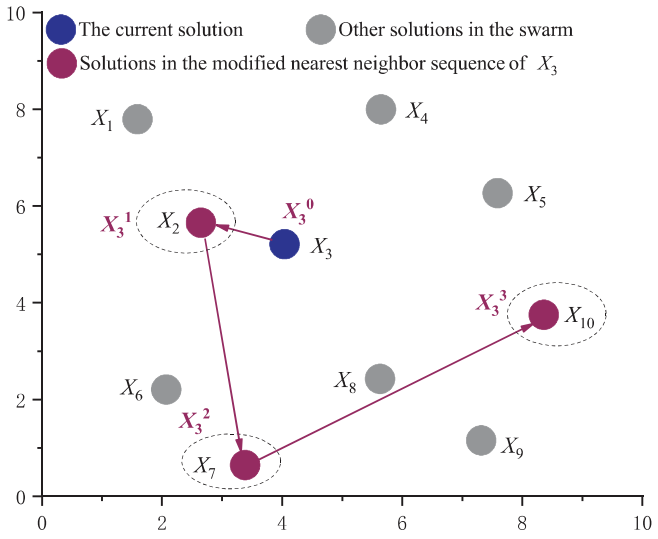


Fig. 2. An example of constructing the modified nearest neighbor sequence of X_3.

$$v_{i,j} = x_{best,j} + \phi_{i,j} \cdot \left(x_{best,j} - x_{k,j}\right). \tag{7}$$

As shown in Eq. (7), X_{best} is used as the base solution and it also participates in generating the step size. Our previous experimental studies showed that Eq. (7) could excessively improve the exploitation and accelerate the convergence, but it might result in premature convergence in solving multimodal optimization problems. To tackle the above issue, an improved search strategy based on MNNS and modified ABC/best/1 is designed as below.

$$v_{i,j} = mcx_{i,j} + \phi_{i,j} \cdot \left(x_{best,j} - x_{k,j}\right), \tag{8}$$

$$mcx_{i,j} = \frac{\sum_{h=0}^{m_i} x_{i,j}^h}{m_i + 1}, \tag{9}$$

where $i = 1, 2, \ldots, SN, j \in [1, D]$ is a random integer, $MCX_i = (mcx_{i,1}, mcx_{i,2}, \ldots, mcx_{i,D})$ is the mean center of the sequence MS_i, $mcx_{i,j}$ is the j-th dimension of MCX_i, and $X_i^h = (x_{i,1}^h, x_{i,2}^h, \ldots, x_{i,D}^h)$ is the h-th solution in MS_i. In Section 3.2, we analyzed an extreme case $m_i = 0$. Under this case, X_i is equal to X_{best}, and Eq. (8) is equal to the modified ABC/best/1 (Eq. (7)).

For any solution X_i^h in MS_i, it is easy to deduce that X_i^{h+1} is better than X_i^h ($h \in [1, m-1]$). Then, the difference $\Delta = \left(X_i^{h+1} - X_i^h\right)$ can be used a step size to guide the search. When X_i^h is added a step size Δ, it will move to a nearer and better position X_i^{h+1}. Inspired by this idea, another improved strategy is proposed as below.

$$v_{i,j} = x_{best,j} + \phi_{i,j} \cdot \left(x_{i,j}^{h+1} - x_{i,j}^h\right), \tag{10}$$

where $h \in [0, m_i - 1]$ is a random integer. To satisfy Eq. (10), $m_i \geqslant 1$ is needed. There is only one case $m_i = 0$ that cannot satisfy the above search strategy, because the set MS_i has only one solution X_{best}. For this case, X_{best} is chosen as X_i^h and another solution X_k randomly selected from $P(t)$ is used as X_i^{h+1}. Then, the new search equation is similar to the modified ABC/best/1 (Eq. (7)). Thus, we also use Eq. (7) to replace Eq. (10) for $m_i = 0$. The second new search strategy is given as below.

$$v_{i,j} = \begin{cases} x_{best,j} + \phi_{i,j} \cdot \left(x_{i,j}^{h+1} - x_{i,j}^{h}\right), & \text{if} m_i \geqslant 1 \\ x_{best,j} + \phi_{i,j} \cdot \left(x_{best,j} - x_{k,j}\right), & \text{if} m_i == 0 \end{cases}, \tag{11}$$

Strategy pool – A recent study reported that multiple search strategies are beneficial for improving the optimization performance (Wang et al., 2014). To implement the above idea, this paper establishes a strategy pool (SP) consisting of two improved search strategies: Eq. (8) and Eq. (11).

$$\text{SP} = \{\text{Eq.}(8), \text{Eq.}(11)\}. \tag{12}$$

Like Wang et al. (2014), each solution X_i is assigned a strategy SX_i. At the initial stage, the search strategy SX_i is randomly selected from SP. The strategy SX_i is dynamically updated at each iteration. If $f(V_i) > f(X_i)$, the strategy (SX_i) continues to be used; otherwise, another different strategy will be selected from SP to replace the original strategy (SX_i). The updating method for SX_i is defined as below.

$$SX_i = \begin{cases} SX_i, & \text{if} f(V_i) > f(X_i) \\ \text{SP} - \{SX_i\}, & \text{otherwise} \end{cases}, \tag{13}$$

where $\text{SP} - \{SX_i\}$ represents another strategy which is different from SX_i. If SX_i = Eq. (8), $\text{SP} - \{SX_i\}$ is equal to Eq. (11); If SX_i = Eq. (11), SP-$\{SX_i\}$ is equal to Eq. (8).

The framework of the employed bee phase is described in Algorithm 1. As seen, each offspring V_i is produced by its corresponding strategy SX_i. For lines 3 and 4, the function value is utilized to judge the quality of solutions. According to the quality of V_i, the corresponding SX_i is updated by Eq. (13).

Algorithm 1: Employed bee phase

```
1  for each X_i do
2      Generate V_i according to the search strategy SX_i;
3      Compute f(V_i), and FEs = FEs + 1;
4      if f(V_i) < f(X_i) then
5          X_i = V_i and set trial_i = 0;
6      else
7          trial_i = trial_i + 1;
8      end
9      Update the strategy SX_i according to Eq. (13);
10 end
```

3.4. Random selection method based on MNNS

In the original ABC, converting the objective function value into a fitness value according to Eq. (4) may cause some errors. The reason is that floating-point operations in computers require exponent matching. Then, $\frac{1}{1+f(X_i)}$ in Eq. (4) many ignore the smaller $f(X_i)$. For instance, when the first solution $f(X_1)$ is 10^{-20}, its fitness value is $\frac{1}{1+10^{-20}} = 1$. The second solution $f(X_2)$ is 10^{-30}, and its fitness value is $\frac{1}{1+10^{-30}} = 1$. Although their objective function values are quite different, they still achieve the same fitness value. Therefore, we cannot distinguish between X_1 and X_2 by using their fitness value. If we measure the quality of the solution according to the fitness value, we will not be able to obtain a more accurate solution, and the search will become slow. If we use the fitness value and Eq. (3) in the onlooker bee search phase, the probability of each solution is $1/SN$ at the last search stage. It makes the onlooker bee search phase ineffective (Cui et al., 2016).

To tackle the above issues, the objective function value is used to judge the solution quality (Banharnsakun et al., 2011). Moreover, a random selection method based on MNNS is utilized to replace the roulette selection in the original ABC. For each solution X_i, its modified nearest neighbor sequence (MNNS) MS_i is built by Eq. (5). Then, a solution $X_i^{r_i}$ is randomly chosen from the solution sequence MS_i, where r_i is an integer and it is defined by

$$r_i = \begin{cases} rnd(1, m_i), & \text{if} m_i \geqslant 1 \\ 0, & \text{if} m == 0 \end{cases}, \tag{14}$$

where $rnd(1, m_i)$ is an integer between 1 and m_i. As seen, there are two cases for r_i. For the first case ($m_i \geqslant 1$), each solution sequence MS_i has two solutions at least. As mentioned before, solutions in MS_i are arranged in descending order by the objective function value. Then, the randomly selected solution $X_i^{r_i}$ is superior to X_i. In the original ABC, the onlooker bees are responsible for choosing some better solutions for further search. Unlike the original roulette selection in ABC, our approach uses another method called random selection to achieve the same purpose. For the second case ($m_i == 0$), X_i^0 is equal to the best solution X_{best}, and $MS_i = \{X_{best}\}$. Then, X_{best} is chosen for further search.

Fig. 3 clearly illustrates the random selection method based on MNNS. As shown, there are two main steps. First, MS_i is obtained for each X_i. Second, a random solution $X_i^{r_i}$ is chosen from MS_i. It is easy to implement the above operations. Compared with the original roulette selection in ABC, the proposed random selection method overcomes its shortcomings.

The framework of the onlooker bee phase is presented in Algorithm 2. As seen, there are two steps for choosing $X_i^{r_i}$. Like the employed bee phase, offspring are produced by their own search strategies. For line 4, we find $X_i^{r_i}$ in the population and get its true index h. This is convenient to describe the following operations.

Algorithm 2: Onlooker bee phase

```
1  for i = 1 to SN do
2      Establish MS_i for X_i according to Eq. (5);
3      Randomly select a solution X_i^{r_i} from MS_i according to Eq. (14);
4      Find the true index of X_i^{r_i} in the population (X_h == X_i^{r_i});
5      Generate V_h according to its corresponding search strategy SX_h;
6      Compute f(V_h), and FEs = FEs + 1;
7      if f(V_h) < f(X_h) then
8          X_h = V_h;
9          trial_h = 0;
10     else
11         trial_h = trial_h + 1;
12     end
13     Update the strategy SX_h according to Eq. (13);
14 end
```

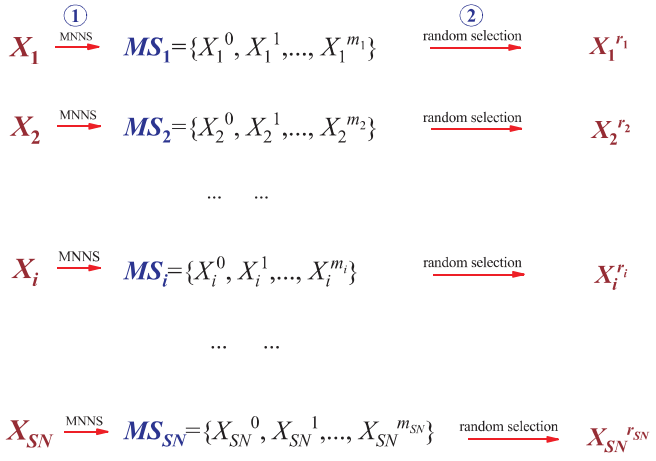


Fig. 3. A random selection method based on MNNS.

3.5. Framework of the proposed NNSABC

In this paper, an improved ABC based on modified nearest neighbor sequence (called NNSABC) is proposed. Contrasted with the original ABC, the detailed modifications in NNSABC can be summarized as below.

- Two search strategies are improved based on MNNS. These strategies are used to build a strategy pool (SP). Initially, each solution X_i has an independent strategy SX_i randomly chosen from SP. During the search process, SX_i is updated by Eq. (13).
- A random selection method based MNNS is designed to replace the roulette selection in ABC.

The framework of NNSABC is presented in Algorithm 3, where *FEs* is the number of function evaluations, and *MaxFEs* is the maximum value of *FEs*. The complexity of ABC is mainly affected by three factors: maximum number of iterations (T_{max}), the complexity of the objective function ($O(f)$), and the number of food sources (SN). Then, the complexity of ABC is $O(Tmax \cdot (SN \cdot f + SN \cdot f + f)) = O(T_{max} \cdot SN \cdot f)$. The complexity of our proposed NNSABC is $O(T_{max} \cdot (SN \cdot f + SN \cdot f + f)) = O(T_{max} \cdot SN \cdot f)$. Therefore, NNSABC and ABC have the same complexity.

Algorithm 3: Proposed Approach (NNSABC)

```
Initialization--Generate SN solutions that contain D variables according to Eq. (1);
while FEs ≤ MaxFEs do
    Execute Algorithm 1 (employed bee phase);
    Execute Algorithm 2 (onlooker bee phase);
    if max{trial_i} > limit then
        Replace X_i with a new one produced by Eq. (1) (scout bee phase);
        trial_i = 0;
    end
end
```

4. Experimental study

4.1. Experimental setting

To study the optimization capability of NNSABC, two benchmark sets (50 problems in total) are used in the following experiments. The first set contains 22 classic problems presented in Table 1, where "U"means unimodal, "M"indicates multimodal, and "Accept"is the threshold value. When the best objective function value obtained by an algorithm in a run is below the predefined threshold value, the algorithm is terminated and the run is regarded as a successful run. The specific mathematical definitions of the first benchmark set can be found in Cui et al. (2016). The second set consists of 28 problems derived from the IEEE CEC 2013 optimization competition. For more details about these problems, please refer to Harfouchi et al. (2018).

In order to verify the performance of our approach NNSABC, the whole experiment incudes six series: 1) investigation of the proposed search strategies; 2) comparison results on classical problems; 3) comparison of NNSABC with some neighbor guided ABC variants; 4) comparison of NNSABC with some recent/powerful evolutionary algorithms; 5) study on the impact of the parameter SN; 6) comparison results on the CEC 2013 benchmark problems; and 7) discussions on the search strategies in NNSABC.

For the above seven series of experiments, the involved algorithms are listed in Table 2. For comparisons on the classic problems, we use the suggested parameter settings reported in Cui et al. (2016), i.e. $SN = 50, limit = SN \cdot D$, and $MaxFEs = 5000 \cdot D$, where D is the problem dimension. For each test problem, each algorithm is run 25 times. For the CEC 2013 benchmark, we use the settings reported in Harfouchi et al. (2018), i.e. *MaxFEs* is set to $5000 \cdot D$.

All algorithms used in the experimental analysis were implemented in C++ programming language on Microsoft Visual C++ 6.0. Experiments were performed on a computer with Intel i7-9750H processor (2.60 GHz), 16 GB system memory, and Windows 10.

4.2. Investigation of the proposed search strategies

In order to construct NNSABC, two improved search strategies are designed: 1) a new search strategy based on MNNS and modified ABC/best/1 (called S1); and 2) an improved ABC/best/1 based on step size and MNNS (called S2). In order to objectively analyze the performance of the proposed search strategies, S1, S2, the original search strategy, and three other improved search strategies are embedded into the original ABC. The involved algorithms are listed as follows.

- ABC: the original ABC (Karaboga, 2005);
- GABC: ABC with Gbest guided search strategy (Zhu and Kwong, 2010);
- ABC/best/1: ABC with the search strategy inspired by DE/best/1 (Gao and Liu, 2012);
- ABC/rand/1: ABC with the search strategy inspired by DE/rand/1 (Gao and Liu, 2011);
- ABC + S1: ABC with the proposed search strategy S1;
- ABC + S2: ABC with the proposed search strategy S2.

Table 3 shows the performance of ABC with different search strategies on the 22 classical problems ($D = 30$). From the results, ABC + S1 and ABC + S2 are much better than other compared algorithms. Among six ABC algorithms, the original ABC achieves the worst results. The Gbest guided search strategy is slightly better than the original ABC. ABC/best/1 and ABC/rand/1 obtain much better results than the original ABC and the best guided search strategy. The above results demonstrate the effectiveness of the proposed search strategies S1 and S2.

According to the problem feature, the 22 classical problems are divided into two groups: unimodal problems (f_1-f_9) and multimodal problems (f_{10}-f_{22}). To study the search characteristics of the proposed strategies S1 and S2, Friedman test is used to calculate the average ranking values of ABC + S1 and ABC + S2 on unimodal problems, multimodal problems and the whole benchmark. Results of average ranking are shown in Table 4. As seen, ABC + S2 obtains the best performance on the whole benchmark. For unimodal problems, ABC + S2 is better than ABC + S1, while ABC + S1 outperforms ABC + S2 on multimodal problems. It demonstrates that S1 is good at solving multimodal problems, and S2 prefers unimodal problems.

According to the search equation of S2, all new solutions are produced in the neighborhood of X_{best}. For unimodal problems, S2 can quickly help ABC find good solutions by the guidance of the global bet solution (X_{best}). However, too much reliance on X_{best} easily results in premature convergence on multimodal problems. The modified nearest neighbor sequence of X_i is $MS_i = \{X_i^0, X_i^1, X_i^2, \ldots, X_i^{m_i}\}$. Then, the mean center of the sequence MS_i is MCX_i. According to the search equation of S1, a new solution is generated in the neighborhood of MCX_i. In general, MCX_i is an excellent solution. Compared with X_{best}, MCX_i can be regarded as a local best solution. Though MCX_i can guide the search and help ABC find good solutions, it reduces the reliance on X_{best}. Moreover, new solutions are located at the neighborhood of different mean centers. This can help ABC avoid falling local minima. Therefore,

Table 1
The 22 classical benchmark problems.

Problems	Name	Accept	Feature	Problems	Name	Accept	Feature
f_1	Sphere	1×10^{-8}	U	f_{12}	NCRastrigin	1×10^{-8}	M
f_2	Elliptic	1×10^{-8}	U	f_{13}	Griewank	1×10^{-8}	M
f_3	SumSquare	1×10^{-8}	U	f_{14}	Schwefel 2.26	1×10^{-8}	M
f_4	SumPower	1×10^{-8}	U	f_{15}	Ackley	1×10^{-8}	M
f_5	Schwefel 2.22	1×10^{-8}	U	f_{16}	Penalized 1	1×10^{-8}	M
f_6	Schwefel 2.21	1×10^{0}	U	f_{17}	Penalized 2	1×10^{-8}	M
f_7	Step	1×10^{-8}	U	f_{18}	Alpine	1×10^{-8}	M
f_8	Exponential	1×10^{-8}	U	f_{19}	Levy	1×10^{-8}	M
f_9	Quartic	1×10^{-1}	U	f_{20}	Weierstrass	1×10^{-8}	M
f_{10}	Rosenbrock	1×10^{-1}	M	f_{21}	Himmelblau	-78	M
f_{11}	Rastrigin	1×10^{-8}	M	f_{22}	Michalewicz	−29, −49, −99	M

Table 2
Involved algorithms for comparisons.

Algorithms	Reference	Year
The original ABC (ABC)	Karaboga (2005)	2005
Gbest guided ABC (GABC)	Zhu and Kwong (2010)	2010
ABC with variable search strategy (ABCVSS)	Kiran et al. (2015)	2015
Bare bones ABC (BABC)	Gao et al. (2015)	2015
ABC with depth-first search and elite strategy (DFSABC-elite)	Cui et al. (2016)	2016
A quick ABC (qABC)	Karaboga and Gorkemli (2014)	2014
ABC with multiple neighborhood topologies (ABC-MNT)	Zhou et al. (2021)	2021
Best neighbor-guided ABC (NABC)	Peng et al. (2019)	2019
ABC with adaptive neighborhood search (ABCN)	Xiao et al. (2021)	2021
DE with composite strategies and control parameters (CoDE)	Wang et al. (2011)	2011
JADE: Adaptive differential evolution with optional external archive (JADE)	Zhang and Sanderson (2009)	2009
Self-adaptive DE (SaDE)	Qin et al. (2008)	2009
DE using population size reduction and three strategie (jDEscop)	Brest and Maucec (2011)	2011
Comprehensive learning PSO (CLPSO)	Liang et al. (2006)	2006
Covariance matrix adaptation evolutionary strategy (CMA-ES)	Hansen and Ostermeier (2001)	2001
Accelerating ABC with an adaptive local search (AABCLS)	Jadon et al. (2015)	2015
ABC with Powell's local search (PABC)	Gao et al. (2013)	2013
Modified ABC (MABC)	Akay and Karaboga (2012)	2012
Compact ABC (comABC)	Dao et al. (2014)	2014
Enhanced compact ABC (EcABC)	Banitalebi et al. (2015)	2015
Compact DE (cDE)	Mininno et al. (2011)	2011
Memetic compact DE (McDE)	Neri and Mininno (2010)	2010
Modified cooperative learning ABC (mCLABC)	Harfouchi et al. (2018)	2018
Our approach NNSABC	–	–

S1 is better than S2 on multimodal problems, but S2 is better than S1 on unimodal problems.

4.3. Comparison results on classical problems

In this section, NNSABC is compared with five ABC variants, i.e., ABC (Karaboga, 2005), GABC (Zhu and Kwong, 2010), ABCVSS (Kiran et al., 2015), BABC (Gao et al., 2015), and DFSABC-elite (Cui et al., 2016) on the classical problems with 30*D*, 50*D*, and 100*D*. Results of ABC, GABC, ABCVSS, BABC, and DFSABC-elite were directly taken from the literature (Cui et al., 2016). In order to quantitatively measure the convergence rate and robustness of NNSABC and other compared ABCs, two indicators called average *FEs* (AVEN) and successful rate (SR%) are utilized (Cui et al., 2016). When an algorithm reaches the "Accept"value, the algorithm is stopped and the cost *FEs* is recorded. For all successful runs, AVEN is obtained by calculating the average *FEs*. A smaller AVEN means that the corresponding algorithm has faster convergence rate. SR is the ratio of the number of successful runs to 25 independent runs. A larger SR means that the corresponding algorithm is more robust.

Table 5 lists the average best objective function values of six ABCs for $D = 30$, where "NA"represents that the corresponding algorithm cannot reach the "Accept"value in all 25 independent runs. The overall comparison results of NNSABC and other ABCs are described by $w/t/l$ in the last row. The symbol w means that our approach NNSABC outperforms the compared algorithm on w problems. The symbol t indicates that our approach NNSABC and the compared algorithm have similar performance on t problems. The last symbol l implies that the compared algorithm performs better than our approach NNSABC on l problems. For f_7,f_8, and f_{21}, all six ABCs converge to the global minima. ABC, ABCVSS, BABC, and DFSABC-elite outperform NNSABC on two problems, while NNSABC is not worse than them on the rest of 20 problems. GABC is better than NNSABC on only one problem f_{22}, but NNSABC is superior to GABC on 17 problems. It is noted that NNSABC converges to the global optimum or sub-optimum on 20 problems (except for f_{10} and f_{22}).

Tables 6 and 7 list the comparison results on classical problems with $D = 50$ and 100, respectively. It aims to investigate the sensitivity of NNSABC on scalable dimensions. From the results, NNSABC can find higher accuracy solutions than ABC on most problems. For $D = 50$, NNSABC is better than GABC, ABCVSS, and DFSABC-elite on 15, 13, and 10 problems, respectively. NNSABC is superior to BABC on 12 problems. As the dimension D increases to 100, NNSABC still obtains better results than other ABCs on the majority of problems. Compared with ABC, GABC, ABCVSS, DFSABC-elite, and NSABC, NNSABC wins on 17, 15, 14, 13, and 9 problems, respectively. Results demonstrate that the growth of D does not seriously affect the overall performance of NNSABC. NNSABC gains better performance on some problems as the dimension increases. The main reason is that the maximum number of function evaluations (*MaxFEs*) is increased with the growth of the problem dimension.

Fig. 4 shows the convergence curves of NNSABC on nine test problems with $D = 30$. We only list the convergence curves of NNSABC, because results of other compared algorithms were taken from the literature (Cui et al., 2016). As seen, NNSABC shows fast convergence rate during the search process. To quantitatively measure the convergence rate of NNSABC and other compared ABCs, the performance metric AVEN is calculated. From the results, the AVEN values achieved by NNSABC is much smaller than other algo-

Table 3
Results of ABC with different search strategies on the classical problems ($D = 30$).

Problems	ABC Mean (SD)	GABC Mean (SD)	ABC/best/1 Mean (SD)	ABC/rand/1 Mean (SD)	ABC + S1 Mean (SD)	ABC + S2 Mean (SD)
f_1	1.04E−17 (1.20E−17)	1.07E−30 (6.09E−31)	9.98E−36 (4.87E−35)	4.76E−34 (1.17E−33)	8.37E−62 (2.81E−61)	**1.42E−108 (3.09E−107)**
f_2	4.38E−10 (4.72E−10)	2.64E−24 (2.48E−24)	1.69E−32 (5.74E−33)	5.47E−29 (3.85E−28)	3.90E−59 (2.15E−58)	**1.82E−105 (7.31E−104)**
f_3	1.14E−19 (9.89E−20)	6.14E−32 (4.74E−32)	7.98E−37 (1.86E−36)	4.43E−34 (1.32E−34)	1.41E−63 (1.52E−61)	**2.72E−109 (1.34E−107)**
f_4	2.02E−31 (5.30E−31)	6.78E−50 (2.91E−49)	5.40E−63 (1.38E−61)	2.36E−43 (7.07E−41)	1.25E−127 (4.08E−126)	**5.73E−176 (5.38E−157)**
f_5	7.69E−11 (3.04E−11)	8.30E−17 (3.10E−17)	4.94E−20 (5.04E−19)	4.68E−18 (2.71E−17)	3.30E−34 (4.75E−32)	**2.93E−56 (1.97E−55)**
f_6	4.39E+00 (1.07E+00)	2.47E−01 (5.89E−02)	4.56E−01 (2.00E+00)	1.16E+00 (6.48E−01)	1.83E−01 (2.55E−01)	**4.88E−02 (2.22E−01)**
f_7	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_8	**7.18E−66 (5.21E−73)**	**7.18E−66 (1.07E−75)**	**7.18E−66 (1.33E−78)**	**7.18E−66 (1.24E−77)**	**7.18E−66 (1.29E−79)**	**7.18E−66 (1.93E−78)**
f_9	6.02E−02 (1.09E−02)	3.08E−02 (6.44E−03)	2.13E−02 (2.54E−02)	2.21E−02 (1.68E−02)	1.72E−02 (2.16E−02)	**1.28E−02 (1.37E−02)**
f_{10}	**5.45E−02 (5.86E−02)**	4.78E+00 (1.54E+01)	5.06E+00 (1.01E+00)	7.63E−01 (3.07E+00)	1.60E+01 (9.88E+01)	1.42E+01 (4.38E+01)
f_{11}	3.50E−14 (1.35E−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)**	**0.00E+00 (0.00E+00)**
f_{12}	1.70E−12 (4.36E−12)	8.38E−15 (4.19E−14)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{13}	2.36E−14 (5.62E−14)	1.48E−11 (7.39E−11)	**0.00E+00 (0.00E+00)**	1.61E−11 (4.93E−09)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{14}	4.58E−12 (1.59E−12)	**3.42E−12 (6.03E−13)**	4.59E−11 (1.74E−11)	5.52E−12 (6.35E−12)	5.32E−12 (1.47E−12)	2.59E−11 (1.49E−11)
f_{15}	4.31E−09 (1.85E−09)	3.71E−14 (5.78E−15)	1.03E−14 (3.67E−14)	8.10E−14 (7.58E−13)	**7.45E−15 (8.54E−15)**	2.98E−14 (4.49E−14)
f_{16}	1.03E−18 (6.90E−19)	7.53E−32 (3.06E−32)	**1.57E−32 (1.50E−47)**	**1.57E−32 (1.50E−47)**	**1.57E−32 (1.50E−47)**	**1.57E−32 (4.50E−47)**
f_{17}	4.88E−18 (5.03E−18)	4.83E−31 (2.50E−31)	**1.50E−33 (3.00E−47)**	1.81E−32 (4.58E−32)	**1.50E−33 (3.00E−47)**	**1.50E−33 (3.00E−47)**
f_{18}	2.35E−06 (1.66E−06)	5.22E−07 (1.12E−06)	7.59E−15 (1.96E−13)	4.99E−10 (1.81E−10)	**1.64E−15 (1.95E−15)**	6.66E−15 (1.00E−14)
f_{19}	4.46E−14 (5.39E−14)	1.40E−30 (1.63E−30)	**1.35E−31 (0.00E+00)**	**1.35E−31 (0.00E+00)**	**1.35E−31 (0.00E+00)**	**1.35E−31 (0.00E+00)**
f_{20}	2.06E−02 (2.35E−02)	3.62E−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 (0.00E+00)**
f_{21}	**−78.332 (0.00E+00)**	**−78.332 (5.02E−15)**	**−78.332 (2.41E−14)**	**−78.332 (4.96E−14)**	**−78.332 (0.00E+00)**	**−78.332 (0.00E+00)**
f_{22}	**−29.999 (6.36E−04)**	−29.997 (1.72E−03)	−29.623 (1.85E−02)	−29.630 (1.05E−06)	−29.669 (1.86E−04)	−29.677 (2.48E−03)

Table 4
Average ranking of ABC with different search strategies obtained by Friedman test.

Algorithms	Unimodal problems Average ranking	Multimodal problems Average ranking	The whole benchmark Average ranking
ABC	5.44	4.50	4.89
GABC	4.33	4.04	4.16
ABC/best/1	3.22	3.19	3.20
ABC/rand/1	4.11	3.58	3.8 0
ABC + S1	2.33	**2.73**	2.57
ABC + S2	**1.56**	2.96	**2.39**

rithms on most problems. It demonstrates that the convergence rate of NNSABC is much faster than other compared ABC algorithms. Furthermore, NNSABC is more robust than other ABCs in terms of the SR values.

Friedman test is used to further compare the overall performance of NNSABC and other ABCs (Wang et al., 2020; Xiao et al., 2019). The calculation of the average ranking values is based on the Friedman test. A smaller ranking value implies a better performance. The average ranking values of those contrasted algorithms are listed in Table 8. As shown, NNSABC gets the smallest average ranking value for $D = 30$, 50, and 100. It means that NNSABC obtains the best overall performance among six ABCs.

4.4. Comparison of NNSABC with some neighbor guided ABC variants

Recently, some neighbor guided ABC variants were proposed. Our approach NNSABC is also a class of neighbor guided ABCs. In order to evaluate the effect of different neighborhood strategies, the proposed NNSABC is compared with four other neighbor guided ABCs in this section. The involved algorithms are listed as follows.

- Quick ABC (qABC) (Karaboga and Gorkemli, 2014);
- ABC with multiple neighborhood topologies (ABC-MNT) (Zhou et al., 2021);
- Best neighbor guided ABC (NABC) (Peng et al., 2019);
- ABC with adaptive neighborhood search (ABCN) (Xiao et al., 2021);
- The proposed NNSABC.

In the experiment, all five neighbor guided ABCs are tested on the 22 classical problems with $D = 30$. The parameters of these four algorithms are set as the suggestion in their original papers. Table 9 shows the comparison results achieved by five neighbor guided ABC variants. As seen, NNSABC outperforms qABC on 17 problems. Both of them obtain the same results on the rest of five problems. ABC-MNT performs better than NNSABC on 3 problems, while NNSABC wins on 9 problems. NNSABC surpasses NABC and ABCN on 12 problems. For the rest of 10 problems, NABC is better than NNSABC on only one problem f_9. Like qABC, ABCN cannot obtain better results than NNSABC on all problems. Table 10 presents the average rankings of qABC, ABC-MNT, NABC, ABCN, and NNSABC achieved by Friedman test. As shown, the best average ranking is achieved by NNSABC. It demonstrates the effectiveness of the proposed neighborhood strategy.

4.5. Comparison of NNSABC with some recent/powerful evolutionary algorithms

In this section, NNSABC is compared with seven other recent/powerful evolutionary algorithms, including CoDE (Wang et al., 2011), JADE (Zhang and Sanderson, 2009), SaDE (Qin et al., 2008), jDEscop (Brest and Maucec, 2011), CLPSO (Liang et al., 2006), CMA-ES (Hansen and Ostermeier, 2001), and AABCLS (Jadon et al., 2015) on the classical problems with $D = 50$.

Table 11 presents the comparison results achieved by NNSABC and seven other evolutionary algorithms. Results of CoDE, JADE, SaDE, jDEscop, CLPSO, CMA-ES, and AABCLS were directly taken from the literature (Cui et al., 2016). The parameter *MaxFES* for all these algorithms is set to 250,000, and each algorithm runs 25 times for each test function. From the results, it is obvious that NNSABC performs better than other algorithms on most test problems. Compared with JADE and jDEscop, NNSABC achieves better results on 16 problems. CoDE and JADE outperforms NNSABC on three problems. SaDE is better than NNSABC on only one problem f_9. CLPSO is worse than NNSABC on 17 problems, and they obtain the same results on three of the remaining five problems. Though AABCLS achieves excellent solutions on most problems, it is still worse than NNSABC on 13 problems. The average ranking values of the above eight algorithms obtained by Friedman test are given

Table 5
Comparison results on classical problems when $D = 30$.

Problems	ABC Mean (SD) SR/AVEN	GABC Mean (SD) SR/AVEN	ABCVSS Mean (SD) SR/AVEN	BABC Mean (SD) SR/AVEN	DFSABC-elite Mean (SD) SR/AVEN	NNSABC Mean (SD) SR/AVEN
f_1	1.04E−17 (1.20E−17) **100**/83,702	1.07E−30 (6.09E−31) **100**/53,134	2.40E−35 (8.54E−35) **100**/50,526	1.14E−43 (1.77E−43) **100**/43,530	4.14E−82 (8.76E−82) **100**/21,410	**1.82E−144 (4.46E−143)** **100/9317**
f_2	4.38E−10 (4.72E−10) **100**/136,290	2.64E−24 (2.48E−24) **100**/81,406	2.29E−27 (9.79E−27) **100**/78,802	4.18E−30 (3.33E−17) **100**/83,026	5.37E−78 (8.66E−78) **100**/28,674	**1.59E−132 (3.90E−131)** **100/11,486**
f_3	1.14E−19 (9.89E−20) **100**/75,402	6.14E−32 (4.74E−32) **100**/48,446	9.40E−37 (2.54E−36) **100**/46,222	7.40E−15 (3.70E−14) **100**/38,022	2.84E−83 (4.66E−83) **100**/19,710	**1.05E−142 (2.56E−141)** **100/8,972**
f_4	2.02E−31 (5.30E−31) **100**/23,578	6.78E−50 (2.91E−49) **100**/14,890	4.31E−44 (1.40E−43) **100**/15,818	4.96E−90 (1.54E−89) **100**/11,222	2.41E−110 (1.19E−109) **100**/7122.0	**5.05E−211 (0.00E+00)** **100/4,062**
f_5	7.69E−11 (3.04E−11) **100**/124,870	8.30E−17 (3.10E−17) **100**/81,006	7.03E−19 (2.18E−18) **100**/72,958	1.61E−24 (8.21E−25) **100**/58,046	2.06E−42 (2.08E−42) **100**/33,426	**5.93E−72 (1.45E−70)** **100/14,980**
f_6	4.39E+00 (1.07E+00) 0/NA	2.47E−01 (5.89E−02) **100**/110,270	2.56E−01 (9.19E−02) **100**/111,070	1.71E+00 (1.15E+00) 32/122,490	**5.08E−07 (3.69E−07)** **100/32,802**	2.25E−03 (1.21E−02) **100**/43,145
f_7	**0.00E+00 (0.00E+00)** **100**/10,994	**0.00E+00 (0.00E+00)** **100**/10,606	**0.00E+00 (0.00E+00)** **100**/10,042	**0.00E+00 (0.00E+00)** **100**/9426	**0.00E+00 (0.00E+00)** **100**/7534.0	**0.00E+00 (0.00E+00)** **100/4334**
f_8	**7.18E−66 (5.21E−73)** **100**/150	**7.18E−66 (1.07E−75)** **100**/150	**7.18E−66 (9.98E−78)** **100**/150	**7.18E−66 (2.04E−77)** **100**/150	**7.18E−66 (3.23E−81)** **100**/150	**7.18E−66 (1.08E−78)** **100/76**
f_9	6.02E−02 (1.09E−02) **100**/91,786	3.08E−02 (6.44E−03) **100**/48,050	2.57E−02 (5.22E−03) **100**/40,846	2.70E−02 (8.28E−03) **100**/35,582	1.20E−02 (3.80E−03) **100**/16,878	**1.08E−02 (1.47E−02)** **100/11,556**
f_{10}	5.45E−02 (5.86E−02) 88/**11,014**	4.78E+00 (1.54E+01) 60/82,303	**3.25E−02 (4.58E−02)** 96/86,483	3.97E−02 (4.96E−02) **100**/83,026	3.45E+00 (1.45E+01) 60/58,683	4.67E−01 (1.53E+00) 72/80,273
f_{11}	3.50E−14 (1.35E−13) **100**/99,134	**0.00E+00 (0.00E+00)** **100**/74,522	**0.00E+00 (0.00E+00)** **100**/51,966	**0.00E+00 (0.00E+00)** **100**/41,354	**0.00E+00 (0.00E+00)** **100**/27,754	**0.00E+00 (0.00E+00)** **100/13300**
f_{12}	1.70E−12 (4.36E−12) **100**/112,080	8.38E−15 (4.19E−14) **100**/83,602	**0.00E+00 (0.00E+00)** **100**/60,578	**0.00E+00 (0.00E+00)** **100**/49,050	**0.00E+00 (0.00E+00)** **100**/28,602	**0.00E+00 (0.00E+00)** **100/10966**
f_{13}	2.36E−14 (5.62E−14) **100**/94,862	1.48E−11 (7.39E−11) **100**/71,778	3.45E−11 (1.73E−10) **100**/69,514	**0.00E+00 (0.00E+00)** **100**/42,942	**0.00E+00 (0.00E+00)** **100/31,066**	**0.00E+00 (0.00E+00)** **100**/39,819
f_{14}	4.58E−12 (1.59E−12) **100**/82,946	3.42E−12 (6.03E−13) **100**/65,126	1.60E−12 (3.45E−13) **100**/52,906	2.18E−13 (7.80E−13) **100**/50,418	4.37E−13 (1.09E−12) **100**/34,430	**1.43E−13 (4.92E−13)** **100/32,392**
f_{15}	4.31E−09 (1.85E−09) **100**/145,410	3.71E−14 (5.78E−15) **100**/95,798	6.50E−15 (2.27E−15) **100**/80,074	5.65E−15 (1.33E−15) **100**/65,210	3.80E−15 (1.69E−15) **100**/37,998	**2.19E−15 (1.00E−15)** **100/20,068**
f_{16}	1.03E−18 (6.90E−19) **100**/77,346	7.53E−32 (3.06E−32) **100**/48,178	**1.57E−32 (5.59E−48)** **100**/46,142	8.98E−14 (4.49E−13) **100**/40,542	**1.57E−32 (5.59E−48)** **100/18,902**	**1.57E−32 (1.37E−47)** **100**/25,619
f_{17}	4.88E−18 (5.03E−18) **100**/86,542	4.83E−31 (2.50E−31) **100**/52,566	**1.50E−33 (0.00E+00)** **100**/48,154	**1.50E−33 (8.28E−33)** **100**/40,810	**1.50E−33 (0.00E+00)** **100/20,970**	**1.50E−33 (2.74E−47)** **100**/28,905
f_{18}	2.35E−06 (1.66E−06) 0/NA	5.22E−07 (1.12E−06) 20/144,750	6.26E−18 (2.91E−17) **100**/80,966	3.33E−17 (1.28E−16) **100**/55,262	3.10E−40 (1.03E−39) **100**/40,454	**1.28E−67 (2.34E−66)** **100/21,924**
f_{19}	4.46E−14 (5.39E−14) **100**/90,558	1.40E−30 (1.63E−30) **100**/52,834	**1.35E−31 (2.23E−47)** **100**/48,330	**1.35E−31 (2.23E−47)** **100**/36,362	**1.35E−31 (2.23E−47)** **100/24,890**	**1.35E−31 (0.00E+00)** **100**/29,973
f_{20}	2.06E−02 (2.35E−02) 0/NA	3.62E−02 (0.00E+00) 0/NA	**0.00E+00 (0.00E+00)** **100**/93,050	2.63E−05 (1.32E−04) 96/80,696	**0.00E+00 (0.00E+00)** **100**/55,910	**0.00E+00 (0.00E+00)** **100/29,552**
f_{21}	**−78.332 (0.00E+00)** **100**/26,594	**−78.332 (5.02E−15)** **100**/16,442	**−78.332 (1.05E−14)** **100**/13,038	**−78.332 (1.23E−14)** **100**/10,992	**−78.332 (5.02E−15)** **100/6502.0**	**−78.332 (4.02E−14)** **100**/7934.0
f_{22}	−29.999 (6.36E−04) **100**/25,458	−29.997 (1.72E−03) **100**/22,410	**−30.000 (3.82E−12)** **100**/18,726	**−30.000 (1.92E−06)** **100**/14,822	**−30.000 (0.00E+00)** **100/5270.0**	−29.674 (7.79E−02) 0/NA
w*/*t*/*l	17/3/2	17/4/1	11/9/2	12/8/2	10/10/2	–

in Table 12. From the results, NNSABC obtains the best average ranking among eight algorithms. It demonstrates that our approach NNSABC can achieve competitive performance when compared with some recent/powerful EAs.

4.6. Study on the impact of the parameter SN

In order to study the influence of parameter *SN* on the performance of NNSABC, different values of *SN* are tested on the classical benchmark ($D = 30$). In the experiments, *SN* is set to 20, 30, 40, 50, 60, 70, and 100, respectively.

Results obtained by NNSABC with different *SN* values on the classical benchmark are given in Table 13, and the best results are marked with boldface. As shown, a small *SN* (e.g., $SN = 20$) can significantly improve the performance of NNSABC in terms of the solution accuracy and convergence rate on unimodal functions. For multimodal functions, a large *SN* is a good choice. When *SN* is set to 20 or 100, NNSABC does not obtain better results than other cases. So, the parameter *SN* cannot be too small or too large.

To select an appropriate *SN*, Friedman test is used to analyze the overall performance of NNSABC with different *SN* values. The average rankings achieved by Friedman test for NNSABC with different *SN* values are given in Table 14. From the results, $SN = 50$ obtains the best average ranking among seven different *SN* values. Therefore, $SN = 50$ is used in the proposed approach NNSABC.

4.7. Comparison results on the CEC 2013 benchmark problems

In Section 4.3, the performance of NNSABC is tested on 22 classical problems, which are not difficult to be solved. To further eval-

Table 6
Comparison results on classical problems when $D = 50$.

Problems	ABC Mean (SD) SR/AVEN	GABC Mean (SD) SR/AVEN	ABCVSS Mean (SD) SR/AVEN	BABC Mean (SD) SR/AVEN	DFSABC-elite Mean (SD) SR/AVEN	NNSABC Mean (SD) SR/AVEN
f_1	4.33E−17 (3.19E−17) **100**/143,530	1.86E−29 (1.33E−29) **100**/90,486	6.68E−23 (3.16E−32) **100**/89,554	1.01E−14 (5.07E−14) **100**/76,430	1.58E−81 (3.87E−81) **100**/36,598	**1.85E−150 (4.53E−149)** **100/13,283**
f_2	2.43E−09 (2.27E−09) **100**/233,158	3.88E−23 (2.65E−23) **100**/137,970	1.11E−23 (4.17E−23) **100**/133,690	3.14E−29 (4.95E−29) **100**/143,790	1.53E−77 (4.78E−77) **100**/49,094	**9.37E−153 (2.28E−151)** **100/16,711**
f_3	2.91E−18 (2.97E−18) **100**/132,410	2.30E−30 (1.62E−30) **100**/86,210	1.77E−33 (7.88E−33) **100**/83,378	1.23E−13 (6.16E−13) **100**/68,830	1.77E−82 (1.82E−82) **100**/34,810	**5.68E−148 (1.39E−146)** **100/12,095**
f_4	2.02E−31 (7.03E−31) **100**/38,254	2.26E−51 (5.03E−51) **100**/24,878	7.21E−41 (3.56E−40) **100**/27,142	2.96E−92 (0.00E+00) **100**/18,226	4.01E−109 (2.00E−108) **100**/11,858	**9.57E−218 (0.00E+00)** **100/5459**
f_5	2.40E−10 (6.62E−11) **100**/217,854	4.58E−16 (9.21E−17) **100**/143,970	2.89E−18 (6.52E−18) **100**/126,290	2.18E−06 (1.06E−05) 92/104,000	3.77E−42 (2.23E−42) **100**/57,146	**1.56E−79 (3.42E−78)** **100/19,204**
f_6	1.21E+01 (1.67E+00) 0/NA	2.76E+00 (4.58E−01) 0/NA	2.06E+00 (3.77E−01) 0/NA	7.20E+00 (2.78E+00) 0/NA	**1.40E−04 (4.52E−05)** **100/81,566**	1.46E−02 (4.44E−02) **100**/104,138
f_7	**0.00E+00 (0.00E+00)** **100**/21,526	**0.00E+00 (0.00E+00)** **100**/20,498	**0.00E+00 (0.00E+00)** **100**/18,554	**0.00E+00 (0.00E+00)** **100**/17,414	**0.00E+00 (0.00E+00)** **100**/13,294	**0.00E+00 (0.00E+00)** **100/7100**
f_8	**2.67E−109 (2.36E−116)** **100**/150	**2.67E−109 (1.44E−118)** **100**/150	**2.67E−109 (3.08E−120)** **100**/150	**2.67E−109 (2.62E−116)** **100**/150	**2.67E−109 (9.65E−125)** **100**/150	**2.67E−109 (4.49E−122)** **100/74**
f_9	1.22E−01 (2.17E−02) 12/20,975	6.91E−02 (1.04E−02) **100**/166,960	6.14E−02 (1.38E−02) **100**/150,246	5.74E−02 (1.11E−02) **100**/134,670	1.99E−02 (4.39E−03) **100**/49,710	**1.96E−02 (2.64E−02)** **100/28,804**
f_{10}	6.88E−02 (9.58E−02) 76/183,900	6.81E+00 (1.60E+01) 32/165,860	1.09E−01 (2.52E−01) 80/149,575	**6.29E−02 (1.24E−01)** **88**/144,910	3.49E−01 (8.66E−01) 60/**110,310**	2.44E−01 (1.92E+00) 76/142,583
f_{11}	1.50E−14 (4.40E−14) **100**/172,974	**0.00E+00 (0.00E+00)** **100**/135,300	**0.00E+00 (0.00E+00)** **100**/92,198	**0.00E+00 (0.00E+00)** **100**/73,170	**0.00E+00 (0.00E+00)** **100**/51,834	**0.00E+00 (0.00E+00)** **100/21,793**
f_{12}	2.38E−08 (1.19E−07) 96/200,180	**0.00E+00 (0.00E+00)** **100**/152,640	**0.00E+00 (0.00E+00)** **100**/107,322	**0.00E+00 (0.00E+00)** **100**/90,146	**0.00E+00 (0.00E+00)** **100**/49,538	**0.00E+00 (0.00E+00)** **100/16,170**
f_{13}	2.92E−15 (1.00E−14) **100**/153,382	**0.00E+00 (0.00E+00)** **100**/103,870	3.67E−14 (1.84E−13) **100**/108,290	**0.00E+00 (0.00E+00)** **100**/72,114	**0.00E+00 (0.00E+00)** **100/46,106**	**0.00E+00 (0.00E+00)** **100**/60,086
f_{14}	1.63E−11 (4.35E−12) **100**/138,630	9.90E−12 (2.23E−12) **100**/143,970	4.66E−12 (2.47E−12) **100**/89,742	3.78E−12 (7.28E−13) **100**/89,594	3.64E−12 (0.00E+00) **100/61,926**	**2.72E−12 (2.58E−12)** **100**/88,067
f_{15}	6.69E−09 (3.14E−09) 80/244,100	9.45E−14 (1.45E−14) **100**/166,390	1.70E−14 (5.75E−15) **100**/140,018	7.50E−15 (2.49E−15) **100**/113,410	6.22E−15 (0.00E+00) **100**/63,634	**4.04E−15 (1.74E−15)** **100/26,437**
f_{16}	3.05E−18 (1.98E−18) **100**/131,438	6.26E−31 (5.56E−31) **100**/81,014	1.07E−32 (5.74E−33) **100**/80,394	1.22E−13 (6.09E−13) **100**/68,334	**9.42E−33 (1.40E−48)** **100/31,810**	**9.42E−33 (1.37E−47)** **100**/46,541
f_{17}	3.55E−17 (2.90E−17) **100**/146,466	9.02E−30 (5.39E−30) **100**/91,138	1.80E−33 (1.08E−33) **100**/84,094	2.50E−15 (1.25E−14) **100**/68,638	**1.50E−33 (0.00E+00)** **100/36,138**	**1.50E−33 (2.74E−47)** **100**/53,967
f_{18}	1.16E−05 (1.11E−05) 0/NA	8.97E−06 (1.90E−05) 80/227,800	2.14E−16 (7.43E−16) **100**/148,370	2.07E−16 (7.79E−16) **100**/99,322	6.66E−17 (2.48E−16) **100**/73,294	**9.37E−68 (1.75E−66)** **100/37,406**
f_{19}	6.98E−14 (1.28E−13) **100**/152,594	1.15E−29 (1.34E−29) **100**/91,802	**1.35E−31 (2.23E−47)** **100**/82,116	**1.35E−31 (2.23E−47)** **100**/63,018	**1.35E−31 (2.23E−47)** **100/41,002**	**1.35E−31 (0.00E+00)** **100**/55,106
f_{20}	1.18E−01 (7.56E−02) 0/NA	1.37E−01 (6.61E−02) 0/NA	**0.00E+00 (0.00E+00)** **100**/165,998	**0.00E+00 (0.00E+00)** **100**/148,860	**0.00E+00 (0.00E+00)** **100**/101,222	**0.00E+00 (0.00E+00)** **100/53,018**
f_{21}	**−78.332 (5.80E−15)** **100**/49,842	**−78.332 (0.00E+00)** **100**/30,754	**−78.332 (1.00E−14)** **100**/22,746	**−78.332 (1.16E−14)** **100**/20,186	**−78.332 (1.20E−14)** **100**/12,682	**−78.332 (1.42E−14)** **100/73**
f_{22}	−49.993 (2.25E−03) **100**/55,654	−49.989 (5.66E−03) **100**/52,490	**−50.000 (3.82E−07)** **100**/40,290	**−50.000 (6.79E−06)** **100**/30,986	**−50.000 (0.00E+00)** **100/11,418**	−49.567 (1.07E−01) 0/NA
w*/*t*/*l	17/3/2	15/6/1	13/7/2	12/8/2	10/10/2	–

uate the performance of NNSABC, some complex optimization problems are utilized. In this section, the second benchmark set consists of 28 problems, which were taken from IEEE CEC 2013 optimization competition. Compared with the classical problems, the CEC 2013 benchmark problems are more difficult (shifted, rotated, composite, and non-separable).

In the experiment, the proposed NNSABC is compared with five ABC variants and two DE variants, including PABC (Gao et al., 2013), MABC (Akay and Karaboga, 2012), comABC (Dao et al., 2014), EcABC (Banitalebi et al., 2015), cDE (Mininno et al., 2011), McDE (Neri and Mininno, 2010), and mCLABC (Harfouchi et al., 2018). The problem dimension D is set to 50 and 100.

Tables 15 and 16 present the average error values of eight algorithms on the CEC 2013 benchmark set with $D = 50$ and 100, respectively. Results of PABC, MABC, comABC, EcABC, cDE, McDE, and mCLABC were directly taken from the literature (Harfouchi et al., 2018). For $D = 50$, NNSABC is better than PABC on 21 problems, while PABC outperforms NNSABC on 7 problems. NNSABC is superior to MABC and comABC on 25 and 26 problems, respectively. MABC and EcABC surpasses NNSABC on 3 and 5 problems, respectively. NNSABC performs better than mCLABC on 16 problems. For the rest of 12 problems, mCLABC achieve better results than NNSABC on 10 problems.

Table 7
Comparison results on classical problems when $D = 100$.

Problems	ABC Mean (SD) SR/AVEN	GABC Mean (SD) SR/AVEN	ABCVSS Mean (SD) SR/AVEN	BABC Mean (SD) SR/AVEN	DFSABC-elite Mean (SD) SR/AVEN	NNSABC Mean (SD) SR/AVEN
f_1	4.78E−16 (4.79E−16) **100**/288,310	2.80E−28 (1.17E−28) **100**/190,830	3.84E−31 (1.39E−30) **100**/197,060	2.59E−13 (1.30E−12) **100**/158,450	9.57E−81 (9.73E−81) **100**/76,302	**1.31E−153 (8.22E−153)** **100/25,064**
f_2	5.30E−09 (4.89E−09) 88/472,080	9.41E−22 (7.14E−22) **100**/287,170	2.42E−23 (8.37E−23) **100**/28,672	2.51E−14 (1.25E−13) **100**/301,710	5.50E−77 (6.42E−77) **100**/102,360	**7.31E−155 (1.76E−154)** **100/31,714**
f_3	1.03E−16 (8.69E−17) **100**/283,630	1.08E−28 (4.72E−29) **100**/185,010	9.03E−29 (4.51E−28) **100**/185,670	1.38E−14 (4.52E−14) **100**/150,390	3.90E−81 (8.94E−81) **100**/74,462	**1.58E−152 (3.87E−151)** **100/25,048**
f_4	1.17E−31 (4.89E−31) **100**/76,902	2.77E−50 (1.12E−49) **100**/48,938	4.81E−47 (1.72E−46) **100**/53,354	2.44E−91 (9.04E−91) **100**/37,402	2.31E−109 (7.26E−109) **100**/23,382	**7.36E−246 (0.00E+00)** **100/12,168**
f_5	1.27E−09 (4.11E−10) **100**/464,180	3.00E−15 (6.55E−16) **100**/306,110	2.26E−16 (4.87E−16) **100**/274,820	3.09E−07 (1.54E−06) 96/212,680	2.13E−41 (2.55E−41) **100**/119,260	**2.32E−79 (5.69E−78)** **100/33,794**
f_6	2.85E+01 (1.31E+00) 0/NA	1.72E+01 (1.34E+00) 0/NA	1.72E+01 (1.43E+00) 0/NA	1.80E+01 (4.07E+00) 0/NA	**3.90E−02 (1.50E−02)** **100/286,090**	4.65E−01 (6.62E−01) **100**/404,590
f_7	**0.00E+00 (0.00E+00)** **100**/51,250	**0.00E+00 (0.00E+00)** **100**/47,670	**0.00E+00 (0.00E+00)** **100**/43,430	**0.00E+00 (0.00E+00)** **100**/40,426	**0.00E+00 (0.00E+00)** **100**/28,982	**0.00E+00 (0.00E+00)** **100/16,761**
f_8	**7.12E−218 (0.00E+00)** **100**/150	**7.12E−218 (0.00E+00)** **100**/150	**7.12E−218 (0.00E+00)** **100**/150	**7.12E−218 (2.32E−16)** **100**/150	**7.12E−218 (0.00E+00)** **100**/150	**7.12E−218 (0.00E+00)** **100/70**
f_9	2.89E−01 (5.49E−02) 0/NA	1.74E−01 (2.05E−02) 0/NA	1.50E−01 (2.64E−02) 40/389,150	1.82E−01 (3.92E−02) 0/NA	5.02E−02 (5.94E−03) **100**/241,390	**3.67E−02 (4.15E−02)** **100/94,852**
f_{10}	1.82E−01 (2.04E−01) 40/331,270	1.96E+01 (3.28E+01) 32/435,640	8.46E−02 (1.65E−01) 76/303,730	**3.76E−02 (4.32E−02)** **96**/353,780	6.28E+00 (1.90E+01) 24/**225,830**	1.86E+00 (2.16E+01) 56/347,849
f_{11}	2.28E−08 (1.14E−07) 96/392,980	**0.00E+00 (0.00E+00)** **100**/306,500	**0.00E+00 (0.00E+00)** **100**/191,710	**0.00E+00 (0.00E+00)** **100**/159,730	**0.00E+00 (0.00E+00)** **100**/116,780	**0.00E+00 (0.00E+00)** **100/41,523**
f_{12}	1.69E−01 (3.97E−01) 80/44,406	**0.00E+00 (0.00E+00)** **100**/338,930	**0.00E+00 (0.00E+00)** **100**/226,480	4.00E−02 (2.00E−01) 96/201,940	**0.00E+00 (0.00E+00)** **100**/112,740	**0.00E+00 (0.00E+00)** **100/31,441**
f_{13}	9.77E−17 (1.90E−16) **100**/296,480	**0.00E+00 (0.00E+00)** **100**/201,490	1.67E−15 (8.33E−15) **100**/206,570	**0.00E+00 (0.00E+00)** **100**/140,960	**0.00E+00 (0.00E+00)** **100**/88,886	**0.00E+00 (0.00E+00)** **100/78,729**
f_{14}	5.79E−11 (7.42E−12) **100**/317,040	3.67E−11 (1.46E−12) **100**/246,620	2.82E−11 (7.07E−12) **100**/192,540	2.27E−11 (3.20E−12) **100**/231,850	2.18E−11 (3.52E−17) **100/141,040**	**1.20E−12 (8.05E−14)** **100**/166,682
f_{15}	1.42E−08 (4.33E−09) 16/497,580	3.34E−13 (4.87E−14) **100**/351,020	3.58E−14 (1.03E−14) **100**/282,110	2.09E−14 (2.77E−15) **100**/240,070	**1.30E−14 (9.84E−16)** **100**/129,850	1.78E−14 (4.77E−14) **100/46,536**
f_{16}	1.48E−17 (1.08E−17) **100**/249,150	6.80E−30 (3.09E−30) **100**/163,190	6.29E−32 (2.05E−31) **100**/171,940	7.89E−15 (3.89E−14) **100**/143,220	**4.71E−33 (6.98E−49)** **100/63,918**	**4.71E−33 (6.84E−48)** **100**/105,905
f_{17}	3.27E−16 (4.05E−16) **100**/302,550	1.54E−28 (8.05E−29) **100**/191,490	7.68E−32 (3.58E−31) **100**/187,960	**1.50E−33 (0.00E+00)** **100**/141,510	**1.50E−33 (0.00E+00)** **100/75,962**	**1.50E−33 (2.74E−47)** **100**/127,038
f_{18}	3.96E−04 (4.66E−04) 0/NA	9.26E−05 (7.96E−05) 0/NA	1.32E−13 (5.33E−13) **100**/320,760	1.09E−16 (2.32E−16) **100**/212,810	2.38E−16 (3.55E−16) **100**/155,110	**1.80E−71 (2.99E−70)** **100/76,528**
f_{19}	4.47E−14 (5.35E−14) **100**/305,340	6.88E−29 (4.80E−29) **100**/189,960	**1.35E−31 (2.23E−47)** **100**/169,640	**1.35E−31 (2.23E−47)** **100**/131,170	**1.35E−31 (2.23E−47)** **100/78,810**	**1.35E−31 (0.00E+00)** **100**/128,382
f_{20}	4.54E−01 (1.24E−01) 0/NA	2.82E−01 (2.39E−01) 0/NA	1.36E−14 (6.82E−14) **100**/348,830	1.80E−04 (8.90E−04) 92/334,880	**0.00E+00 (0.00E+00)** **100**/214,830	**0.00E+00 (0.00E+00)** **100/102,123**
f_{21}	**−78.332 (2.23E−14)** **100**/114,880	**−78.332 (1.36E−14)** **100**/69,386	**−78.332 (2.90E−15)** **100**/49,382	**−78.332 (0.00E+00)** **100**/44,382	**−78.332 (0.00E+00)** **100**/27,598	**−78.332 (1.64E−15)** **100/70**
f_{22}	−99.970 (5.00E−03) **100**/160,730	−99.951 (1.15E−02) **100**/153,310	**−100.000 (2.48E−04)** **100**/119,990	**−100.000 (4.96E−06)** **100**/91,062	**−100.000 (4.52E−07)** **100/32,006**	−99.485 (1.46E−01) 0/NA
w*/*t*/*l	17/3/2	15/6/1	14/6/2	13/7/2	9/10/3	–

For $D = 100$, NNSABC is not worse than cDE and comABC on all problems. NNSABC is worse than McDE on only one problem. For the rest of 27 problems, NNSABC wins on 26 problems. Compared with EcABC, NNSABC wins on 24 problems and loses on two problems. mCLABC surpasses NNSABC on eight problems, but NNSABC achieves better results on 16 problems. With the growth of dimension, our approach NNSABC still outperforms seven other compared algorithms on the majority of problems. The average ranking values of the above eight algorithms are given in Table 17. From the results, NNSABC obtains the best average ranking for $D = 50$ and 100. It demonstrates that our approach NNSABC can achieve competitive performance on complex problems.

4.8. Discussions on the search strategies in NNSABC

The main contributions of NNSABC can be attributed to the proposed two search strategies S1 and S2. In NNSABC, S1 and S2 are used to construct a strategy pool (SP). Each solution X_i is assigned a strategy $SX_i \in \{S1, S2\}$. At the initial stage, the search strategy SX_i is randomly selected from SP. During the search process, the strategy SX_i is dynamically updated in terms of Eq. (13). To study the effectiveness of S1 and S2 in the search process, we observe the frequency of using S1 and S2 in the entire population.

Let $NSX_j(t)$ be the number solutions using the jth strategy in the population at the tth iteration. In the search process, we focus on investigating the changes of $NSX_j(t)$. This helps to recognize the roles of different strategies at different evolutionary stages.

Fig. 5 presents the changes of strategies in the population during the search process. As seen, NSX_2 is larger than NSX_1 for f_1 and f_2 at the whole search stages. It means that S2 plays an important role in the current population, and more candidate solutions are generated by this strategy. Especially for the initial and last search stages, more search operations are conducted by S2 than S1. The main reason is that f_1 and f_2 are unimodal problems. In Section 4.2,

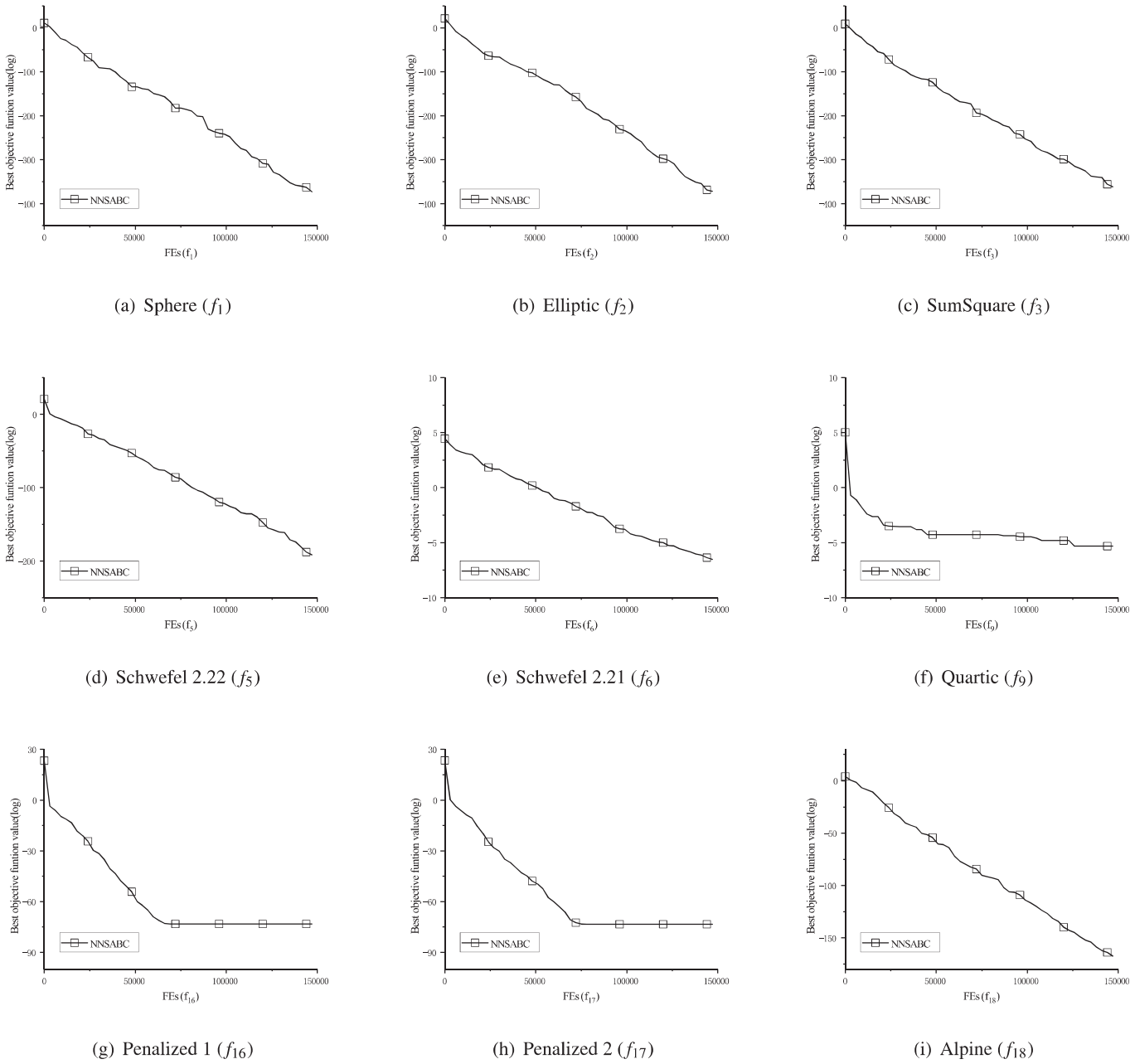


Fig. 4. Convergence curves of NNSABC on nine classical problems with $D = 30$.

Table 8
Average ranking values of six ABC algorithms obtained by Friedman test.

Algorithms	$D = 30$ Average ranking	$D = 50$ Average ranking	$D = 100$ Average ranking
ABC	5.20	5.07	5.20
GABC	4.57	4.23	4.15
ABCVSS	3.27	3.48	3.36
BABC	3.32	3.64	3.75
DFSABC-elite	2.48	2.41	2.36
NNSABC	**2.18**	**2.18**	**2.18**

Table 9
Comparison results achieved by five neighbor guided ABC variants.

Problems	qABC Mean (SD)	ABC-MNT Mean (SD)	NABC Mean (SD)	ABCN Mean (SD)	NNSABC Mean (SD)
f_1	5.54E−15 (9.48E−15)	5.20E−39 (8.84E−39)	1.60E−34 (6.90E−34)	7.52E−97 (5.37E−96)	**1.82E−144 (4.46E−143)**
f_2	2.55E−10 (4.08E−10)	2.39E−35 (4.31E−35)	2.68E−29 (6.20E−29)	7.28E−86 (1.14E−84)	**1.59E−132 (3.90E−131)**
f_3	3.67E−16 (2.82E−16)	3.57E−39 (7.59E−39)	2.72E−38 (5.51E−38)	8.97E−94 (4.02E−92)	**1.05E−142 (2.56E−141)**
f_4	4.23E−21 (1.04E−20)	3.91E−81 (1.06E−80)	4.25E−63 (1.71E−62)	4.13E−91 (7.65E−89)	**5.05E−211 (0.00E+00)**
f_5	2.17E−08 (6.03E−09)	1.43E−26 (1.22E−26)	1.66E−19 (2.42E−19)	5.07E−57 (1.85E−54)	**5.93E−72 (1.45E−70)**
f_6	1.65E−01 (3.88E−02)	**2.07E−06 (1.22E−06)**	4.67E+00 (2.35E+00)	5.27E−03 (1.03E−01)	2.25E−03 (1.21E−02)
f_7	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_8	**7.18E−66 (8.06E−72)**	**7.18E−66 (6.17E−78)**	**7.18E−66 (2.66E−74)**	**7.18E−66 (4.35E−78)**	**7.18E−66 (1.08E−78)**
f_9	3.47E−02 (7.70E−03)	**1.42E−03 (8.63E−04)**	1.47E−03 (6.89E−04)	1.33E−02 (1.06E−01)	1.08E−02 (1.47E−02)
f_{10}	7.49E−01 (8.02E−01)	4.76E+01 (1.66E+01)	4.88E+01 (4.29E+01)	7.90E+00 (5.42E+00)	**4.67E−01 (1.53E+00)**
f_{11}	1.71E−01 (4.10E−10)	**0.00E+00 (0.00E+00)**	**0.00E+00 (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 (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)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{14}	2.03E−10 (8.58E−10)	8.39E−13 (5.66E−13)	4.16E−13 (2.22E−12)	2.26E−13 (6.37E−13)	**1.43E−13 (4.92E−13)**
f_{15}	2.72E−06 (1.21E−06)	3.61E−15 (2.03E−15)	3.33E−14 (1.03E−14)	3.31E−14 (2.97E−14)	**2.19E−15 (1.00E−15)**
f_{16}	2.83E−14 (1.86E−13)	**1.57E−32 (1.50E−47)**	**1.57E−32 (8.20E−33)**	**1.57E−32 (4.50E−47)**	**1.57E−32 (1.37E−47)**
f_{17}	2.08E−15 (2.11E−15)	**1.50E−33 (3.00E−47)**	**1.50E−33 (1.69E−33)**	**1.50E−33 (3.00E−47)**	**1.50E−33 (2.74E−47)**
f_{18}	1.15E−05 (3.19E−05)	2.33E−10 (7.13E−10)	5.73E−04 (1.46E−03)	9.77E−59 (6.52E−53)	**1.28E−67 (2.34E−66)**
f_{19}	3.15E−09 (6.66E−09)	**1.35E−31 (4.21E−46)**	**1.35E−31 (9.57E−33)**	**1.35E−31 (4.11E−46)**	**1.35E−31 (0.00E+00)**
f_{20}	2.38E−02 (2.12E−02)	**0.00E+00 (0.00E+00)**	4.15E−01 (4.20E−01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{21}	**−78.332 (0.00E+00)**	**−78.332 (0.00E+00)**	**−78.332 (2.72E−14)**	**−78.332 (0.00E+00)**	**−78.332 (4.02E−14)**
f_{22}	−29.611 (7.79E−01)	**−29.686 (1.13E+00)**	−29.653 (8.10E−02)	−29.625 (4.68E−01)	−29.674 (7.79E−02)
$w/t/l$	17/5/0	9/10/3	12/9/1	12/10/0	–

Table 10
Average ranking values of qABC, ABC-MNT, NABC, ABCN, and NNSABC achieved by Friedman test.

Algorithms	Average ranking
qABC	4.09
ABC-MNT	2.64
NABC	3.45
ABCN	2.68
NNSABC	**2.14**

experimental results show that S2 performs better than S1 on unimodal problems and S1 outperforms S2 on multimodal problems.

For f_{13} and f_{17}, they are multimodal problems. It is obvious that more search operations are conducted by S1 than S2. At the initial stages, the search prefers to conduct S2. As the iteration increases, the search gradually focuses on S1. Because more S1 operations are frequently required to avoid local minima. The above results show that the proposed strategies S1 and S2 are dynamically changed during the search process.

Table 11
Comparison results achieved by NNSABC and seven other evolutionary algorithms.

Problems	CoDE Mean (SD)	JADE Mean (SD)	SaDE Mean (SD)	jDEscop Mean (SD)	CLPSO Mean (SD)	CMA-ES Mean (SD)	AABCLS Mean (SD)	NNSABC Mean (SD)
f_1	1.16E−37 (2.69E−37)	2.50E−87 (8.62E−87)	1.48E−57 (6.02E−57)	1.15E−36 (5.75E−36)	3.59E−13 (1.50E−13)	1.21E−28 (2.09E−29)	4.60E−35 (1.67E−35)	**1.85E−150 (4.53E−149)**
f_2	1.25E−34 (1.50E−34)	4.43E−78 (2.21E−77)	8.97E−55 (2.38E−54)	4.77E−36 (1.85E−35)	5.99E−11 (2.37E−11)	4.88E−23 (9.20E−24)	4.48E−33 (2.16E−33)	**9.37E−153 (2.28E−151)**
f_3	1.94E−38 (4.08E−38)	2.27E−85 (1.14E−82)	5.22E−58 (1.49E−57)	1.80E−41 (1.47E−16)	4.78E−14 (1.33E−14)	4.22E−27 (9.12E−28)	2.54E−35 (1.24E−35)	**5.68E−148 (1.39E−146)**
f_4	6.20E−143 (3.15E−142)	3.01E−100 (1.00E−99)	3.38E−80 (1.59E−79)	3.58E−124 (1.79E−123)	1.70E−65 (2.62E−65)	6.82E−13 (5.39E−13)	1.52E−50 (4.88E−50)	**9.57E−218 (0.00E+00)**
f_5	1.21E−20 (8.94E−21)	1.89E−41 (8.32E−41)	1.71E−38 (7.47E−38)	1.71E−22 (8.46E−22)	6.75E−09 (1.61E−09)	3.69E−04 (1.84E−03)	4.98E−18 (1.34E−18)	**1.56E−79 (3.42E−78)**
f_6	6.00E−05 (9.51E−05)	8.20E−10 (9.51E−10)	3.42E−01 (4.67E−01)	2.52E+00 (5.61E−01)	1.04E+01 (7.69E−01)	**6.00E−15 (7.45E−16)**	1.16E−01 (1.24E−01)	1.46E−02 (4.44E−02)
f_7	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	1.60E−01 (4.73E−01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_8	**2.67E−109 (9.67E−125)**	**2.67E−109 (9.60E−125)**	**2.67E−109 (3.71E−122)**	**2.67E−109 (9.65E−125)**	**2.67E−109 (7.60E−116)**	5.40E−66 (1.99E−65)	**2.67E−109 (2.77E−119)**	**2.67E−109 (4.49E−122)**
f_9	8.17E−03 (2.79E−03)	**2.50E−03 (1.54E−03)**	1.33E−02 (3.33E−03)	3.91E−03 (1.47E−03)	1.32E−02 (2.50E−03)	2.80E−01 (6.70E−02)	1.63E−02 (4.68E−03)	1.96E−02 (2.64E−02)
f_{10}	3.32E+01 (2.26E+01)	3.19E−01 (1.10E+00)	9.45E+00 (2.08E+01)	2.58E+01 (2.68E+01)	5.44E+01 (2.37E+01)	**1.75E−25 (4.31E−26)**	3.09E+00 (1.36E+01)	2.44E−01 (1.92E+00)
f_{11}	7.34E−01 (8.82E−01)	1.78E−11 (1.74E−11)	6.77E−01 (6.87E−01)	1.03E−14 (1.31E−14)	2.00E−04 (1.31E−04)	3.89E+02 (7.11E+01)	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
f_{12}	2.28E+01 (4.68E+00)	2.74E−08 (2.07E−08)	4.40E−01 (6.51E−01)	8.02E−02 (2.77E−01)	4.24E−03 (2.67E−03)	3.78E+02 (4.90E+01)	8.00E−02 (2.77E−01)	**0.00E+00 (0.00E+00)**
f_{13}	2.96E−04 (1.48E−03)	7.88E−04 (2.82E−03)	6.88E−03 (1.22E−02)	1.47E−16 (2.60E−16)	1.23E−09 (2.48E−09)	1.38E−03 (3.34E−03)	4.44E−18 (2.22E−17)	**0.00E+00 (0.00E+00)**
f_{14}	4.74E+00	4.75E+00	3.64E−12	6.18E+01	9.48E+00	9.01E+03	1.41E−11	**2.72E−12**

(*continued on next page*)

Table 11 (*continued*)

Problems	CoDE Mean (SD)	JADE Mean (SD)	SaDE Mean (SD)	jDEscop Mean (SD)	CLPSO Mean (SD)	CMA-ES Mean (SD)	AABCLS Mean (SD)	NNSABC Mean (SD)
	(2.37E+01)	(2.32E+01)	(0.00E+00)	(1.69E+02)	(4.74E+01)	(1.13E+03)	(3.53E−12)	**(2.58E−12)**
f_{15}	**2.81E−15**	6.22E−15	1.32E+00	1.43E+01	7.32E−06	1.99E+01	2.65E−14	4.04E−15
	(7.11E−16)	(0.00E+00)	(4.90E−01)	(6.50E+00)	(4.28E−06)	(2.63E−02)	(3.48E−15)	(1.74E−15)
f_{16}	**9.42E−33**	2.49E−03	3.24E−02	1.91E−31	1.20E−13	4.98E−03	**9.42E−33**	**9.42E−33**
	(1.40E−48)	(1.24E−02)	(9.01E−02)	(5.28E−31)	(5.37E−14)	(1.72E−02)	**(1.40E−48)**	**(1.37E−47)**
f_{17}	1.55E−33	1.01E−04	1.09E−02	5.02E−32	1.87E−13	8.01E+03	**1.50E−33**	**1.50E−33**
	(2.47E−34)	(8.05E−05)	(3.00E−02)	(5.95E−32)	(6.87E−14)	(6.25E+03)	**(0.00E+00)**	**(2.74E−47)**
f_{18}	4.39E−03	1.01E−04	1.75E−16	2.95E−05	1.13E−03	8.59E−01	5.32E−11	**9.37E−68**
	(6.44E−03)	(8.05E−05)	(2.63E−16)	(3.09E−05)	(2.87E−04)	(8.49E−01)	(9.68E−11)	**(1.75E−66)**
f_{19}	**1.35E−31**	**1.35E−31**	7.39E−02	1.30E−30	1.84E−14	3.77E−01	**1.35E−31**	**1.35E−31**
	(2.47E−33)	**(2.23E−47)**	(1.07E−01)	(3.69E−30)	(6.58E−15)	(1.22E+00)	**(2.23E−47)**	**(0.00E+00)**
f_{20}	3.45E+00	3.33E−01	4.17E−04	**0.00E+00**	2.79E−01	9.41E+00	2.56E−06	**0.00E+00**
	(2.94E−01)	(4.45E−02)	(5.94E−04)	**(0.00E+00)**	(4.51E−02)	(4.19E+00)	(7.39E−06)	**(0.00E+00)**
f_{21}	**−78.332**	−78.287	−78.31	**−78.332**	**−78.332**	−64.286	**−78.332**	**−78.332**
	(4.10E−15)	(2.26E−01)	(1.13E−01)	**(1.12E−13)**	**(7.67E−15)**	(2.63E+00)	**(4.10E−15)**	**(1.42E−14)**
f_{22}	−48.557	−49.785	−48.311	−49.983	−49.709	−40.95	**−49.998**	−49.567
	(4.45E−01)	(3.86E−02)	(2.70E−01)	(9.23E−03)	(7.12E−02)	(2.65E+00)	**(9.13E−04)**	(1.07E−01)
w/t/l	14/5/3	16/3/3	19/2/1	16/4/2	17/3/2	20/0/2	13/7/2	–

Table 12
Average ranking values of eight algorithms obtained by Friedman test.

Algorithms	Average ranking
CoDE	4.39
JADE	3.70
SaDE	5.00
jDEscop	4.16
CLPSO	5.55
CMA-ES	7.14
AABCLS	3.84
NNSABC	**2.23**

If a search strategy can generate better solutions than their corresponding parent solutions, more solutions will choose this strategy in the next iteration. By the ensemble of S1 and S2, an appropriate search strategy is chosen according to the current search status and the problem characteristics. The above results also confirm that the proposed strategies S1 and S2 can be adaptively selected to generate offspring during the search process.

Table 13
Results obtained by NNSABC with different *SN* values on the classical benchmark ($D = 30$).

Problems	$SN = 20$ Mean (SD)	$SN = 30$ Mean (SD)	$SN = 40$ Mean (SD)	$SN = 50$ Mean (SD)	$SN = 60$ Mean (SD)	$SN = 70$ Mean (SD)	$SN = 100$ Mean (SD)
f_1	**1.37E−228**	2.42E−181	1.72E−162	1.82E−144	7.44E−131	1.69E−121	4.98E−103
	(0.00E+00)	(0.00E+00)	(4.28E−161)	(4.46E−143)	(1.47E−129)	(3.24E−120)	(1.22E−101)
f_2	**3.70E−237**	4.64E−183	1.73E−162	1.59E−132	1.03E−125	5.73E−127	2.82E−104
	(0.00E+00)	(0.00E+00)	(4.28E−161)	(3.90E−131)	(2.52E−124)	(1.38E−125)	(6.91E−103)
f_3	**2.77E−228**	1.81E−197	8.28E−151	1.05E−142	1.05E−138	5.08E−129	2.21E−103
	(0.00E+00)	(0.00E+00)	(2.03E−149)	(2.56E−141)	(2.09E−137)	(1.06E−127)	(5.41E−102)
f_4	**0.00E+00**	8.13E−286	7.55E−249	5.05E−211	7.79E−198	6.87E−175	1.36E−142
	(0.00E+00)	(0.00E+00)	(0.00E+00)	(0.00E+00)	(0.00E+00)	(0.00E+00)	(3.34E−141)
f_5	**2.46E−122**	1.36E−98	9.13E−83	5.93E−72	3.00E−66	4.96E−65	8.56E−50
	(5.89E−121)	(1.91E−97)	(2.24E−81)	(1.45E−70)	(7.34E−65)	(1.18E−63)	(1.67E−48)
f_6	**5.66E−05**	4.47E−04	1.81E−03	2.25E−03	6.14E−03	5.95E−03	1.05E−02
	(2.77E−04)	(2.03E−03)	(1.85E−02)	(1.21E−02)	(2.42E−02)	(2.56E−02)	(3.85E−02)
f_7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(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**	**7.18E−66**	**7.18E−66**
	(9.07E−79)	**(7.69E−79)**	**(8.88E−79)**	**(1.08E−78)**	**(1.10E−78)**	**(1.00E−78)**	**(1.12E−78)**
f_9	1.12E−02	1.10E−02	1.20E−02	**1.08E−02**	1.20E−02	1.12E−02	1.18E−02
	(2.12E−02)	(1.79E−02)	(1.82E−02)	**(1.47E−02)**	(1.50E−02)	(1.77E−02)	(2.16E−02)
f_{10}	5.82E−01	3.40E−01	5.31E−01	4.67E−01	3.57E−01	**3.22E−01**	5.08E−01
	(1.70E+00)	(1.44E+00)	(1.57E+00)	(1.53E+00)	(1.25E+00)	**(1.23E+00)**	(1.27E+00)
f_{11}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(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**	**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)**
f_{13}	7.47E−03	2.56E−03	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(6.75E−02)	(4.12E−02)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
f_{14}	1.92E+02	8.32E+00	2.54E+01	**1.43E−13**	3.10E−13	3.36E−13	4.17E−13
	(2.39E+03)	(2.32E+02)	(4.27E+02)	**(4.92E−13)**	(2.58E−13)	(5.92E−13)	(1.48E−13)
f_{15}	8.38E−15	2.51E−15	7.85E−15	**2.19E−15**	2.28E−15	2.28E−15	2.35E−15

Table 13 (*continued*)

Problems	$SN = 20$ Mean (SD)	$SN = 30$ Mean (SD)	$SN = 40$ Mean (SD)	$SN = 50$ Mean (SD)	$SN = 60$ Mean (SD)	$SN = 70$ Mean (SD)	$SN = 100$ Mean (SD)
	(5.28E−14)	(8.59E−15)	(3.06E−14)	**(1.00E−15)**	(1.36E−15)	(1.45E−15)	(1.24E−15)
f_{16}	5.04E−05	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**	**1.57E−32**
	(7.46E−04)	**(1.37E−47)**	**(1.37E−47)**	**(1.37E−47)**	**(1.37E−47)**	**(1.37E−47)**	**(1.37E−47)**
f_{17}	4.74E−19	**1.50E−33**	**1.50E−33**	**1.50E−33**	**1.50E−33**	**1.50E−33**	**1.50E−33**
	(6.19E−18)	**(2.74E−47)**	**(2.74E−47)**	**(2.74E−47)**	**(2.74E−47)**	**(2.74E−47)**	**(1.21E−33)**
f_{18}	1.95E−23	3.43E−45	**2.64E−79**	1.28E−67	1.77E−61	1.15E−57	4.87E−44
	(4.79E−21)	(6.92E−44)	**(3.71E−78)**	(2.34E−66)	(3.62E−60)	(2.25E−56)	(8.64E−43)
f_{19}	2.91E−06	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**	**1.35E−31**
	(5.18E−05)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
f_{20}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
f_{21}	**−78.332**	**−78.332**	**−78.332**	**−78.332**	**−78.332**	**−78.332**	**−78.332**
	(1.42E−14)	**(1.42E−14)**	**(0.00E+00)**	**(4.02E−14)**	**(4.71E−14)**	**(3.76E−14)**	**(6.96E−14)**
f_{22}	−29.592	−29.602	−29.642	**−29.674**	−29.657	−29.644	−29.671
	(1.33E−01)	(9.10E−02)	(9.25E−02)	**(7.79E−02)**	(5.57E−02)	(6.63E−02)	(4.01E−02)

Table 14
Average rankings achieved by Friedman test for NNSABC with different *SN* values.

Parameter	Average ranking
$SN = 20$	4.39
$SN = 30$	3.52
$SN = 40$	3.91
$SN = 50$	**3.25**
$SN = 60$	4.07
$SN = 70$	4.07
$SN = 100$	4.80

5. Conclusion

ABC is an excellent global optimization algorithm with strong exploration ability and simple concept. However, there are some deficiencies for ABC. The exploitation ability of ABC is not as strong as its exploration ability. The original roulette selection in the onlooker bee search will gradually lose its effect with increasing of iterations. In order to overcome these problems, this paper proposes an improved ABC algorithm with the aid of machine learning. The new approach is called ABC based on modified nearest

Table 15
Comparison results on the CEC 2013 benchmark set ($D = 50$).

Problems	PABC Mean (SD)	MABC Mean (SD)	comABC Mean (SD)	EcABC Mean (SD)	mCLABC Mean (SD)	NNSABC Mean (SD)
F_1	1.36E−12 (1.49E−13)	1.60E−12 (9.18E−13)	1.06E−01 (6.65E−01)	1.59E−12 (1.84E−13)	2.27E−13 (1.45E−13)	**2.09E−13 (2.20E−13)**
F_2	1.30E+07 (9.72E+06)	2.08E+07 (6.04E+06)	5.65E+08 (4.10E+07)	5.83E+06 (3.11E+05)	**5.78E+06 (4.02E+06)**	3.75E+07 (4.78E+07)
F_3	1.47E+09 (8.25E+08)	1.08E+09 (2.13E+09)	2.90E+11 (9.53E+10)	6.31E+08 (4.35E+07)	**1.56E+08 (2.31E+08)**	5.49E+09 (1.33E+10)
F_4	1.44E+05 (4.32E+04)	1.58E+05 (1.94E+04)	7.31E+04 (8.28E+03)	**3.40E+04 (2.25E+03)**	1.15E+05 (1.89E+04)	1.06E+05 (5.39E+04)
F_5	1.25E−12 (8.31E−13)	1.47E−12 (8.23E−12)	1.11E+03 (3.64E+02)	6.06E−04 (2.30E−05)	**1.14E−13 (3.77E−13)**	1.06E−12 (2.58E−12)
F_6	4.41E+02 (3.99E+01)	4.67E+01 (7.01E+00)	2.24E+03 (7.36E+02)	3.97E+01 (9.39E−01)	4.29E+01 (5.07E−01)	**3.03E+01 (7.60E+01)**
F_7	1.64E+02 (6.56E+01)	1.50E+02 (5.16E+01)	1.48E+02 (8.05E+02)	1.86E+02 (6.22E+01)	1.06E+02 (1.46E+01)	**1.01E+02 (1.52E+01)**
F_8	2.12E+01 (9.84E+00)	2.12E+01 (2.46E+00)	**2.11E+01 (5.21E+00)**	**2.11E+01 (3.14E+00)**	2.12E+01 (3.58E−02)	**2.11E+01 (2.53E−01)**
F_9	6.01E+01 (1.06E+00)	6.16E+01 (4.87E+00)	7.79E+01 (2.23E+01)	7.14E+01 (1.44E+01)	5.66E+01 (4.05E+00)	**5.36E+01 (1.26E+01)**
F_{10}	2.34E+00 (9.28E−02)	2.71E+00 (4.92E−01)	2.66E+12 (4.58E+02)	9.71E−02 (4.46E−03)	**8.63E−02 (5.80E−02)**	1.70E+01 (2.07E+01)
F_{11}	2.27E−13 (1.98E−14)	4.70E−13 (8.46E−14)	7.76E+02 (5.73E+01)	2.86E+02 (9.63E+00)	5.68E−14 (3.04E−14)	**5.16E−14 (1.27E−13)**
F_{12}	3.84E+02 (9.51E+01)	8.26E+02 (1.74E+02)	8.84E+02 (6.48E+01)	8.33E+02 (7.35E+01)	**1.96E+02 (5.10E+01)**	4.38E+02 (2.81E+02)
F_{13}	6.86E+02 (4.22E+01)	7.04E+02 (4.18E+01)	8.05E+02 (6.12E+01)	7.56E+02 (8.01E+01)	3.02E+02 (4.45E+01)	**2.84E+02 (2.50E+02)**
F_{14}	1.17E+01 (4.17E+00)	3.41E+00 (3.91E−01)	1.06E+04 (1.18E+04)	3.73E+03 (1.07E+02)	2.12E+00 (2.31E+00)	**3.95E−01 (1.87E+00)**
F_{15}	8.88E+03 (6.01E+02)	6.49E+03 (2.56E+02)	1.40E+04 (7.85E+03)	6.50E+03 (9.17E+02)	8.24E+03 (6.90E+02)	**1.04E+03 (3.77E+02)**
F_{16}	1.97E+00 (5.39E+00)	1.52E+00 (7.92E−01)	3.65E+00 (5.62E+00)	1.20E+00 (6.02E−01)	2.19E+00 (3.44E−01)	**1.08E+00 (6.73E−01)**
F_{17}	5.15E+01 (1.71E+00)	5.10E+01 (3.07E+01)	1.03E+03 (2.03E+03)	5.98E+02 (4.08E+01)	**5.08E+01 (1.37E−02)**	**5.08E+01 (8.74E−02)**
F_{18}	5.61E+02 (4.04E+01)	7.90E+02 (2.42E+01)	1.24E+03 (8.91E+02)	1.38E+03 (4.38E+02)	**2.22E+02 (3.24E+01)**	5.04E+02 (1.47E+02)
F_{19}	2.27E+00 (1.75E−01)	9.30E+01 (3.76E+02)	5.09E+05 (3.48E+01)	2.50E+01 (5.21E+00)	**7.10E−01 (1.99E−01)**	1.78E+00 (4.87E+00)
F_{20}	8.67E+01 (8.81E+00)	2.46E+01 (1.51E+00)	2.50E+01 (3.73E+00)	2.05E+01 (6.24E+00)	2.22E+01 (8.89E−01)	**2.03E+01 (2.11E+00)**
F_{21}	**2.00E+02 (3.26E+01)**	2.02E+03 (9.87E+02)	4.51E+03 (2.84E+03)	1.12E+03 (3.67E+02)	3.88E+02 (3.28E+02)	8.37E+02 (1.74E+03)
F_{22}	6.00E+01 (1.72E+00)	2.71E+03 (6.23E+02)	1.03E+04 (7.32E+03)	2.56E+03 (8.85E+02)	4.10E+01 (3.01E+01)	**3.04E+01 (3.88E+02)**
F_{23}	1.02E+04 (4.65E+03)	8.46E+04 (6.86E+03)	1.55E+04 (5.08E+03)	9.59E+03 (2.61E+02)	**2.32E+03 (9.70E+02)**	1.07E+04 (4.94E+03)
F_{24}	3.45E+02 (1.06E+01)	3.87E+02 (2.32E+01)	4.45E+02 (4.19E+01)	3.79E+02 (1.36E+01)	3.32E+02 (1.85E+01)	**3.28E+02 (3.42E+01)**
F_{25}	3.76E+02 (8.51E+01)	4.26E+02 (1.58E+01)	4.19E+02 (1.74E+02)	3.87E+02 (9.87E+00)	3.57E+02 (5.64E+00)	**3.44E+02 (3.54E+01)**
F_{26}	2.02E+02 (3.56E+01)	2.02E+02 (3.56E+01)	5.08E+02 (3.67E+01)	2.08E+02 (1.97E+01)	**2.00E+02 (1.05E−01)**	2.01E+02 (4.15E+00)
F_{27}	1.95E+03 (6.98E+02)	2.21E+03 (1.85E+03)	2.52E+03 (2.44E+02)	1.89E+03 (8.90E+01)	1.58E+03 (4.59E+02)	**4.08E+02 (3.02E+03)**
F_{28}	**3.78E+02 (3.58E+01)**	4.11E+03 (2.81E+02)	7.63E+03 (2.86E+03)	2.12E+03 (7.79E+02)	4.00E+02 (1.16E−12)	4.00E+02 (4.76E−05)
$\boldsymbol{w/t/l}$	21/0/7	25/0/3	26/1/1	22/1/5	16/2/10	–

Table 16
Comparison results on the CEC 2013 benchmark set ($D = 100$).

Problems	cDE Mean (SD)	McDE Mean (SD)	comABC Mean (SD)	EcABC Mean (SD)	mCLABC Mean (SD)	NNSABC Mean (SD)
F_1	4.06E−02 (8.14E−03)	1.08E+05 (9.02E+04)	3.15E+00 (4.17E−01)	1.29E−02 (2.78E−03)	**2.27E−13 (4.97E−13)**	1.23E−03 (1.19E−02)
F_2	1.99E+08 (9.13E+07)	1.17E+09 (4.89E+08)	1.84E+09 (2.93E+08)	**5.62E+06 (9.57E+06)**	1.34E+07 (5.80E+06)	1.15E+07 (1.44E+07)
F_3	5.18E+10 (3.71E+10)	1.79E+16 (3.37E+16)	4.74E+14 (3.69E+13)	3.06E+09 (1.57E+09)	**1.54E+09 (1.20E+09)**	2.79E+09 (2.29E+09)
F_4	1.07E+05 (7.15E+04)	1.88E+05 (2.41E+04)	2.55E+05 (4.03E+04)	**7.72E+03 (4.85E+04)**	2.17E+05 (3.27E+04)	3.09E+04 (6.79E+03)
F_5	4.21E−02 (9.33E−02)	1.34E+04 (9.56E+03)	5.69E−01 (5.75E−01)	1.76E−02 (9.59E−01)	**5.68E−13 (8.15E−13)**	2.85E−04 (8.26E−04)
F_6	5.60E+03 (3.92E+02)	2.74E+04 (1.68E+04)	1.44E+04 (6.49E+05)	1.55E+02 (6.78E+02)	1.89E+02 (2.33E+01)	**1.13E+02 (2.04E+02)**
F_7	8.16E+04 (7.06E+03)	1.45E+04 (7.31E+03)	3.18E+06 (4.51E+06)	3.89E+05 (2.76E+05)	1.41E+02 (2.11E+01)	**1.27E+02 (4.93E+01)**
F_8	**2.13E+01 (6.94E+00)**	3.31E+01 (6.44E+00)	2.32E+01 (2.96E+00)	**2.13E+01 (8.23E+00)**	**2.13E+01 (3.32E−02)**	**2.13E+01 (1.42E−01)**
F_9	1.40E+02 (4.38E+02)	1.61E+02 (3.68E+01)	4.64E+02 (1.83E+01)	1.50E+02 (7.95E+01)	1.34E+02 (5.58E+00)	**1.31E+02 (1.54E+01)**
F_{10}	2.79E+03 (6.46E+02)	7.11E+03 (6.25E+03)	1.36E+04 (7.85E+03)	6.00E+02 (1.86E+02)	**8.44E−02 (6.58E−02)**	7.13E+01 (7.62E+01)
F_{11}	1.02E+03 (6.79E+02)	9.43E+02 (7.75E+02)	8.96E+02 (4.86E+02)	2.15E+02 (7.09E+02)	**5.68E−14 (1.32E−13)**	5.99E−05 (1.10E−03)
F_{12}	1.24E+03 (4.98E+03)	2.06E+03 (5.08E+02)	3.76E+03 (3.06E+02)	1.99E+03 (6.55E+01)	5.98E+02 (7.07E+01)	**3.51E+02 (6.21E+01)**
F_{13}	1.92E+03 (2.55E+02)	2.27E+03 (7.94E+02)	2.69E+03 (8.71E+03)	2.71E+03 (5.85E+02)	8.18E+02 (8.10E+01)	**6.00E+02 (7.94E+01)**
F_{14}	9.03E+03 (5.05E+03)	1.01E+04 (8.01E+03)	1.39E+04 (2.78E+04)	1.05E+03 (5.47E+02)	**3.53E+00 (1.59E+00)**	1.08E+01 (1.58E+01)
F_{15}	1.75E+04 (8.48E+03)	**1.13E+04 (3.52E+03)**	2.06E+04 (9.39E+03)	1.93E+04 (1.38E+02)	1.70E+04 (1.19E+03)	1.33E+04 (2.30E+03)
F_{16}	3.11E+00 (2.54E+00)	3.64E+00 (8.75E−01)	4.35E+00 (5.54E−01)	1.80E+00 (9.29E−01)	2.70E+00 (3.17E−01)	**1.49E+00 (2.94E−01)**
F_{17}	1.22E+03 (6.16E+02)	2.56E+03 (3.01E+02)	2.40E+03 (2.07E+02)	1.06E+03 (3.49E+01)	**1.02E+02 (4.06E−02)**	**1.02E+02 (3.98E+00)**
F_{18}	3.09E+03 (4.73E+03)	2.39E+03 (8.41E+02)	2.56E+03 (2.31E+03)	2.72E+03 (5.85E+02)	**5.79E+02 (7.13E+01)**	1.62E+03 (4.14E+02)
F_{19}	3.01E+02 (5.49E+01)	1.25E+05 (1.75E+04)	6.74E+05 (2.25E+04)	8.19E+01 (7.57E+00)	**1.90E+00 (3.89E−01)**	6.99E+00 (7.18E+00)
F_{20}	**5.00E+01 (0.00E+00)**	**5.00E+01 (0.00E+00)**	**5.00E+01 (0.00E+00)**	**5.00E+01 (0.00E+00)**	**5.00E+01 (0.00E+00)**	**5.00E+01 (0.00E+00)**
F_{21}	4.27E+02 (9.34E+01)	7.74E+03 (1.84E+02)	3.07E+03 (4.21E+02)	4.26E+02 (7.79E+01)	3.82E+02 (3.88E+01)	**3.61E+02 (2.85E+02)**
F_{22}	1.10E+04 (4.69E+04)	1.35E+04 (4.38E+03)	1.78E+04 (9.04E+03)	2.08E+04 (7.94E+03)	9.78E+01 (4.75E+01)	**7.53E+01 (4.51E+02)**
F_{23}	2.28E+04 (3.11E+03)	3.35E+04 (2.58E+03)	3.44E+04 (4.08E+03)	2.21E+04 (6.01E+03)	2.31E+04 (1.66E+03)	**2.12E+04 (2.62E+03)**
F_{24}	7.65E+02 (6.89E+01)	7.36E+02 (6.02E+02)	6.79E+02 (7.11E+01)	6.16E+02 (7.48E+00)	5.27E+02 (3.43E+01)	**4.76E+02 (5.00E+01)**
F_{25}	7.82E+02 (4.50E+02)	6.11E+02 (3.21E+01)	6.48E+02 (1.17E+01)	6.33E+02 (1.52E+01)	5.57E+02 (1.12E+01)	**4.89E+02 (4.82E+01)**
F_{26}	6.62E+02 (9.61E+01)	7.05E+02 (5.07E+01)	7.45E+02 (4.24E+01)	2.08E+02 (2.59E+01)	**2.01E+02 (2.72E−01)**	**2.01E+02 (3.12E−01)**
F_{27}	2.52E+03 (4.31E+02)	4.54E+03 (2.62E+02)	5.13E+03 (8.01E+02)	3.20E+03 (2.63E+02)	3.57E+03 (8.43E+02)	**1.44E+03 (8.18E+03)**
F_{28}	1.51E+04 (3.49E+03)	1.45E+04 (5.78E+03)	1.82E+04 (4.83E+03)	1.42E+04 (5.79E+03)	3.22E+03 (7.98E+02)	**3.15E+03 (5.13E+02)**
w*/*t*/*l	26/2/0	26/1/1	27/1/0	24/2/2	16/4/8	–

Table 17
Average ranking values achieved by Friedman test on the CEC 2013 benchmark set.

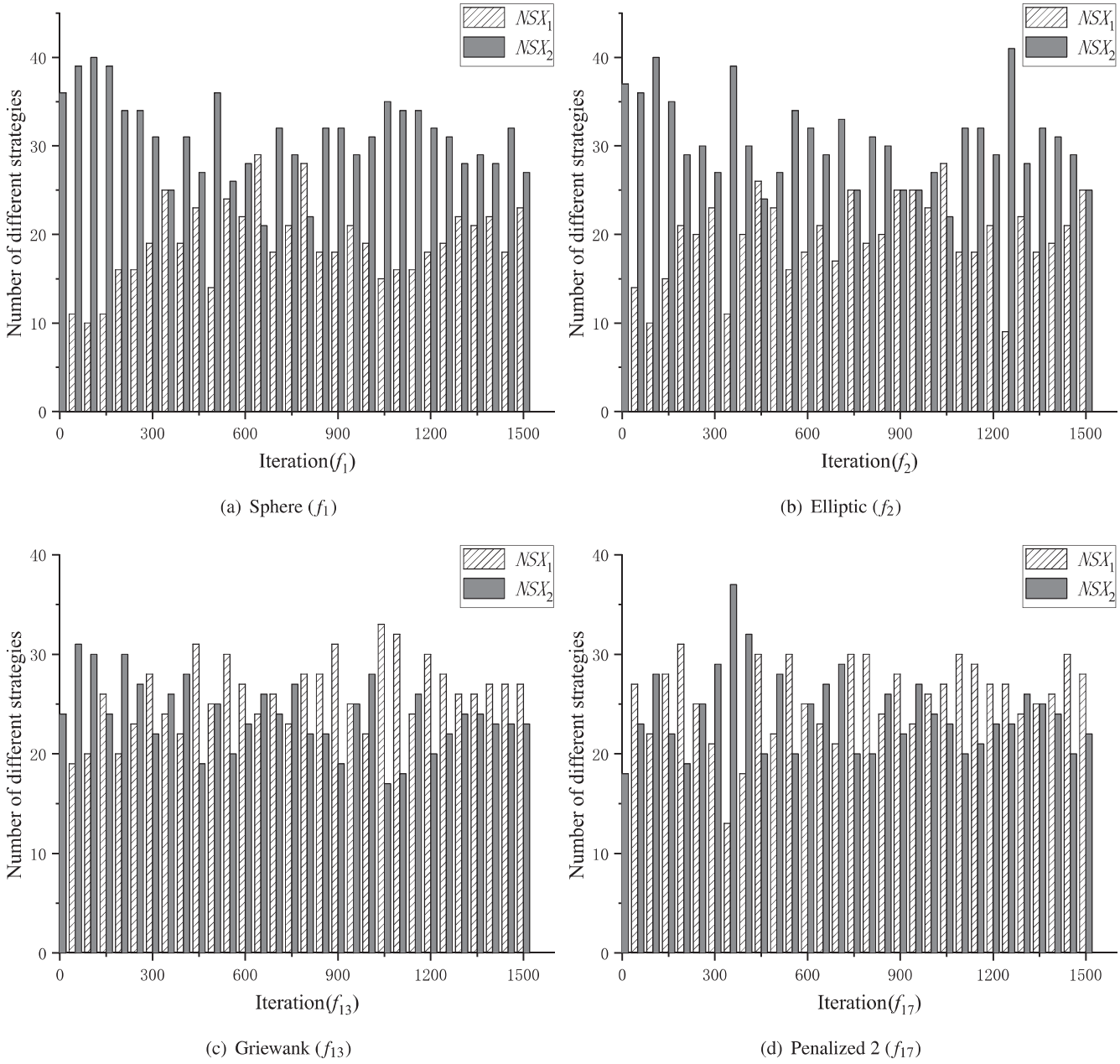
Algorithms	$D = 50$ Average ranking	Algorithms	$D = 100$ Average ranking
PABC	3.48	cDE	3.96
MABC	4.27	McDE	4.70
comABC	5.46	comABC	5.34
EcABC	3.68	EcABC	3.29
mCLABC	2.11	mCLABC	2.14
NNSABC	**2.00**	NNSABC	**1.57**

neighbor sequence (NNSABC). Firstly, a concept of modified nearest neighbor is used to construct solution sequences. Unlike the roulette selection in ABC, NNSABC randomly selects a solution from the corresponding nearest neighbor sequence to generate offspring. Moreover, two novel search strategies based on the modified nearest neighbor sequence are employed to build a strategy pool. During the evolution, different search strategies are dynamically chosen from the strategy pool according to the quality of offspring.

To verify the performance of NNSABC, it is compared with twenty-three algorithms on 22 classical problems and 28 CEC 2013 complex problems. Comparison results show NNSABC achieves competitive performance on two benchmark sets. Results also confirm the effectiveness of the proposed two search strategies. Compared with other neighbor guided ABC variants, our neighborhood strategy can obtain better performance.

From the experimental results, NNSABC obtains competitive performance on many optimization problems. These improvements can be attributed to the advantages of our approach. The proposed search strategies can greatly improve the quality of solutions. The new selection method is effective throughout the search process. However, NNSABC has some drawbacks. To find the modified nearest neighbor, the Euclidean distance between two different solutions is frequently calculated. This may cost much computational time. The updating rule for choosing search strategies is too simple to consider previous search experiences.

In this work, the concept of modified nearest neighbor sequence successfully helps ABC achieve good performance. It may be applied to other bio-inspired optimization algorithms, like monarch butterfly optimization (MBO) (Wang et al., 2019), earthworm optimization algorithm (EWA) (Wang et al., 2018), elephant herding optimization (EHO) (Wang et al., 2015), moth search (MS) (Wang, 2018) algorithm, slime mould algorithm (SMA) (Li et al., 2020), Harris hawks optimization (HHO) (Heidari et al., 2019), backtracking search optimization algorithm (BSA) (Toz, 2019; Wang et al., 2020), and fruit fly optimization algorithm (FOA) (Peng et al., 2020; Fan et al., 2021). This will be a potential direction in the future research.



(a) Sphere (f_1) (b) Elliptic (f_2)

(c) Griewank (f_{13}) (d) Penalized 2 (f_{17})

Fig. 5. The changes of strategies in the population during the search process.

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.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 62166027).

References

Akay, B., Karaboga, D., 2012. A modified artificial bee colony algorithm for real-parameter optimization. Inf. Sci. 192, 120–142.
Asghari, S., Navimipour, N.J., 2019. Cloud service composition using an inverted ant colony optimisation algorithm. Int. J. Bio-Inspired Comput. 13 (4), 257–268.
Banharnsakun, A., Achalakul, T., Sirinaovakul, B., 2011. The best-so-far selection in artificial bee colony algorithm. Appl. Soft Comput. 11 (2), 2888–2901.
Banitalebi, A., Aziz, M.I.A., Bahar, A., Aziz, Z.A., 2015. Enhanced compact artificial bee colony. Inf. Sci. 298, 491–511.
Brest, J., Maucec, M.S., 2011. Self-adptive differential evolution algorithm using population size reduction and three strategies. Soft. Comput. 15, 2157–2174.
Cai, X.J., Geng, S.J., Zhang, J.B., Wu, D., Cui, Z.H., Zhang, W.S., Chen, J.J., 2020. A sharding scheme based many-objective optimization algorithm for enhancing security in blockchain-enabled industrial internet of things. IEEE Trans. Industr. Inf. https://doi.org/10.1109/TII.2021.3051607. to be published.
Cui, L.Z., Li, G.H., Lin, Q.Z., Du, Z.H., Gao, W.F., Chen, J.Y., Lu, N., 2016. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. Inf. Sci. 367 (368), 1012–1044.
Cui, L.Z., Li, G.H., Zhu, Z.X., Lin, Q.Z., Wen, Z.K., Lu, N., Wong, K.C., Chen, J.Y., 2017. A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. Inf. Sci. 414, 53–67.
Cui, Z.H., Zhang, J.J., Wang, Y.C., Cao, Y., Cai, X.J., Zhang, W.S., Chen, J.J., 2019. A pigeon-inspired optimization algorithm for many-objective optimization problems. Sci. China 62 (7), 70212.
Dao, T.K., Chu, S.C., Shieh, C.S., Horng, M.F., 2014. Compact artificial bee colony. In: Modern 316 Advances in Applied Intelligence. Springer, pp. 96–105.

Das, S., Abraham, A., Chakraborty, U., Konar, A., 2009. Differential evolution using a neighborhood-based mutation operator. IEEE Trans. Evol. Comput. 13 (3), 526–553.
Fan, Y., Wang, P.J., Mafarja, M., Wang, M.J., 2021. A bioinformatic variant fruit fly optimizer for tackling optimization problems. Knowl.-Based Syst. 213, (15) 106704.
Gao, W.F., Liu, S.Y., 2011. Improved artificial bee colony algorithm for global optimization. Inf. Process. Lett. 111 (17), 871–882.
Gao, W.F., Liu, S.Y., 2012. A modified artificial bee colony algorithm. Comput. Oper. Res. 39 (3), 687–697.
Gao, W.F., Liu, S.Y., Huang, L.L., 2013. A novel artificial bee colony algorithm with powell's method. Appl. Soft Comput. 13 (9), 3763–3775.
Gao, W.F., Chan, F.T.S., Huang, L.L., Liu, S.Y., 2015. Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. Inf. Sci. 316, 180–200.
Gao, W.F., Huang, L.L., Liu, S.Y., Chan, F.T.S., Dai, C., Shan, X., 2015. Artificial bee colony algorithm with multiple search strategies. Appl. Math. Comput. 271 (15), 269–287.
Hansen, N., Ostermeier, A., 2001. Completely derandomized self-adaptation in evolution strategies. Evol. Comput. 9 (2), 159–195.
Harfouchi, F., Habbi, H., Ozturk, C., Karaboga, D., 2018. Modified multiple search cooperative foraging strategy for improved artificial bee colony optimization with robustness analysis. Soft. Comput. 22 (19), 6371–6394.
Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H.L., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872.
Jadon, S.S., Bansal, J.C., Tiwari, R., Sharma, H., 2015. Accelerating artificial bee colony algorithm with adaptive local search. Memetic Comput. 7, 215–230.
Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization, Erciyes University, Engineering Faculty. Computer engineering Department, Technical Report-tr06.
Karaboga, D., Gorkemli, B., 2014. A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl. Soft Comput. 23, 227–238.
Kiran, M.S., Hakli, H., Gunduz, M., Uguz, H., 2015. Artificial bee colony algorithm with variable search strategy for continuous optimization. Inf. Sci. 300 (10), 140–157.
Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S., 2006. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans. Evol. Comput. 10 (3), 281–295.
Li, S., Chen, H.L., Wang, M., Heidari, A.A., Mirjalili, S., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Gener. Comput. Syst. 111, 300–323.
Lin, Q., Zhu, M., Li, G., 2017. A novel artificial bee colony algorithm with local and global information interaction. Appl. Soft Comput. 62, 702–735.
Liu, F.X., Sun, Y.H., Wang, G.G., Wu, T.T., 2018. An artificial bee colony algorithm based on dynamic penalty and lévy flight for constrained optimization problems. Arab. J. Sci. Eng. 43 (12), 7189–7208.
Liu, N.X., Pan, J.S., Sun, C.L., Chu, S.C., 2020. An efficient surrogate-assisted quasi-affine transformation evolutionary algorithm for expensive optimization problems. Knowl.-Based Syst. 209, (17) 106418.
Manoj, V.J., Elias, E., 2012. Artificial bee colony algorithm for the design of multiplier-less nonuniform filter bank transmultiplexer. Inf. Sci. 192, 193–203.
Mininno, E., Neri, F., Cupertino, F., Naso, D., 2011. Compact differential evolution. IEEE Trans. Evol. Comput. 15 (1), 203–219.
Neri, F., Mininno, E., 2010. Memetic differential evolution for cartesian robot control 5 (2), 54–65.
Peng, H., Deng, C., Wu, Z., 2019. Best neighbor guided artificial bee colony algorithm for continuous optimization problems. Soft. Comput. 23 (18), 8723–8740.
Peng, L., Zhu, Q., Lv, S.X., Wang, L., 2020. Effective long short-term memory with fruit fly optimization algorithm for time series forecasting. Soft. Comput. 24 (19), 15059–15079.
Qin, A.K., Huang, V.L., Suganthan, P.N., 2008. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. 12 (1), 64–79.
Saad, A., Khan, S.A., Mahmood, A., 2018. A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. Swarm Evol. Comput. 38, 187–201.
Toz, G., Yücedağ, İ., ErdoğMuş, P., 2019. A fuzzy image clustering method based on an improved backtracking search optimization algorithm with an inertia weight parameterz. J. King Saud Univ. 31(3), 295–303..
Wang, G.G., 2018. Moth search algorithm: A bio-inspired Metaheuristic algorithm for global optimization problems. Memetic Comput. 10 (2), 151–164.
Wang, H.Q., Yi, J.H., 2017. An improved optimization method based on krill herd and artificial bee colony with information exchange. Memetic Comput. 10 (2), 1–22.
Wang, Y., Cai, Z.X., Zhang, Q.F., 2011. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. 15 (1), 55–66.
Wang, H., Sun, H., Li, C., Rahnamayan, S., Pan, J.S., 2013. Diversity enhanced particle swarm optimization with neighborhood search. Inf. Sci. 223, 119–135.
Wang, H., Rahnamayan, S., Sun, H., Omran, M.G.H., 2013. Gaussian bare-bones differential evolution. IEEE Ninth International Conference on Natural Computation 43 (2), 634–647.
Wang, H., Wu, Z., Rahnamayan, S., Sun, H., Liu, Y., Pan, J., 2014. Multi-strategy ensemble artificial bee colony algorithm. Inf. Sci. 279 (20), 587–603.
Wang, G.G., Deb, S., Coelho, L.D.S., 2015. Elephant herding optimization. In: the 3rd International Symposium on Computational and Business Intelligence, pp. 1–5.
Wang, F., Zhang, H., Li, Y.X., Zhao, Y.Y., Rao, Q., 2018. External archive matching strategy for MOEA/D. Soft. Comput. 22, 7833–7846.
Wang, G.G., Deb, S., Coelho, L.D.S., 2018. Earthworm optimisation algorithm: A bio-inspired Metaheuristic algorithm for global optimisation problems. Int. J. Bio-inspired Comput. 12 (1), 1–22.
Wang, F., Li, Y.X., Zhang, H., Hu, T., Shen, X.L., 2019. An adaptive weight vector guided evolutionary algorithm for preference-based multi-objective optimization. Swarm Evol. Comput. 49, 220–233.
Wang, G.G., Deb, S., Cui, Z., 2019. Monarch butterfly optimization. Neural Comput. Appl. 31 (7), 1995–2014.
Wang, F., Li, Y.X., Zhou, A., Tang, K., 2020. An estimation of distribution algorithm for mixed-variable newsvendor problems. IEEE Trans. Evol. Comput. 24 (3), 479–493.
Wang, F., Li, Y.X., Liao, F., Yan, H.Y., 2020. An ensemble learning based prediction strategy for dynamic multi-objective optimization. Appl. Soft Comput. 96, 106592.
Wang, H., Wang, W., Xiao, S., Cui, Z., Xu, M., Zhou, X., 2020. Improving artificial bee colony algorithm using a new neighborhood selection mechanism. Inf. Sci. 527, 227–240.
Wang, L., Peng, L., Wang, S., Liu, S., 2020. Advanced backtracking search optimization algorithm for a new joint replenishment problem under trade credit with grouping constraint. Appl. Soft Comput. 86, 105953.
Wang, F., Zhang, H., Zhou, A., 2021. A particle swarm optimization algorithm for mixed-variable optimization problems. Swarm Evol. Comput. 60, (2) 100808.
Xiao, S.Y., Wang, W.J., Wang, H., Tan, D.K., Wang, Y., Yu, X., Wu, R.X., 2019. An improved artificial bee colony algorithm based on elite strategy and dimension learning. Mathematics 7 (3), 289.
Xiao, S., Wang, H., Wang, W., Huang, Z., Zhou, X., Xu, M., 2021. Artificial bee colony algorithm based on adaptive neighborhood search and gaussian perturbation. Appl. Soft Comput. 100, 106955.
Yeh, W.C., Hsieh, T.J., 2011. Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Comput. Oper. Res. 38 (11), 1465–1473.
Yildiz, A.R., 2013. Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. Inf. Sci. 220, 339–407.
Zhang, J.Q., Sanderson, A.C., 2009. JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. 13 (5), 945–958.
Zhang, X., Li, X., Yin, M., 2020. An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem. Int. J. Bio-Inspired Comput. 15 (2), 113–124.
Zhou, X.Y., Wu, Y.L., Zhong, M.S., Wang, M.W., 2021. Artificial bee colony algorithm based on multiple neighborhood topologies. Appl. Soft Comput. 111, 107697.
Zhu, G., Kwong, S., 2010. Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl. Math. Comput. 217 (7), 3166–3173.