Artificial bee colony algorithm based on knowledge fusion

Hui Wang¹ · Wenjun Wang² · Xinyu Zhou³ · Jia Zhao¹ · Yun Wang¹ · Songyi Xiao¹ · Minyang Xu¹

Received: 13 March 2020 / Accepted: 17 June 2020 / Published online: 4 July 2020
© The Author(s) 2020

Abstract

Artificial bee colony (ABC) algorithm is one of the branches of swarm intelligence. Several studies proved that the original ABC has powerful exploration and weak exploitation capabilities. Therefore, balancing exploration and exploitation is critical for ABC. Incorporating knowledge in intelligent optimization algorithms is important to enhance the optimization capability. In view of this, a novel ABC based on knowledge fusion (KFABC) is proposed. In KFABC, three kinds of knowledge are chosen. For each kind of knowledge, the corresponding utilization method is designed. By sensing the search status, a learning mechanism is proposed to adaptively select appropriate knowledge. Thirty-two benchmark problems are used to validate the optimization capability of KFABC. Results show that KFABC outperforms nine ABC and three differential evolution algorithms.

Keywords Artificial bee colony (ABC) · Knowledge fusion · Exploration and exploitation · Opposition-based learning · Optimization

Introduction

Many optimization problems appear in real-world industry and service systems. Traditional optimization methods encounter some difficulties in solving those problems because of various complexities. To deal with complex prob-

lems, several intelligent optimization methods have been proposed, such as ant colony optimization (ACO) [1], particle swarm optimization (PSO) [2,3], artificial bee colony (ABC) [4,5], firefly algorithm (FA) [6,7], grey wolf optimizer (GWO) [8], estimation of distribution algorithm (EDA) [9], bat algorithm (BA) [10], krill herd algorithm (KHA) [11], and monarch butterfly optimization (MBO) [12].

As a branch of swarm intelligence, the optimization process of ABC is similar to the foraging behaviors of bees [13]. To find nectar sources (food sources), different kinds of bees work together in a cooperative manner and each kind of bee has its own responsibility. Due to its powerful search capability, ABC is widely used to solve various complex problems, such as data clustering [14], image segmentation [15], network planning [16], and numerical optimization [17,18]. However, several studies claimed that ABC cannot effectively balance the exploration and exploitation search [19,20]. The main reason is that ABC has strong exploration and weak exploitation capabilities. To address this issue, some excellent strategies were proposed [17,19,21].

Knowledge is the induction and summary of human understanding of the laws of various things in the objective world. Incorporating knowledge in intelligent optimization algorithms is important to strengthen the optimization capability. In view of this, a novel ABC algorithm based on knowl-

✉ Hui Wang
huiwang@nit.edu.cn

✉ Wenjun Wang
wangwenjun@nit.edu.cn

Xinyu Zhou
xyzhou@jxnu.edu.cn

Jia Zhao
zhaojia925@163.com

Yun Wang
wangyun@nit.edu.cn

Songyi Xiao
speaknow@126.com

Minyang Xu
ncxmy2019@163.com

¹ School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China

² School of Business Administration, Nanchang Institute of Technology, Nanchang 330099, China

³ College of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China

edge fusion (called KFABC) is proposed in this paper. To construct KFABC, three kinds of knowledge are selected. For each kind of knowledge, the corresponding utilization method is designed. By sensing the search status, a learning mechanism is proposed to dynamically choose appropriate knowledge. To verify KFABC, 32 benchmark problems are used. Computational results demonstrate that KFABC can effectively enhance the optimization capability and achieve promising performance compared with other ABC and DE algorithms.

The rest of this work is organized as follows: The original ABC and its recent work are introduced in Sect. 2 and the proposed KFABC is described in Sect. 3. In Sect. 4, computational results and discussions are presented. The work is concluded in Sect. 5.

Artificial bee colony

Descriptions of ABC

As mentioned before, the search manner of ABC is similar to the foraging behaviors of bees. Food sources scattered in nature are called solutions in the solution space. In the swarm, bees consist of employed bees, onlooker bees, and scout bees [13]. The employed bees fly around the current solutions and try to find some better ones. All bees share the search experiences with each other. Some better solutions are chosen by the onlooker bees and they are conducted on further search. The scout bees observe the changes of all solutions in the swarm. The scout bees produce a random solution to replace the unchanged solution in several iterations.

Initialization Supposing that the initial swarm has SN solutions and SN is the swarm size. Each initial solution X_i is randomly produced as follows [13]:

$$x_{i,j} = \text{low}_j + \text{rand} \cdot (\text{up}_j - \text{low}_j), \\ i = 1, 2, \dots, SN, \quad j = 1, 2, \dots, D, \quad (1)$$

where rand is a random value between 0 and 1, D is the size of dimension, and $[\text{low}_j, \text{up}_j]$ is the boundary constraint.

Search phase of employed bees At this stage, the employed bees search around each solution X_i and find a new one V_i [13]:

$$v_{i,jr} = x_{i,jr} + \phi_{i,jr} \cdot (x_{i,jr} - x_{k,jr}), \quad i = 1, 2, \dots, SN, \quad (2)$$

where X_k is taken from the swarm ($X_k \neq X_i$) randomly, and $jr \in [1, D]$ is a random integer. The weight $\phi_{i,jr}$ is randomly generated between -1 and 1. As seen, Eq. (2) only modifies the jr th dimension of X_i . For the rest of dimensions, both V_i and X_i have the same values. When V_i is superior to its parent X_i , X_i is replaced by V_i .

Search phase of onlooker bees At this stage, the employed bees complete the neighborhood search. Compared with the employed bees, the operation of the onlooker bees is different. They do not conduct the search on all solutions but on some better ones. According to the fitness proportional selection (FPS) method [22], each X_i has a selection probability p_i . Some better solutions are selected in terms of p_i and the onlooker bees conduct further search on them. The probability p_i is computed by [13]:

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

$$\text{fit}(X_i) = \begin{cases} \frac{1}{1+f(X_i)}, & \text{if } f(X_i) \geq 0 \\ 1 + |f(X_i)|, & \text{otherwise} \end{cases}, \quad (4)$$

where $\text{fit}(X_i)$ and $f(X_i)$ are the fitness value and function value of X_i , respectively.

For each solution X_i in the current swarm, when its corresponding selection probability p_i is satisfied, the onlooker bees use Eq. (2) to produce offspring V_i . Like the employed bees, the onlooker bees also use the greedy selection to compare the quality of V_i and X_i . The better one between them is used as the new X_i [13].

Search phase of scout bees At each iteration, the scout bees use a counter to monitor the changes of each solution in the swarm. If the solution X_i cannot be replaced by V_i , the counter trial_i is added by 1; otherwise, trial_i is equal to 0. When trial_i exceeds a preset value limit , the corresponding solution X_i is re-initialized by Eq. (1) [13].

Brief review of recent work on ABC

Recently, ABC was applied to various complex problems [23]. Many kinds of ABC variants were proposed to enhance the search performance. A short review of ABC is presented in this section.

To improve the exploitation capability, a global best ABC (GABC) was proposed [19], in which the search strategy is modified by using the global best solution X_{best} . In [24], another search strategy based on X_{best} was designed. In [25], different strategies based on X_{best} can be self-adaptively chosen. Cui et al. [20] designed two elite guided search strategies, in which some top best solutions are utilized to lead the search. In [26], the best neighbor was introduced to modify the search strategy.

Some studies showed that two or more search strategies are beneficial for the search [27,28]. Wang et al. [27] used three search strategies and each solution can dynamically choose a strategy. Similarly, the literature [29] also employed three search strategies. In [30], three search strategies were designed based on Gaussian distribution. Each solution in the swarm can choose suitable search strategy in terms of an

adaptive probability mechanism. Kiran et al. [28] employed five search strategies. Each strategy has a selection probability and its selection is based on a roulette-wheel selection method. In [31], two strategy pools were built for different search phases. Each strategy pool contains three different search strategies.

Saad et al. [32] presented a multi-objective ABC for network topology of computer communication, in which five optimization objectives including reliability, availability, mean link utilization, cost, and delay are simultaneously considered. Based on some genetic operations, Ozturk et al. [33] proposed a binary ABC. At the initial stage, each dimension of a solution is randomly assigned 0 or 1. The approach is tested on three types of problems including image clustering, Knapsack problem, and numerical optimization. Using some top best solutions in the population, Bajer and Zorić [34] proposed a modified search strategy. By combining X_{best} and two randomly chosen solutions, new solutions are produced in the search phase of scout bees.

Proposed approach

The proposed KFABC aims to balance the exploration and exploitation search by incorporating knowledge in the original ABC. To construct the approach, three crucial questions should be addressed. First, what kind of knowledge can be utilized? Second, how do we make use of knowledge to enhance the optimization? Third, how do we design effective learning mechanisms based on knowledge?

Knowledge representation

According to the attributes of optimization problems, they are divided into two categories: unimodal and multimodal. For unimodal problems, good exploitation capability can accelerate the search and find more accurate solutions. For multimodal problems, good exploration capability can avoid falling local minima. In general, the search features (exploitation or exploration) are determined by search strategies (iterative equations or models). Different search strategies may have different search features. For example, the search strategy of the original ABC prefers exploration. In [27], the search strategy (modified ABC/best) prefers exploitation.

When solving an optimization problem, an idea ABC should effectively keep balance the exploration and exploitation search. At the initial search stage, ABC should prefer the exploration search. This is helpful to expand the search area and cover the global optimum as much as possible. As the iteration increases, ABC should switch from the exploration search to the exploitation search. This is beneficial for fine search and accelerate the convergence speed. When solutions are stagnant, the current search should switch from

the exploitation search to the exploration search. It may help the trapped solutions to escape from local minima. From the above analysis, the features of search strategies can be used as the first kind of knowledge. By combining multiple search strategies with different features (knowledge) effectively, ABC can obtain a better optimization capability.

In the original ABC, employed bees search around all solutions in the swarm and generate offspring. Then, onlookers select some better solutions from the swarm in terms of fitness proportional selection (FPS) [22]. These chosen solutions can be called elite solutions. The onlooker bees search around those elite solutions and produce offspring. Therefore, the purpose of the onlooker bees is to accelerate the search.

However, the onlooker bees cannot use FPS (Eq. (3)) to choose elite solutions in many cases. Assume that X_c and X_d are two different solutions and their objective function values are $1.0\text{E}-30$ and $1.0\text{E}-20$, respectively. Then, their fitness values are $1/(1+1.0\text{E}-30)=1$ and $1/(1+1.0\text{E}-20)=1$, respectively. It is apparent that two different solutions have the same fitness value. The fitness value cannot distinguish X_c and X_d , even if X_c is actually better than X_d . Based on FPS, better and worse solutions may have the same selection probability. Consequently, the FPS does not work. In essence, the search operation of onlooker bees aims to use better (elite) solutions to guide the search and find more accurate solutions. Based on this point of view, some elite solutions can be used as the second kind of knowledge to lead the search.

When a solution stay in a position in several iterations, the solution is considered to be trapped. Then, a scout bee abandons the trapped solution and employs a random solution to substitute for it. Though the abandoned solution is worse than other ones in the swarm, it releases a signal that the current swarm runs the risk of falling into local minima. However, the efficiency of the random initialization method in ABC is low, because it easily expands the search area and slows down the convergence. How to help abandoned solutions jump out of local optima is important for the convergence of ABC. Therefore, trapped solutions are used as the third kind of knowledge.

Based on the above analysis, three kinds of knowledge including different features of multiple search strategies, elite solutions, and abandoned solutions, are used in our approach. Though multiple search strategies and elite solutions were used in some modified ABCs [20,27,28,30,31], our proposed KFABC employs a different method to incorporate this knowledge to obtain good optimization capability.

Knowledge utilization

As mentioned before, two types of knowledge are utilized in our approach. How to make good use of this knowledge is discussed in this section. The first kind of knowledge focuses on

the different features of search strategies. Two search strategies including Eqs. (5) and (6) are employed [27,35]. The first search strategy Eq. (5) prefers exploration, and the second search strategy Eq. (6) is good at exploitation. Those two strategies are sufficient to help ABC deal with different kinds of optimization problems. The involved search strategies are described as follows [27,35]:

$$v_{i,jr} = x_{r1,jr} + \phi_{i,jr} \cdot (x_{r1,jr} - x_{r2,jr}), \quad (5)$$

$$v_{i,jr} = x_{\text{best},jr} + \phi_{i,jr} \cdot (x_{\text{best},jr} - x_{k,jr}), \quad (6)$$

where X_{r1} , X_{r2} , and X_k are selected from the swarm randomly ($r1 \neq r2 \neq i$ and $k \neq i$), X_{best} is the global best solution, $jr \in [1, D]$ is a random integer, and $\phi_{i,jr} \in [-1, 1]$ is a random value.

For the employed bee search stage, there are two alternative search strategies (exploration search Eq. (5) and exploitation search Eq. (6)). The initial search is assigned exploration described in Eq. (5). When the external search environment changes, employed bees should make appropriate response. In Sect. 3.3, a learning mechanism is designed to sense the current search status. For example, if the current exploration search is inappropriate, the employed bees choose the exploitation search to generate offspring.

The second kind of knowledge is elite solutions. In ABC, some better solutions are chosen in terms of FPS and they are conducted on further search. However, the FPS does not work in some cases. Therefore, the onlooker bees directly conduct further search on the elite solutions. According to the suggestions of [20], the top best 100% solutions in the current swarm are called elite solutions and $\rho \in (0, 1.0]$. Unlike the employed bee search stage, the onlooker bees only use the exploitation search strategy Eq. (6) [27].

The third kind of knowledge is abandoned solutions. When the scouts detect a abandoned solution, the whole swarm will run the risk of falling into local minima. To avoid this case, the original ABC employs a random initialization method to create a new solution, replace the trapped one. Though the random solution may easily jump out of the local minima, it can easily expand the search area and slow down the convergence speed. In our approach, opposition-based learning (OBL) and Cauchy disturbance are used to produce two new solutions, respectively [36,37]. Besides the random solution, there are two another candidate solutions. Then, the best solution among them is chosen to replace the abandoned one.

Supposing that X_a is the abandoned solution. A random solution RX is produced as below:

$$rx_j = \text{low}_j + \text{rand}(0, 1) \cdot (\text{up}_j - \text{low}_j), \quad (7)$$

where $j = 1, 2, \dots, D$, and $[\text{low}_j, \text{up}_j]$ is the boundary constraint.

OBL was developed by Tizhoosh [36], and it was applied to strengthen the search capability of many intelligent optimization algorithms [37–40]. For the abandoned solution X_a , its opposite solution OX is produced as below [36]:

$$ox_j = x_j^{\min} + x_j^{\max} - x_{a,j}, \quad (8)$$

where $j = 1, 2, \dots, D$, and $[x_j^{\min}, x_j^{\max}]$ is the boundary of the current swarm.

Many references proved that Cauchy disturbance could help trapped solution to escape from local minima [39–41]. For the abandoned solution X_a , a new solution CX is generated in the neighborhood of X_a [37]:

$$cx_j = x_{a,j} + \text{cauchy}(), \quad (9)$$

where $j = 1, 2, \dots, D$, and $\text{cauchy}()$ is a random value based on the Cauchy distribution.

Among RX, OX, and CX, the best one is selected to replace the abandoned solution X_a . Three solutions have quite different features. The random initialization can directly help abandoned solutions jump out of the local optima. The OBL provides a large probability to search a better solution. The Cauchy disturbance is helpful to find more accurate solutions around the abandoned solution.

Learning mechanism

According to the attribute of a problem, choosing an appropriate search strategy (exploration or exploitation) can help ABC to obtain good performance. In many cases, the fixed search strategy is not suitable. The algorithm should automatically select the exploration or exploitation search during the search process. How to make a change on the search strategy is determined by the search status of the swarm. In our approach, a learning mechanism is proposed to adaptively change the search strategy.

Based on iterative search, ABC gradually converges. It means that all solutions in the swarm move towards the global or local optimum. The average function value (AFV) of all solutions in the swarm is also approaching to the optimum. By observing the changes of AFV, the search status of the current swarm is obtained. Supposing that $\text{AFV}(t)$ is the average function value at the t th iteration, and it is defined by:

$$\text{AFV}(t) = \frac{f(X_i(t))}{\sum_{i=1}^{SN} f(X_i(t))}, \quad (10)$$

where $f(X_i(t))$ is the i th solution at t th iteration.

Based on AFV, a new indicator (called IRAFV) is defined to measure the improvement rate of the average function value:

Fig. 1 The improvement rate of the average function value (IRARV)

$$IRAFV(t) = \frac{\Delta AFV(t)}{\Delta t}, \quad (11)$$

where $\Delta AFV(t) = |AFV(t) - AFV(t-1)|$ and $\Delta t = |t - (t-1)| = 1$.

Figure 1 clearly illustrates how to calculate the IRAFV. As seen, the definition of the IRAFV is the tangent function $\tan \theta$. Initially, the employed bees use the exploration search strategy (Eq. (5)). At each iteration, the indicator IRAFV is computed in terms of Eq. (11). Then, the current $IRAFV(t)$ is compared with the last $IRAFV(t-1)$. If $IRAFV(t) > IRAFV(t-1)$, it means that the improved rate of AFV in the current iteration is larger than the one in the last iteration. The current exploration search can continuously find much better solutions. Therefore, the employed bees still use the current exploration search pattern. If $IRAFV(t) \leq IRAFV(t-1)$, it means that the improved rate of AFV is decreased with the growth of iterations. To enhance the improved rate of AFV, the employed bees switch from the exploration search (Eq. (5)) to the exploitation search (Eq. (6)).

Framework of KFABC

To construct KFABC, three kinds of knowledge including different features of multiple search strategies, elite solutions, and abandoned solutions are used. The first kind of knowledge determines the search feature exploration or exploitation. The second kind of knowledge helps the onlooker bees avoid using FPS to choose better solutions. The third kind of knowledge implies that the current swarm runs the risk of falling into local minima. To make full use of the above knowledge, different methods are designed. In addition,

a learning mechanism is designed to adaptively change the exploration or exploitation search.

The framework of KFABC is shown in Algorithm 1, where *flag* represents the selection of strategy, *FES* is the number of function evaluations, and *MaxFES* is the maximum value of *FES*. In line 3, *flag* = 0 means that the initial search strategy is the exploration search (Eq. (5)). The value of *flag* is updated in lines 41 and 44. *flag* = 1 indicates the exploitation search (Eq. (6)) which is used for the employed bees in the next iteration. The value of *flag* does not affect the onlooker bees. In the whole search process, the onlooker bees have been using the exploitation search based on elite solutions. For abandoned solutions, OBL and Cauchy disturbance are used besides the original random initialization method. This is beneficial for reducing the risk of trapping in local optima.

Experimental study

Test problems

To evaluate the performance of KFABC, two different benchmark sets with 32 test problems are utilized. The first benchmark set contains 22 test problems, and they are briefly described in Table 1. Their detailed mathematical definitions can be found in [20]. In the test, *D* is set to 30 and 100. The global minimum of each problem is given in the last column of Table 1. The second benchmark set consists of ten test problems, which are selected from the CEC 2013 benchmark set [42].

Algorithm 1: Framework of KFABC

```

1 Randomly produce  $SN$  initial solutions;
2 Calculate the function value of  $X_i, i = 1, 2, \dots, SN$ ;
3 Set  $FES = SN, flag = 0, t = 0$ , and  $trial_i = 0, i = 1, 2, \dots, SN$ ;
4 while  $FES \leq MaxFES$  do
    /* Search phase of employed bees */
5   for  $i = 1$  to  $SN$  do
        /* Exploration search */
6       if  $flag == 0$  then
7           Generate  $V_i$  by Eq. (5);
8       end
        /* Exploitation search */
9       if  $flag == 1$  then
10          Generate  $V_i$  by Eq. (6);
11       end
12       Compute the function value of  $V_i$  and set  $FES = FES + 1$ ;
13       if  $f(V_i) < f(X_i)$  then
14          Replace  $X_i$  with  $V_i$  and set  $trial_i = 0$ ;
15       end
16       else
17           $trial_i = trial_i + 1$ 
18       end
19   end
    /* Search phase of onlooker bees */
20   for  $i = 1$  to  $SN$  do
        /* Exploitation search */
21       Randomly select a solution from the elite set as  $X_i$ ;
22       Generate  $V_i$  by Eq. (6);
23       Compute the function value of  $V_i$  and set  $FES = FES + 1$ ;
24       if  $f(V_i) < f(X_i)$  then
25          Replace  $X_i$  with  $V_i$  and set  $trial_i = 0$ ;
26       end
27       else
28           $trial_i = trial_i + 1$ 
29       end
30   end
    /* Search phase of scout bees */
31   if  $max\{trial_i\} > limit$  then
32       Generate three solutions  $RX, OX$ , and  $CX$  by Eqs. (7), (8), and (9), respectively;
33       Calculate the function values of  $RX, OX$ , and  $CX$ ;
34        $FES = FES + 3$ ;
35       Select the best one from  $RX, OX$ , and  $CX$  to replace  $X_i$ ;
36   end
    /* Learning mechanism */
37   Calculate the  $IRAFV(t)$  and  $IRAFV(t - 1)$  according to Eq. (11);
38   if  $IRAFV(t) > IRAFV(t - 1)$  then
39       /* Change to the exploration search */
40        $flag = 0$ ;
41   end
42   else
43       /* Change to the exploitation search */
44        $flag = 1$ ;
45   end
46    $t = t + 1$ ;
47 end
48 Output the best solution;

```

Table 1 Twenty-two test problems

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

have the same selection probability $1/SN$ on average. Some better solutions will not have larger selection probability. It deviates from the idea of the original ABC. Then, different ρ values are tested to investigate the parameter ρ in this section.

In the experiments, ρ is set to 0.1, 0.2, 0.3, 0.5, and 1.0, respectively. For other parameters, $SN = 50$, $D = 30$, $MaxFES = 5000 \cdot D$, and $limit = 100$ are used. For each problem, KFABC with each ρ is run 30 trials. Results of KFABC with different ρ values are presented in Table 2. As seen, a smaller ρ can help KFABC obtain better results for f_1 – f_3 and f_5 , while a larger ρ is better for f_4 . The performance of KFABC is not influenced by ρ on 11 problems (f_7 , f_8 , f_{11} – f_{14} , f_{16} , f_{17} , and f_{19} – f_{21}). For f_6 , f_9 , f_{10} , f_{15} , and f_{18} , $\rho = 0.1$ achieves the best results among five different settings. $\rho = 0.1, 0.3$, and 0.5 outperform $\rho = 0.2$ and 1.0 .

From the above results, the parameter ρ cannot seriously influence the performance of KFABC. To select the best setting of ρ , Friedman test is utilized to obtain the mean ranking of KFABC with different ρ values [43,44]. The mean ranking results are shown in Table 3. As seen, $\rho = 0.1$ obtains the best ranking. It means that $\rho = 0.1$ can help KFABC achieve a better performance than other ρ values. Thus, $\rho = 0.1$ is considered as the best setting.

Investigation of the parameter ρ

In the search phase of the onlooker bees, the top best $100\rho\%$ solutions in the swarm are chosen as elite solutions and $\rho \in (0, 1.0]$. By controlling the parameter ρ , the size of elite solutions is adjusted. For an extreme case, ρ is set to the minimum value. $100\rho\% \cdot SN = 1$ and the elite set has only one solution X_{best} . Then, the onlooker bees will search around X_{best} SN times. For another extreme case, ρ is set to the maximum value. $\rho = 1.0$ and the elite set has SN solutions. The elite set is equal to the current swarm. By choosing a solution from the elite set randomly, all solutions

Table 2 Comparison results of different ρ values in KFABC

Problem	Mean				
	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.3$	$\rho = 0.5$	$\rho = 1.0$
f_1	6.28E-69	1.54E-68	1.51E-65	6.91E-62	7.21E-56
f_2	1.79E-65	2.25E-63	1.45E-62	2.31E-57	1.05E-52
f_3	2.39E-71	3.25E-69	6.02E-66	1.59E-61	2.81E-56
f_4	4.73E-97	2.98E-110	3.04E-121	5.68E-126	2.43E-133
f_5	4.01E-36	4.21E-35	6.07E-34	1.94E-32	5.62E-30
f_6	1.06E+00	1.31E+00	1.23E+00	1.52E+00	1.94E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	7.18E-66	7.18E-66	7.18E-66	7.18E-66	7.18E-66
f_9	1.34E-02	2.16E-02	2.17E-02	2.54E-02	2.28E-02
f_{10}	1.69E-01	1.39E+00	3.67E+00	1.07E+00	5.13E+00
f_{11}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{13}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{14}	-12,569.5	-12,569.5	-12,569.5	-12,569.5	-12,569.5
f_{15}	1.84E-14	2.55E-14	2.19E-14	2.47E-14	2.90E-14
f_{16}	1.57E-32	1.57E-32	1.57E-32	1.57E-32	1.57E-32
f_{17}	1.35E-32	1.35E-32	1.35E-32	1.35E-32	1.35E-32
f_{18}	1.67E-17	1.55E-16	6.29E-16	3.11E-16	3.34E-16
f_{19}	1.35E-31	1.35E-31	1.35E-31	1.35E-31	1.35E-31
f_{20}	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{21}	-7.83E+01	-7.83E+01	-7.83E+01	-7.83E+01	-7.83E+01
f_{22}	-2.96E+01	-2.95E+01	-2.96E+01	-2.96E+01	-2.95E+01

The best results are given in bold

Table 3 Mean ranking values achieved by KFABC with different ρ values

Algorithm value	Mean ranking value
KFABC ($\rho = 0.1$)	2.23
KFABC ($\rho = 0.2$)	2.89
KFABC ($\rho = 0.3$)	3.00
KFABC ($\rho = 0.5$)	3.18
KFABC ($\rho = 1.0$)	3.70

The best results are given in bold

Comparison of KFABC with other ABC variants

In this section, KFABC is compared with the original ABC and four other ABCs. The involved ABC algorithms are described as below.

- ABC [4].
- Global best guided ABC (GABC) [19].
- Modified ABC (MABC) [45].
- Multi-strategy ensemble ABC (MEABC) [27].
- Bare bones ABC (BABC) [46].
- Proposed KFABC.

In GABC, X_{best} was used to modify the search strategy and strengthen the exploitation capability [19]. In MABC, a new search strategy-based differential evolution (DE) mutation scheme (DE/best/1) was proposed [45]. Ensemble learning was embedded into MEABC and three search strategies mutually compete to generate new solutions [27]. In BABC, the onlooker bees used a new search strategy based on Gaussian distribution [46]. For the above six ABC algorithms, they are run on 22 problems with $D = 30$ and 100. The stopping condition for each algorithm is $\text{MaxFes} = 5000 \cdot D$ [20,27]. For ABC and KFABC, SN and $limit$ are set 50 and 100, respectively. In KFABC, the parameter ρ is set to 0.1. For other parameters of GABC, MABC, MEABC, and BABC, the same settings are used in terms of their corresponding literature [19,27,45,46]. For each test problem, all ABC algorithms are run 30 times.

Table 4 shows the mean best function values of KFABC and five other ABCs for $D = 30$. In the last row, “+/-/-” indicates that KFABC is better than, similar to, and worse than its competitor on the corresponding quantity of problems, respectively. From the results, KFABC is superior to ABC on 19 problems. For the rest of three problems, ABC is slightly better than KFABC on f_{10} and both of them have the same results on two problems. Similarly, KFABC is worse

Fig. 2 Convergence curves of six ABCs on six problems for $D = 30$

Table 4 Results achieved by six ABC algorithm when $D = 30$

Problem	Mean					
	ABC	GABC	MABC	MEABC	BABC	KFABC
f_1	9.49E-16	4.45E-16	3.75E-26	3.12E-40	2.48E-56	6.28E-69
f_2	3.41E-08	5.32E-16	5.61E-23	3.93E-37	1.59E-56	1.79E-65
f_3	7.22E-16	5.97E-16	3.71E-41	5.26E-41	1.88E-55	2.39E-71
f_4	3.38E-14	6.63E-17	1.52E-52	3.30E-90	5.89E-144	4.73E-97
f_5	1.58E-10	1.61E-15	4.29E-14	1.62E-21	2.71E-30	4.01E-36
f_6	3.72E+01	1.45E+01	1.07E+01	4.57E+00	1.76E+01	1.06E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	1.36E-22	9.08E-24	7.18E-66	7.18E-66	7.18E-66	7.18E-66
f_9	1.86E-01	1.03E-01	4.81E-02	3.75E-02	5.48E-02	1.34E-02
f_{10}	1.46E-01	6.84E-02	6.11E-01	3.01E-01	1.41E+00	1.69E-01
f_{11}	4.09E-14	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	9.63E-03	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{13}	7.97E-14	2.22E-16	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{14}	-12,533.8	-12,569.5	-12,569.5	-12,569.5	-12,569.5	-12,569.5
f_{15}	7.21E-05	2.32E-10	4.54E-11	3.26E-14	2.89E-14	1.84E-14
f_{16}	7.69E-16	5.53E-16	2.05E-27	1.57E-32	1.57E-32	1.57E-32
f_{17}	3.39E-15	5.47E-16	3.25E-26	1.35E-32	1.35E-32	1.35E-32
f_{18}	1.46E-06	4.55E-07	4.31E-13	1.09E-21	6.11E-16	1.67E-17
f_{19}	2.16E-13	9.69E-16	4.73E-26	1.35E-31	1.35E-31	1.35E-31
f_{20}	2.58E-01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{21}	-7.83E+01	-7.83E+01	-7.83E+01	-7.83E+01	-7.83E+01	-7.83E+01
f_{22}	-2.93E+01	-2.95E+01	-2.96E+01	-2.96E+01	-2.96E+01	-2.96E+01
+/-/-	19/2/1	15/6/1	13/9/0	9/12/1	9/12/1	-

The best results are given in bold

than GABC on f_{10} . For the rest of 21 problems, KFABC outperforms GABC. Compared with MABC, KFABC achieves better results on 13 problems, and they have similar performance on the rest of nine problems. MEABC obtains more accurate solutions than KFABC on f_{18} , while KFABC is better than MEABC on nine problems. KFABC, BABC, and MEABC gain the same results on 12 problems. BABC surpasses KFABC on f_4 , but KFABC outperforms BABC on nine problems.

Table 5 gives the mean results of KFABC and five other ABC algorithms on the test set with $D = 100$. As the dimension size D increases from 30 to 100, the overall comparison performance summarized by “+/-/-” is almost unchanged. It means KFABC is nearly not worse than ABC algorithms on all problems. There is a special case that KFABC hardly obtain reasonable solutions on f_{10} . However, GABC and ABC gain good solutions on this problem.

Figure 2 lists the convergence graphs of KFABC and other ABC algorithms on some problems. For f_1 , f_2 , f_5 , f_6 , and f_{15} , KFABC shows much faster convergence than other algorithms. For f_2 , BABC converges faster than KFABC at the early search stage. The

convergence speed of KFABC quickly surpasses BABC with increasing iterations. The convergence curves of KFABC, MEABC, and BABC approach to a straight line at the late search stage. It means that these algorithms gain similar results and cannot effectively improve the accuracy of solutions. For f_{16} , though KFABC, MEABC, and BABC achieve the same results, KFABC is much faster than MEABC and BABC. It demonstrates the proposed KFABC can strengthen the exploitation search. Therefore, both the exploitation and exploration search capabilities are effectively balanced.

Statistical results of Friedman test on all ABCs are given in Table 6. Based on Friedman test, mean ranking values of all algorithms are calculated. A smaller ranking value means a better overall optimization performance. As seen, KFABC obtains the smallest mean ranking values on $D = 30$ and 100. It demonstrates KFABC achieves the best overall optimization performance among six ABC algorithms.

Table 7 presents the statistical results of Wilcoxon test. The p values below the 0.05 significance level are shown in bold. From the results on $D = 30$, KFABC is significantly better than five other ABC algorithms. For $D = 100$, though KFABC is not significantly better than MEABC, KFABC is still significantly better than other ABCs.

Table 5 Results achieved by six ABC algorithm when $D = 100$

Problem	Mean					
	ABC	GABC	MABC	MEABC	BABC	KFABC
f_1	8.15E–15	3.24E–15	1.63E–22	2.52E–36	4.04E–50	1.98E–65
f_2	2.75E–06	3.07E–15	4.48E–19	4.48E–33	1.21E–51	1.54E–62
f_3	5.24E–15	3.23E–15	5.86E–23	7.84E–37	2.88E–50	3.36E–65
f_4	1.22E–08	4.75E–11	2.33E–52	4.03E–87	5.79E–129	1.25E–89
f_5	1.29E–09	6.82E–15	3.62E–12	2.44E–19	8.59E–28	3.67E–34
f_6	8.82E+01	8.55E+01	5.16E+01	3.98E+01	8.56E+01	9.71E+00
f_7	1.28E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	5.14E–40	6.33E–45	7.12E–218	7.12E–218	7.12E–218	7.12E–218
f_9	1.51E+00	7.37E–01	2.20E–01	1.83E–01	7.41E–01	1.39E–01
f_{10}	1.74E–01	3.18E–01	4.65E+00	1.13E+00	4.14E+01	2.73E+01
f_{11}	1.43E–11	7.11E–15	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{12}	9.75E+00	1.93E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{13}	2.48E–14	3.18E–15	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{14}	–41164.8	–41856.9	–41898.3	–41898.3	–41898.3	–41898.3
f_{15}	5.24E–05	2.75E–07	5.27E–07	3.16E–13	1.57E–13	8.81E–14
f_{16}	7.57E–15	3.23E–15	1.76E–24	4.71E–33	4.71E–33	4.71E–33
f_{17}	1.42E–13	3.31E–15	6.86E–23	1.35E–32	1.35E–32	1.35E–32
f_{18}	7.65E–04	2.96E–05	1.52E–10	2.04E–16	1.80E–15	1.48E–15
f_{19}	9.67E–12	1.22E–14	1.43E–23	1.35E–31	1.35E–31	1.35E–31
f_{20}	2.92E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{21}	–7.83E+01	–7.83E+01	–7.83E+01	–7.83E+01	–7.83E+01	–7.83E+01
f_{22}	–9.72E+01	–9.82E+01	–9.94E+01	–9.95E+01	–9.95E+01	–9.95E+01
+/=–	20/1/1	18/3/1	13/8/1	8/12/2	9/12/1	–

The best results are given in bold

Table 6 Mean ranking values achieved by KFABC and other ABC algorithms

Algorithm	Mean ranking	
	$D = 30$	$D = 100$
ABC	5.59	5.66
GABC	4.23	4.43
MABC	3.52	3.55
MEABC	2.75	2.55
BABC	2.80	2.73
KFABC	2.11	2.09

The best results are given in bold

Table 7 Results of Wilcoxon test between KFABC and five other ABC algorithms

KFABC vs.	p value	
	$D = 30$	$D = 100$
ABC	7.80E–04	7.96E–04
GABC	6.13E–03	1.69E–03
MABC	1.47E–03	1.32E–02
MEABC	2.84E–02	2.03E–01
BABC	6.91E–03	6.91E–03

The p values below the 0.05 significance level are given in bold

Comparison of KFABC with other algorithms

To further validate the performance of KFABC, we compare it with other algorithms. In this section, there are two experiments: (1) KFABC is compared with some well-known DE algorithms; and (2) KFABC is compared some recently published ABC algorithms on CEC 2013 test problems [42].

In the first experiment, KFABC is compared with jDE [47], SaDE [48], and JADE [49]. Table 8

presents the mean function values achieved by KFABC, jDE, SaDE, and JADE on several classical test problems with $D = 30$. All algorithms use the same $MaxFEs$ as the stopping condition. Results of jDE, SaDE, and JADE are taken from Table 9 in the literature [29]. From the results, KFABC outperforms jDE, SaDE, and JADE on 9 out of 10 problems. For the Schwefel 2.26 problem, all four algorithms can converge to the global minimum.

Table 8 Results achieved by KFABC and some well-known DE variants

Problem	<i>MaxFEs</i>	Mean			
		jDE	SaDE	JADE	KFABC
Sphere	1.50E+05	1.46E−28	3.28E−20	2.69E−55	6.28E−69
Schwefel 2.22	2.00E+05	9.02E−24	3.51E−25	3.18E−25	3.09E−48
Step	1.00E+04	6.31E+02	5.07E+01	5.62E+00	0.00E+00
Rastrigin	1.00E+05	3.32E−04	2.43E+00	1.33E−01	0.00E+00
Griewank	5.00E+04	7.29E−06	2.52E−09	1.57E−08	1.76E−13
Schwefel 2.26	1.00E+05	−12,569.5	−12,569.5	−12,569.5	−12,569.5
Ackley	5.00E+04	2.37E−04	3.81E−06	3.35E−09	1.21E−09
Penalized 1	5.00E+04	1.80E−05	1.93E−09	1.87E−10	2.59E−22
Penalized 2	5.00E+04	7.03E−08	8.25E−12	1.67E−15	1.17E−20
Alpine	3.00E+05	6.08E−10	2.94E−06	2.78E−05	3.31E−41
+/=/−		9/1/0	9/1/0	9/1/0	−

The best results are given in bold

Table 9 Results on CEC 2013 benchmark problems

Problem	Mean				
	dABC	IABC	OABC	DRABC	KFABC
Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rotated high conditioned elliptic	1.81E+06	1.51E+06	5.05E+06	1.00E+06	1.02E+06
Rotated bent cigar	5.62E+06	5.54E+07	4.36E+07	9.68E+05	2.90E+05
Rotated discus	1.08E+04	5.63E+03	1.03E+04	8.91E+03	6.73E+03
Different Powers	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
Rotated Rosenbrock	5.17E−01	3.22E+00	1.08E+00	3.68E−01	1.86E−01
Rotated Schaffers F7	3.28E+01	1.89E+01	5.05E+01	1.68E+01	6.53E+00
Rotated Ackley	2.04E+01	2.04E+01	2.04E+01	2.04E+01	2.04E+01
Rotated Weierstrass	5.06E+00	4.15E+00	6.11E+00	3.27E+00	6.08E+00
Rotated Griewank	1.09E+00	1.03E+00	2.59E+00	2.95E−01	4.15E−01
+/=/−	6/3/1	5/3/2	7/3/0	4/3/3	−

The best results are given in bold

In the second experiment, KFABC is compared with dABC [50], IABC [51], OABC [52], and DRABC [34]. Table 9 gives the mean error values obtained by KFABC, dABC, IABC, OABC, and DRABC on ten CEC 2013 benchmark problems with $D = 10$. All algorithms use the same *MaxFEs* as the stopping condition and it is set to $1.0E+05$ [42]. The problems names are listed in the first column of Table 9, and the specific mathematical definitions of these problems can be found in [42]. Results of dABC, IABC, OABC, and DRABC are taken from Table 13 in the literature [34]. From the results, all five ABC algorithms obtain the same solutions on three problems. dABC performs better than KFABC on only one problem, while KFABC is better on six problems. IABC outperforms KFABC on two problems. For the rest of five problems, KFABC achieves better results. In contrast with OABC, KFABC obtain more accurate solutions on seven problems. DRABC surpasses KFABC on three problem, but KFABC is better than DRABC on four problems.

Effects of different knowledge

The proposed KFABC approach is a hybrid version of ABC, and it incorporates ABC with three types of knowledge: multiple search strategies (K1), elite solutions (K2), and abandoned solutions (K3). To investigate the effects of different knowledge, ABC with each kind of knowledge is tested. The involved algorithms are described as follows.

- ABC: ABC without any knowledge.
- ABC + K1: ABC with the first kind of knowledge (K1).
- ABC + K2: ABC with the second kind of knowledge (K2).
- ABC + K3: ABC with the third kind of knowledge (K1).
- Proposed KFABC (ABC + K1 + K2 + K3): ABC with three kinds of knowledge (K1 + K2 + K3).

Table 10 presents the mean function values obtained by ABC with different kinds of knowledge on ten test problems

Table 10 Results on ABC with different kinds of knowledge

Problem	Mean				
	ABC	ABC+K1	ABC+K2	ABC+K3	KFABC (ABC+K1+K2+K3)
f_1	9.49E–16	6.21E–23	1.84E–49	3.01E–17	6.28E–69
f_2	3.41E–08	6.36E–13	7.16E–45	3.32E–09	1.79E–65
f_3	7.22E–16	3.09E–23	1.57E–49	1.38E–19	2.39E–71
f_4	3.38E–14	5.45E–18	1.96E–35	2.29E–24	4.73E–97
f_5	1.58E–10	1.76E–23	3.13E–25	1.16E–10	4.01E–36
f_6	3.72E+01	8.07E+00	2.92E+01	1.02E+01	1.06E+00
f_7	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_8	1.36E–22	7.18E–66	7.18E–66	7.18E–66	7.18E–66
f_9	1.86E–01	3.59E–02	1.21E–01	1.32E–01	1.34E–02
f_{10}	1.46E–01	4.95E–01	5.49E–02	7.76E–02	1.69E–01

The best results are given in bold

($D = 30$). The parameter settings are the same with Sect. 4.3. From the results, ABC + K1, ABC + K2, ABC + K3, and KFABC are better than ABC on all test problems except for f_7 and f_{10} . All kinds of knowledge can help ABC obtain better solutions. Especially for the second kind of knowledge, it significantly improves the quality of solutions. For the problem f_{10} , ABC + K1 and KFABC are worse than ABC, while ABC + K2 and ABC + K3 outperform ABC. It means that the first kind of knowledge is not suitable for this problem. By combining ABC and three kinds of knowledge, KFABC obtains much better solutions than ABC with a single knowledge.

Conclusions

To solve complex problems, various improved strategies have been used to enhance the search capabilities of intelligent optimization algorithms. In this work, a novel ABC algorithm (namely KFABC) is presented based on the view of knowledge fusion. To construct KFABC, three issues are addressed: (1) knowledge representation; (2) knowledge utilization; and (3) learning mechanism. In our approach, three kinds of knowledge is selected. For each kind of knowledge, the corresponding utilization method is designed. By sensing the search status, a learning mechanism is proposed to adaptively choose appropriate knowledge. To validate the performance of KFABC, 32 benchmark problems are tested. Performance of KFABC is compared with nine ABC and three DE algorithms.

For the second kind of knowledge, a parameter ρ is introduced to control the quantity of elite solutions. Different ρ values are tested to investigate the effects of ρ on the performance of KFABC. Results show that the parameter ρ does not seriously affect the performance of KFABC. Five different ρ values can help KFABC obtain good performance. Based on Friedman test, $\rho = 0.1$ is a relatively good choice.

From the comparisons among KFABC, BABC, MEABC, MABC, GABC, and ABC, KFABC is superior to other ABC algorithms on $D = 30$ and 100. The convergence curves show that the proposed method can strengthen the exploitation search. Both the exploitation and exploration search capabilities are effectively balanced. Compared with some famous DE algorithms, KFABC also achieves better results. For some complex CEC 2013 benchmark functions, KFABC is better than some recently published ABC algorithms. Another experiment demonstrates that each kind of knowledge can help ABC obtain better solutions. ABC with three kinds of knowledge is better than that with a single knowledge.

This paper presents a preliminary study on incorporating knowledge in ABC. The idea can be similarly applied to other intelligent optimization algorithms. How to represent and use knowledge still needs further research. There exist several explorative and exploitative search strategies and only two simple search strategies are used in the approach. Other search strategies may be more effective. In addition, the IRAFV may not work at the last search stage. More experiments will be studied in the future work.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Nos. 61663028 and 61966019) and the Innovation Research Team Program of Nanchang Institute of Technology.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Asghari S, Navimipour NJ (2019) Cloud service composition using an inverted ant colony optimisation algorithm. *Int J Bio Inspir Comput* 13(4):257–268
- Wang F, Zhang H, Li KS, Li ZY, Yang J, Shen XL (2018) A hybrid particle swarm optimization algorithm using adaptive learning strategy. *Inf Sci* 436(437):162–177
- Wang GG, Tan Y (2019) Improving metaheuristic algorithms with information feedback models. *IEEE Trans Cybern* 49(2):542–555
- Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer engineering Department
- Wang H, Wang WJ, Xiao SY, Cui ZH, Xu MY, Zhou XY (2020) Improving artificial Bee colony algorithm using a new neighborhood. *Inf Sci* 527:227–240
- Wang H, Wang WJ, Cui ZH, Zhou XY, Zhao J, Li Y (2018) A new dynamic firefly algorithm for demand estimation of water resources. *Inf Sci* 438:95–106
- Wang H, Wang WJ, Cui LZ, Sun H, Zhao J, Wang Y, Xue Y (2018) A hybrid multi-objective firefly algorithm for big data optimization. *Appl Soft Comput* 69:806–815
- Hu P, Pan JS, Chu SC (2020) Improved binary grey wolf optimizer and its application for feature selection. *Knowl Based Syst*. <https://doi.org/10.1016/j.knsys.2020.105746>
- Wang F, Li YX, Zhou AM, Tang K (2019) An estimation of distribution algorithm for mixed-variable News vendor problems. *IEEE Trans Evol Comput*. <https://doi.org/10.1109/TEVC.2019.2932624>
- Cui ZH, Cao Y, Cai XJ, Cai JH, Chen JJ (2019) Optimal LEACH protocol with modified bat algorithm for big data sensing systems in Internet of Things. *J Parallel Distrib Comput* 132:217–229
- Wang GG, Guo L, Gandomi AH, Hao GS, Wang H (2014) Chaotic krill herd algorithm. *Inf Sci* 274:17–34
- Wang GG, Deb S, Cui Z (2019) Monarch butterfly optimization. *Neural Comput Appl* 31(7):1995–2014
- Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. *Appl Math Comput* 214:108–132
- Amiri E, Dehkordi MN (2018) Dynamic data clustering by combining improved discrete artificial bee colony algorithm with fuzzy logic. *Int J Bio Inspir Comput* 12(3):164–172
- Ma LB, Wang XW, Shen H, Huang M (2019) A novel artificial bee colony optimiser with dynamic population size for multi-level threshold image segmentation. *Int J Bio Inspir Comput* 13(1):32–44
- Ma LB, Wang XW, Huang M, Lin ZW, Tian LW, Chen HN (2019) Two-level master-slave RFID networks planning via hybrid multiobjective artificial bee colony optimizer. *IEEE Trans Syst Man Cybern Syst* 49(5):861–880
- Cui LZ, Li GH, Wang XZ, Lin QZ, Chen JY, Lu N, Lu J (2017) A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. *Inf Sci* 417:169–185
- Agarwal P, Mehta S (2019) ABC_DE_FP: a novel hybrid algorithm for complex continuous optimisation problems. *Int J Bio Inspir Comput* 14(1):46–61
- Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. *Appl Math Comput* 217:3166–3173
- Cui LZ, Li GH, Li QZ, Du ZH, Gao WF, Chen JY, Lu N (2016) A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. *Inf Sci* 367:1012–1044
- Zhou XY, Wu ZJ, Wang H, Rahnamayan S (2016) Gaussian bare-bones artificial bee colony algorithm. *Soft Comput* 20(3):907–924
- Hussain A, Muhammad YS (2019) Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator. *Complex Intell Syst*. <https://doi.org/10.1007/s40747-019-0102-7> (to be published)
- Karaboga D, Akay B (2019) A survey: algorithms simulating bee swarm intelligence. *Artif Intell Rev* 31(1):68–85
- Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. *Appl Soft Comput* 11(2):2888–2901
- Xue Y, Jiang J, Zhao B, Ma T (2018) A self-adaptive artificial bee colony algorithm based on global best for global optimization. *Soft Comput* 22(9):2935–2952
- Peng H, Deng C, Wu Z (2019) Best neighbor guided artificial bee colony algorithm for continuous optimization problems. *Soft Comput* 23(18):8723–8740
- Wang H, Wu Z, Rahnamayan S, Sun H, Liu Y, Pan JS (2014) Multi-strategy ensemble artificial bee colony algorithm. *Inf Sci* 279:587–603
- Kiran MS, Hakli H, Gunduz M, Uguz H (2015) Artificial bee colony algorithm with variable search strategy for continuous optimization. *Inf Sci* 300:140–157
- Gao WF, Wei Z, Luo Y, Cai J (2019) Artificial bee colony algorithm based on Parzen window method. *Appl Soft Comput* 74:679–692
- Gao WF, Huang LL, Liu SY, Chan FTS, Dai C, Shan X (2015) Artificial bee colony algorithm with multiple search strategies. *Appl Math Comput* 271:269–287
- Xiao SY, Wang WJ, Wang H, Zhou XY (2019) A new artificial bee colony based on multiple search strategies and dimension selection. *IEEE Access* 7:133982–133995
- Saad A, Khan SA, Mahmood A (2018) A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design. *Swarm Evol Comput* 38:187–201
- Ozturk C, Hancer E, Karaboga D (2015) A novel binary artificial bee colony algorithm based on genetic operators. *Inf Sci* 297:154–170
- Bajer D, Zorić B (2019) An effective refined artificial bee colony algorithm for numerical optimisation. *Inf Sci* 504:221–275
- Gao WF, Liu SY, Huang LL (2013) A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. *IEEE Trans Cybern* 43(3):1011–1024
- Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: *Proceedings of International Conference on Computational Intelligence for Modeling Control and Automation*, Vienna, pp 695–701
- Wang H, Wu ZJ, Rahnamayan S, Liu Y, Ventresca M (2011) Enhancing particle swarm optimization using generalized opposition-based learning. *Inf Sci* 181:4699–4714
- Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. *IEEE Trans Evol Comput* 12:4–79
- Wang H, Liu Y, Zeng SY, Li H, Li CH (2007) Opposition-based particle swarm algorithm with Cauchy mutation. In: *Proceedings of IEEE Congress on Evolutionary Computation*, Singapore, pp 4750–4756
- Sapre S, Mini S (2019) Opposition-based moth flame optimization with Cauchy mutation and evolutionary boundary constraint handling for global optimization. *Soft Comput* 23:6023–6041
- Ali M, Pant M (2011) Improving the performance of differential evolution algorithm using Cauchy mutation. *Soft Comput* 15:991–1007
- Liang JJ, Qu BY, Suganthan PN, Hernández-Díaz AG (2013) Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Technical Report 201212. Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Computational Intelligence Laboratory
- Wang H, Rahnamayan S, Sun H, Omran MGH (2013) Gaussian bare-bones differential evolution. *IEEE Trans Cybern* 43(2):634–647

44. Wang H, Sun H, Li CH, Rahnamayan S, Pan JS (2013) Diversity enhanced particle swarm optimization with neighborhood search. *Inf Sci* 223:119–135
45. Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. *Comput Oper Res* 39:687–697
46. Gao WF, Chan FTS, Huang LL, Liu SY (2015) Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. *Inf Sci* 316:180–200
47. Brest J, Greiner S, Bošković B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans Evol Comput* 10(6):646–657
48. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaption for global numerical optimization. *IEEE Trans Evol Comput* 13(2):398–417
49. Zhang J, Sanderson AC (2009) JADE: Adaptive differential evolution with optional external archive. *IEEE Trans Evol Comput* 13(5):945–958
50. Kiran MS, Findik O (2015) A directed artificial bee colony algorithm. *Appl Soft Comput* 26:454–462
51. Cao Y, Lu Y, Pan X, Sun N (2019) An improved global best guided artificial bee colony algorithm for continuous optimization problems. *Cluster Comput* 22:3011–3019
52. Sharma TK, Gupta P (2018) Opposition learning based phases in artificial bee colony. *Int J Syst Assur Eng Manag* 9(1):262–273

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.