The Journal of Supercomputing (2019) 75:5189–5226
https://doi.org/10.1007/s11227-019-02786-w

Improving the performance of feature selection and data clustering with novel global search and elite-guided artificial bee colony algorithm

Zhenxin Du[1,2] · Dezhi Han[2] · Kuan-Ching Li[3]

Published online: 27 February 2019

Abstract

As known, the artificial bee colony (ABC) algorithm is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm that has been proven its efficacy and successfully applied to a large number of practical problems. Aiming at the trade-off between convergence speed and precocity of ABC algorithm with elite-guided search equations (ABC_elite), an enhanced version, namely EABC_elite, is proposed in this paper, and the improvements are twofold. As the global best (gbest) solution is introduced to the search equation and acceleration of the convergence in the bee phase of EABC_elite, the former in the ordinary solution is embodied to the search equation yet balance the gbest's ability. The enhancement to the global search by making the information of gbest and ordinary solutions be adequately used while keeping the exploration–exploitation balance well maintained, the usual solution is introduced to the search equation to avoid the precocity problem in the onlooker bee phase of EABC_elite as the latter one. Experimental analysis and evaluations of EABC_elite against several state-of-the-art variants of the ABC algorithm demonstrate that the EABC_elite is significantly better than the compared algorithms in the feature selection problem. Also, the proposed EABC_elite algorithm is modified to combine the *K*-means initialization strategy and chaotic parameters strategy to further enhance the global search of EABC_elite for data clustering. Experimental results show that the derived EABC_elite clustering algorithm "Two-step EABC_elite," TEABC_elite for short, delivered better and promising results than previous works for data clustering.

Keywords Artificial bee colony · Data clustering · Feature selection · Global search

✉ Kuan-Ching Li
kuancli@pu.edu.tw

Extended author information available on the last page of the article

1 Introduction

With the advances in science and engineering moving at a faster pace than ever, optimization techniques play an essential role. During the recent past, evolutionary algorithms (EAs) have achieved with success yet solving in effectively way optimization problems characterized by non-convex, discontinuous, and non-differentiable. Some famous EAs have been proposed, such as genetic algorithm (GA) [1], differential evolution (DE) [2], particle swarm optimization (PSO) [3], and ant colony optimization (ACO) [4]. Artificial bee colony algorithm (ABC) [5–8] is a most recently proposed EA that belongs to the group of swarm intelligence algorithms that mimics the intelligent foraging behavior of honey bees. When compared to selected state-of-the-art EAs, such as GA, DE, and PSO [5–7], comparison results indicate its efficacy and also competitive performance. It has been proven to show superior performance when dealing with optimization problems, owing to its simple structure and excellent performance [8], such as flowshop scheduling problem [9], filter design problem [10], and vehicle routing problem [11].

Despite ABC's excellent performance, it suffers from slow convergence speed yet easily being trapped by local optimum, which is mainly due to its solution search equation, that is very good in exploration though poor in exploitation, unfortunately. For the sake of the excellent performance on optimization problems, the primary challenge is how to maintain the exploration–exploitation balance during the search process [12].

A large number of ABC variants have been proposed to improve the overall performance. Firstly, Zhu and Kwong [13] introduced the global best (gbest) solution into the search equation of ABC to enhance the exploitation ability of ABC, though some follow-up researches indicate that the use of gbest easily outcome in the precocity problem since all individuals learn from the gbest solution. To settle this problem, Gao and Liu [14] proposed a novel crossover operator-based ABC (CABC), which has no bias in any search direction. Cui et al. pointed out that, despite CABC can avert precocity effectively, the useful information of the population has not been utilized effectively, especially the information of gbest [15]. After that, they proposed a novel elite-guided ABC, ABC_elite for short, which can keep a better balance between accelerating convergence and averting precocity problem. Experiments show that the ABC_elite is significantly better than several state-of-the-art ABC variants, such as BABC [12], CABC [14], ABCVSS [16], best-so-far ABC [17], MABC [18], qABC [19], EABC [20], and several PSO and DE variants on most of the test functions in terms of solution quality, robustness, and convergence speed. Thus, it is noted that the novel revised search equations are the main factors for the success of ABC_elite. Nevertheless, all candidates are generated around elite solutions in the search equations of ABC_elite, where the information of ordinary solutions has not been utilized effectively, so the search area is relatively small and the global search ability should be improved. Meanwhile, the information of gbest is only utilized by the elite individuals, added to the hard exploitation ability of ABC_elite in the onlooker bee phase, which quickly falls in precocity problem.

To solve the abovementioned items, this paper proposes an enhanced version of ABC_elite, namely EABC_elite. The contributions of this paper are as follows:

- A novel enhanced version of ABC_elite is proposed using two novel search equations. In EABC_elite, all individuals are guided by the global best solution that can accelerate the convergence process. Ordinary individuals are also embedded in the search equation to balance exploration and exploitation that effectively enhance the global search ability of ABC_elite. Through the experimental data, EABC_elite has not only faster convergence speed but also good global search ability, maintaining the simplicity of ABC_elite, and bringing the computation complexity of EABC_elite and ABC_elite approximately the same. Experimental results show that EABC_elite performs well on unimodal, multimodal, shifted, and rotated functions when compared with recently ABC and non-ABC variants. Additional experiments on UCI machine learning datasets show that EABC_elite is a compelling feature selection tool.
- By combining the *K*-means initialization strategy and chaotic parameters strategy with EABC_elite, a novel data clustering method named TEABC_elite is proposed. Experimental results on UCI machine learning datasets show its effectiveness as clustering tool, owing to its excellent global search ability.

The remaining of this paper is organized as follows. Related work on ABC is presented in Sect. 2, and a novel elite-guided ABC with global search equations is proposed in Sect. 3, EABC_elite for short. Section 4 presents the comparison experiments with other ABC variants and deriving a variant of the EABC_elite named two-step EABC_elite (TEABC_elite for short) by combining the *K*-means initialization strategy and chaotic parameter strategy to further enhance the global search ability aiming at solving the clustering problem in Sect. 5. Finally, concluding remarks are given in Sect. 6.

2 Related work

2.1 Original ABC

As known, the ABC algorithm has been developed to mimic the foraging behaviors of honey bee colonies, where the location of the food source represents the potentially best solution to a problem, and the amount of nectar per food source represents the quality of the solution. It consists of four sequentially realized phases, namely initialization, employed bee, onlooker bee, and scout bee phases. After initialization, it turns to be a cycle that uses the employed bee phase, onlooker bee phase, and scout bee phase. The complete execution for each phase is depicted as follows:

- Initialization phase: At the beginning of ABC, each food source is randomly generated, following

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

where $i=1,\ldots SN$, $j=1,\ldots D$, SN denoting the number of food sources ($SN=$the number of employed bees$=$the number of onlooker bees) and D the dimensionality of the optimization problem. The x_j^L and x_j^U are the lower and upper bounds of the j-th dimension, respectively, and $rand_j$ is a randomly generated number in the range [0, 1]. Next, the fitness value of each food source is obtained as:

$$fit_i=\begin{cases} \frac{1}{1+f(x_i)} & if\ (f(x_i) \geq 0) \\ 1+\left|f(x_i)\right|, & otherwise \end{cases} \tag{2}$$

where fit_i denotes the fitness of the i-th food source x_i, and $f(x_i)$ the objective function value of the food source x_i. In the initialization phase, the parameter *limit* should be predetermined, whereas the parameter *counter* is used to record the number of unsuccessful updates and set to zero for all food sources.

- Employed bee phase: Each employed bee searches for a food source and tries to locate a candidate food source near the corresponding parent food source according to

$$v_{i,j} = x_{i,j} + \phi_{i,j} * (x_{i,j} - x_{k,j}) \tag{3}$$

where i, $k \in \{1, 2,\ldots SN\}$, $j \in \{1, 2\ldots D\}$,$v_{i,j}$ is the j-th dimension of the i-th candidate food source (new solution); $x_{i,j}$ is the j-th dimension of the i-th food source; $x_{k,j}$ is the j-th dimension of the k-th food source; k is picked up from $\{1, 2,\ldots, SN\}$ randomly and $k \neq i$; j is randomly selected from $\{1, 2,\ldots, D\}$; $\phi_{i,j}$ is a randomly generated number in the range of $[-1, 1]$. After establishing a new food source, the fitness of the candidate food sources is calculated by Eq. (2). If the candidate food source is superior to the old food source, the candidate food source will replace the old food source and the *counter* value of the food source will be reset to zero. Otherwise, the *counter* value is incremented by 1.

- Onlooker bee phase: According to the quality information of the food sources provided by the employed bees, each of the onlooker bees will fly to the food source x_s, as chosen by the roulette wheel to generate a candidate food source using Eq. (3). Besides, the selection probability of the i-th food source is calculated as Eq. (4). Note that, the higher the fitness value is, the higher the selection probability is. If a candidate food source v_s generated by the onlooker bee is better than the food source x_s, x_s will be replaced by the new one, and its *counter* value is reset to zero. Otherwise, its *counter* value is increased by 1.

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i} \tag{4}$$

- Scout bee phase: The food source with the highest *counter* value is selected. If the *counter* value is larger than the *limit* value, the food source is reinitialized according to Eq. (1). After the new food source is generated, the corresponding *counter* value is reset to zero. Note that, if $v_{i,j}$ violates the boundary constraints in the employed bee phase and onlooker bee phase, the reset is required, according to Eq. (1).

2.2 Improved ABCs

The balance between exploration and exploitation abilities plays an essential role in EAs. The exploration ability denotes the ability of an EA to search unknown area, whereas the exploitation ability denotes the ability of an EA to search around the already found area elaborately. An EA with strong exploration ability can easily escape the local optima, though the EA will evolve slowly. Nevertheless, if an EA has strong exploitation ability, the EA will evolve fast and quickly get trapped into the local minima. Whether an EA can balance the two contradictory aspects is the key to obtain a relatively high performance yet efficiency.

2.2.1 GABC algorithm

Inspired by PSO in 2010, Zhu and Kwong [13] proposed an improved version of ABC algorithm called GABC, which incorporates the information of the global best (gbest) solution into their solution search equation, enhancing the exploitation ability of ABC.

$$x_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j}) + \psi_{i,j} \cdot (x_{best,j} - x_{i,j}) \tag{5}$$

where $\psi_{i,j}$ is a randomly generated number in the range of [0, 1.5]. The term $x_{best,j}$ denotes the j-th element of the gbest, a newly proposed term. Experimental results demonstrate that GABC is better than the original ABC on most of the cases. Based on GABC, many improved versions have been proposed consecutively.

2.2.2 IABC algorithm

As claimed in [14], Eq. (5) may cause oscillations, so the convergence may be deteriorated, since the guidance of the last two terms may be in opposite directions. To solve this problem, Gao and Liu [21] proposed IABC, an improved search equation given by:

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

where $r1$ is randomly picked up from $\{1, 2, \ldots, SN\}$, $r1 \neq i$.

2.2.3 CABC algorithm

Gao et al. [14] identified that all candidates are generated around gbest according to Eq. (6), so that the exploitation ability of IABC is too strong and easy to result in precocity problem. Therefore, to address the above issues in (5) and (6), they proposed CABC, an enhanced search equation inspired by the crossover operator of GA, as shown in (7):

$$v_{i,j} = x_{r1,j} + \phi_{i,j} \cdot (x_{r1,j} - x_{r2,j}) \tag{7}$$

where $r1$ and $r2$ are two distinct integers randomly picked up from $\{1, 2, \ldots, SN\}$, and both are different from the base index i. Equation (7) has no bias to any search direction, and there is only one guidance $\phi_{i,j}(x_{r1,j} - x_{r2,j})$ in (7) that can effectively avoid oscillation phenomenon. After that, the search capability of ABC is significantly improved by (7).

2.2.4 MGABC algorithm

To avoid the oscillation phenomenon in GABC, Cui et al. [22] proposed an improved version of GABC, namely MGABC, shown as

$$v_{i,j} = \begin{cases} x_{i,j} + \phi_{i,j}.(x_{i,j} - x_{k,j}), & \textit{if rand} < P \\ x_{i,j} + \psi_{i,j}.(x_{best,j} - x_{i,j}), & \textit{otherwise}. \end{cases} \tag{8}$$

where P is a newly introduced parameter, $0<P<1$, and other symbols have the same meaning as (5).

2.2.5 ABC_elite and DFSABC_elite algorithms

Cui et al. [15] have pointed out that, despite CABC has strong global search ability, the success information of the population is not utilized, either not utilized the valuable information of gbest. To best utilize the useful information and maintain the balance between exploration and exploitation, they proposed ABC_elite, a novel version of elite-guided ABC using two novel search equations as shown in (9) and (10):

$$x_{i,j} = x_{e,j} + \phi_{i,j} \cdot (x_{e,j} - x_{k,j}) \tag{9}$$

$$x_{e,j} = \frac{1}{2}(x_{e,j} + x_{best,j}) + \phi_{e,j} \cdot (x_{best,j} - x_{k,j}) \tag{10}$$

where x_e is a randomly selected elite solution from the top $p.SN$ solutions, $p \in (0, 1)$, x_k is randomly chosen from the current population; $e \neq k \neq i$, x_{best} is the global best solution; $\phi_{i,j}$ and $\phi_{e,j}$ are two randomly generated numbers in $[-1, 1]$.

Equation (9) is used in the employed bee phase that exploits the beneficial information from the elite solutions, while Eq. (10) is used in the onlooker bee phase to simultaneously exploit the valuable information among current best solution and other elite solutions. Meanwhile, Cui et al. [15] proposed a novel depth-first strategy (DFS) to accelerate the convergence process. In DFS, a food source will search its vicinity continuously until a failed search is finished. By combining ABC_elite with DFS, the DFSABC_elite algorithm is proposed in [15].

Under the guidance from only one term, Eqs. (9) and (10) can also easily avoid the oscillation problem. In this way, the ABC_elite and DFSABC_elite algorithms can better balance the exploration and exploitation and have shown better performance when compared with other state-of-the-art EA variants, such as the GABC [13], CABC [14], best-so-far ABC [17], MABC [18], qABC [19], EABC [20], ABCVSS [16], BABC [12], and several PSO and DE variants.

2.2.6 IABC_elite algorithm

The high performance of ABC_elite and DFSABC_elite has attracted some follow-up researches. Inspired by the theory of labor division of honey bees, Du et al. [7]

proposed IABC_elite, an improved version of the ABC_elite algorithm to enhance the exploitation ability of ABC_elite by using two new search equations in the employed bee phase and onlooker bee phase of ABC_elite, respectively.

$$v_{i,j} = N\left(\frac{x_{best,j} + x_{i,j}}{2}, |x_{best,j} - x_{i,j}|\right) \tag{11}$$

$$v_{e,j} = \frac{1}{2}(x_{e,j} + x_{best,j}) + \phi_{e,j}(x_{best,j} - x_{e',j}) \tag{12}$$

where $x_{i,j}$ is the j-th element of elite solution x_i; $x_{best,j}$ is the j-th element of the global best solution found so far; j is randomly selected from $\{1, 2,\ldots, D\}$; e' is the number of a randomly selected elite solution; and Eq. (11) is used in the employed bee phase only to refine the elite individuals and enhance the exploitation ability. For the sake of the exploration–exploitation balance, ordinary individuals still use Eq. (9). In the onlooker bee phase of IABC_elite, the elite individuals alternatively use Eqs. (10) and (12) at probability P_o and $1-P_o$, respectively, and P_o decreases as the iteration number increases to enhance the exploitation ability gradually. Given that Eqs. (11) and (12) have strong exploitation ability, the DFS strategy of DFSABC_elite is discarded in IABC_elite to maintain a better balance of exploration–exploitation.

2.2.7 ECABC algorithm

To further enhance the exploitation ability of DFSABC_elite and inspired by the natural phenomenon that honey bees follow the elite group in the foraging process, Kong et al. [23] proposed ECABC, a novel elite group center-based artificial bee colony algorithm. In ECABC, Eqs. (9) and (10) are all replaced by equation

$$v_{i,j} = XEC_j + \phi_{i,j}(x_{best,j} - x_{k,j}) \tag{13}$$

where XEC is the center of the elite group. By comparing Eq. (13) to Eqs. (9) and (10), we can identify that Eq. (13) has strong exploitation ability, since the base vector XEC_j of Eq. (13) is only composed of elite individuals and the disturbation part $\phi_{i,j}(x_{best,j} - x_{k,j})$ always include the gbest term x_{best} both in the employed bee phase and in the onlooker bee phase. That is, ECABC only searches around elite individuals. To better maintain the balance of exploration–exploitation, ECABC abandoned the DFS strategy of DFSABC_elite in the employed bee phase and still use the DFS strategy in the onlooker bee phase.

2.2.8 ABCLGII algorithm

With the introduction of communication mechanisms into ABC, Lin et al. [24] proposed ABCLGII, a novel ABC algorithm with local and global information interaction. They use Eq. (14) in the employed bee phase to mimic the local interaction of honey bees.

$$v_{i,j} = x_{i,j} + rand(0, 1) \cdot (x_{nbest,j} - x_{i,j}) \tag{14}$$

where x_i is a randomly selected ordinary individual; $j=1,\ldots D$; x_{nbest} is the best food source with the smallest objective function within the distance md from x_i. In the onlooker bee phase, ABCLGII alternatively uses two new search Eqs. (15) and (16) at probability P_{str} and $1-P_{str}$, respectively. At the initial stage, P_{str} is initialized to 0.5, and after all high-quality food source positions are searched by the onlooker bees (i.e., elite individuals), P_{str} will be updated.

$$v_{i,j} = x_{pbest,j} + \phi \cdot (x_{i,j} - x_{k,j}) \tag{15}$$

$$v_{i,j} = x_{best,j} + \phi \cdot (x_{best,j} - x_{i,j}) \tag{16}$$

where x_i and x_{pbest} are all randomly selected elite individuals, $i \neq pbest$. That is, only elite individuals (high-quality food sources) have a chance to attract onlooker bees to exploit within their vicinity, which is the same as DFSABC_elite.

3 Proposed approach

In this section, we will first analyze the merits and demerits of ABC_elite and then propose an enhanced global search ABC_elite, EABC_elite for short.

3.1 Evaluations of ABC_elite

In contrast to GABC, CABC, and IABC, the main advantage of ABC_elite is that it can better balance the exploration and exploitation ability by using elite-guided search equations. GABC and IABC are guided by gbest, yet easy to result in precocity problem. Although CABC can solve precocity problem effectively by removing gbest from its search equation and maintain higher global search ability, CABC can also suffer from a slow convergence speed due to the lack of the previous success information of the population.

Although ABC_elite has shown to be competitive to other EAs, there are still drawbacks in its solution search equations. In such equations, a candidate solution is produced by adding a disturbation vector to a base vector. To be specific, in Eq. (9), the base vector is x_e and the disturbation vector is x_e-x_k. In Eq. (10), the base vector is $(x_{best}+x_e)/2$, and the disturbation vector is $x_{best}-x_k$.

For simplicity, the coefficient ϕ is not considered since it is the same in all ABCs. As noted, the base vectors of these equations are elite solutions, and all candidates are generated around elite solutions, so the search area of ABC_elite is relatively small since elite solutions only account for a small proportion p ($p=0.1$ in [15]).

In the search equation of ABC, GABC, and CABC (respectively, Eqs. (3), (5) and (7)), the base vectors are all ordinary solutions, providing sufficient opportunity for ordinary solutions to take part in the evolution process. Therefore, the algorithms have a high global search ability. However, in the search Eqs. (9) and (10) of ABC_elite, the base vectors are all elite solutions. Thus, the ordinary solutions have no sufficient opportunities to be exploited, as they take only part in the evolution process as a disturbation vector but not a base vector. Besides, the disturbation amplitude in (9) is relatively significant, since the x_{best} is the current best solution in the population, and x_k is an ordinary solution. Generally speaking, the fitness of x_{best} is far better than x_k, thus $|x_{best,j} - x_{k,j}|$ is a relatively big disturbation with high probability, which will result in the candidate generated by (10) away from elite solutions and x_{best}.

3.2 Motivation

In the literature, the high performance of ABC_elite and DFSABC_elite has attracted much attention. Although recently developed ABC_elite variants have improved the performance of ABC_elite, they have their shortcomings. IABC_elite is the first improved ABC_elite variant, but in IABC_elite only elite individuals have a chance to be guided by the gbest solution since Eq. (11) is used only by elite individuals to maintain exploration–exploitation balance and Eqs. (10) and (12) in IABC_elite are all used for elite individuals. That is, the search of the ordinary individuals is almost blind, which make up most of the population. Therefore, the proposed Eqs. (11) and (12) are mainly used to refine elite solutions.

ECABC is the latest proposed ABC_elite variant and has shown excellent performance when compared to several state-of-the-art ABC variants, though we have not seen its comparisons with non-ABCs especially on shifted and rotated functions or real-world problems. The most significant shortcoming of ECABC is its excessive exploitation ability since, as mentioned above, the basic vector of the right hand of Eq. (13) is only composed of elite individuals (including gbest) and the disturbation part in the right hand of Eq. (13) includes the gbest term. Results show that ECABC beats DFSABC_elite when $D = 30$ by a large score 3:9, but when the dimension D increases to 50 and 100, the scores are only 5:6 and 5:7, respectively [23]. That is, ECABC only beats DFSABC_elite in low-dimensional functions due to the excessive exploitation ability which is beneficial for solving simple functions (i.e., unimodal functions and low-dimensional functions). ABCLGII faces the same problem as ABC_elite and IABC_elite, i.e., ordinary individuals are not influenced by gbest.

In this paper, a novel enhanced ABC_elite (EABC_elite) is proposed, where all the ordinary individuals are guided by gbest while the balance of exploration–exploitation can still be well maintained. Experimental results show that EABC_elite has significant advantages over DFSABC_elite on 22 basic functions and CEC 2015 [25] shifted and rotated functions. By contrast, several recent proposed ABC variants have similar performance with DFSABC_elite. For example, the newly proposed grey ABC beats DFSABC_elite only at a score of 15:14 on CEC functions [26]; DFSABC_elite beats the newly proposed ABCG on CEC functions [8].

3.3 Proposed algorithm

Li and Zhan [27] summarized the developing rules of several EAs and gave a conclusion that “the more information is efficiently utilized to guide the flying, the better performance the algorithm will have.” In the original PSO, all particles learn from gbest, which often result in the precocity problem. To settle this problem, a series of improved PSO is proposed consecutively, such as the competitive and cooperative

PSO [27], social learning PSO [28], and self-learning PSO [29]. Although the theory of PSO variants is different, they all use more population information to escape the local minima differently. The development of ABC has gone through a similar process. In contrast with GABC, CABC, and IABC, ABC_elite uses more information to help the algorithm to escape the local minima, so ABC_elite results the best performance.

In EAs, a prevalent theory is that if an EA employs stronger heuristic information to guide the evolution, a better balance strategy between exploration and exploitation should be employed simultaneously, or the EA will trap in local minima fastly under the guidance of firm heuristic information. ABC_elite uses the gbest to guide the evolution and uses the elite solutions to weaken the excellent guidance of gbest, so the balance between exploration and exploitation can be well maintained. Although Eqs. (9) and (10) of ABC_elite can significantly improve the performance of ABC, the valuable information of the gbest is not fully exploited in Eq. (9). To further improve the performance of ABC by using gbest and get a better exploration–exploitation balance effectively, two novel search Eqs. (19) and (20) are proposed, as follows:

$$\mu = \frac{1}{3} \cdot (x_{best,j} + x_{e,j} + x_{k,j}) \tag{17}$$

$$\delta = \frac{1}{3} \cdot \left(\left| x_{best,j} - x_{e,j} \right| + \left| x_{e,j} - x_{k,j} \right| + \left| x_{best,j} - x_{k,j} \right| \right) \tag{18}$$

$$v_{i,j} = \mu + \phi_{i,j} \cdot \delta \tag{19}$$

$$v_{e,j} = \mu + \phi_{e,j} \cdot \delta \tag{20}$$

where $\phi_{i,j}$ and $\phi_{e,j}$ are random real numbers in the range of $[-1, 1]$; $|\cdot|$ is the absolute value symbol, μ is the base vector, δ is the disturbance vector; x_e is a randomly generated elite solution from the top $p.SN$ solution, $p \in (0, 1)$; x_k is randomly chosen from the current population; $e \neq k \neq i$, x_{best} is the current best solution. Equation (19) is used in the employed bee phase of the proposed algorithm and replace Eq. (9) of ABC_elite; Eq. (20) is used in the onlooker bee phase of the proposed algorithm and replace Eq. (10) of ABC_elite.

In the left-hand side of Eq. (20), only elite solutions have a chance to produce candidates, which is the same as (10) of ABC_elite. By doing so, the computing resources can be focused on elite solutions and the exploitation ability of the algorithm can be enhanced [15]. Herein, the proposed algorithm is called EABC_elite (enhanced ABC_elite). Except for (19) and (20), the rest of EABC_elite is the same as ABC_elite.

3.4 Execution process Of EABC_elite

The pseudocode of the complete EABC_elite is shown in Algorithm 1. In each generation, an employed bee will search the neighbor of a randomly selected solution

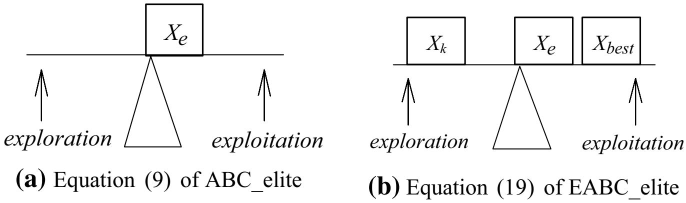

Fig. 1 Balance strategy comparison

x_i and produce a candidate solution v_i according to (19) (line 5) in the employed bee phase. If the candidate solution v_i is better than x_i, v_i will be recorded by its employed bee and replace x_i. (lines 6–7). In the onlooker bee phase, an elite solution x_e is selected randomly to generate a candidate solution v_e by (20). If the candidate solution v_e outperforms x_e, v_e will replace x_e. (lines 15–16). After the employed bee phase and onlooker bee phase, the scout bee phase will begin (lines 22–25). The above three phases will be repeated until the predetermined termination threshold is met. The global best solution which has the smallest objective function value in the final population will be treated as the final optimization results.

3.5 Discussions

In EABC_elite, population information is efficiently utilized to guide the search, as EABC_elite has no bias to any search directions. Therefore, the global search ability is enhanced, and the precocity problem is effectively averted. The global best (gbest) individual x_{best} is introduced to Eq. (19) to accelerate convergence. The ordinary individual x_k is introduced to the search equation to balance the gbest's great leadership ability as well enhance the global search ability of EABC_elite, so the information of x_{best} and ordinary individuals x_k can all be used and the balance between exploration and exploitation can be well maintained.

In Eq. (9) of ABC_elite, the base vector is composed of only one term, the elite solution x_e, while in Eq. (19) of EABC_elite, the base vector is composed of the global best solution x_{best}, the elite solution x_e, and the ordinary solution x_k. Because the global best solution x_{best} has the strongest exploitation ability and the ordinary solution x_k has the strongest exploration ability, x_{best} is "neutralized" by adding x_k. Finally, the proposed algorithm EABC_elite can still maintain a good balance between exploration and exploitation.

The balance strategy of Eqs. (9) and (19) has been shown in Fig. 1a, b. In the latter, although the use of x_{best} can enhance the exploitation ability of EABC_elite greatly, the use of ordinary solution x_k can enhance the exploration ability and help EABC_elite escape from the local minimum. Thus, the balance between exploration and exploitation can be well maintained. Similarly, by using the ordinary solution x_k in the base vector of (20), the global search ability of EABC_elite is enhanced, and the precocity problem is effectively averted. Also, the oscillation phenomenon will be effectively avoided since there is only one guiding term in Eqs. (19) and (20).

4 Experimental results

In this section, three experiments are conducted to compare the proposed EABC_elite with some recently developed ABC and non-ABC variants to validate the performance of EABC_elite. Two classic test suites are used in experiments 1 and 2, the former one is widely adopted by BCABC [12], CABC [14], ABC_elite [15], ABCVSS [30] and ECABC [23], and the latter one is the set of famous test suite (CEC 2015 [25]) that consists of 15 shifted and rotated functions, which is harder to solve compared to the basic functions. Experiment 3 is conducted to test the performance of EABC_elite in solving the feature selection problem, and seven well-know UCI machine learning datasets (http://archive.ics.uci.edu/ml) are selected to this experiment.

EABC_elite is compared to ABCLGII [24], ECABC [23], DGABC [31], ABC_elite [15], and DFSABC_elite [15], since the search equation of the basic ABC algorithm is improved using these methods. For the sake of fairness, the initial population of each algorithm is created randomly according to Eq. (1). Experimental results are shown in Tables 3 and 4.

To show the difference between the EABC_elite and other algorithms, the Wilcoxon [32] rank sum test is carried out for the nonparametric statistics of the independent sample, with the experimental results carried out at the significant level 0.05. That is, the symbols "−," "+," and "=" represent the performance of the corresponding algorithm worse than, better than and similar to that of EABC_elite, respectively, at a 0.05 significance level of Wilcoxon's rank test in Tables 3, 4, 6 and 7. In Tables 3, 4, 6, 7 and 9, the best results are marked in boldface.

Algorithm 1: The pseudo-code of the proposed EABC_elite

```
Initialization: Generate SN solutions that contain D variables according to (1); FEs=0
while FEs<max_FEs
    Select the top T=p.SN solutions as elite solutions from population
    for i=1 to SN        //employed bee phase
        Generate a new candidate solution v_i in the neighborhood of x_i using (19), evaluate the candidate solution v_i
        if f(v_i)< f(x_i)
            Replace x_i by v_i and counter(i)=0
        else
            counter(i)= counter(i)+1
        end if
    end for          //end employed bee phase
    for i=1 to r·p·SN     //onlooker bee phase
        Select a solution x_e from elite solutions randomly to search
        Generate a new candidate solution v_e using (20), evaluate the new candidate solution v_e
        if f(v_e)< f(x_e)
            Replace x_e by v_e and counter(e)=0
        else
            counter(e)= counter(e)+1
        end if
    end for        //end onlooker bee phase
    FEs=Fes+ SN + r·p·SN
    Select the solution x_max with max counter value     //Scout bee phase
    if counter(max)>limit    //only one food source with max counter value can be initialized
            Replace x_max by a new solution generated according to (1), FEs=FEs+1,counter(max)=0;
    end if                 //end scout bee phase
end while
```

4.1 Experiment 1: comparison of state-of-the-art ABCs on benchmark functions

In this section, 22 scalable benchmark functions with dimensions $D=50$ and $D=100$ are used to evaluate the performance of EABC-elite, as shown in Table 1. These functions include continuous, discontinuous, unimodal, and multimodal

Table 1 Benchmark functions used in experiment 1 ($D=50$)

	Function	Search range	Min		Function	Search range	Min
f_1	Sphere	$[-100, 100]^D$	0	f_{12}	NCRastrigin	$[-5.12, 5.12]^D$	0
f_2	Elliptic	$[-100, 100]^D$	0	f_{13}	Griewank	$[-600, 600]^D$	0
f_3	SumSquare	$[-10, 10]^D$	0	f_{14}	Schwefel2.26	$[-500, 500]^D$	0
f_4	SumPower	$[-1, 1]^D$	0	f_{15}	Ackley	$[-50, 50]^D$	0
f_5	Schwefel2.22	$[-10, 10]^D$	0	f_{16}	Penalized1	$[-100, 100]^D$	0
f_6	Schwefel2.21	$[-100, 100]^D$	0	f_{17}	Penalized2	$[-100, 100]^D$	0
f_7	Step	$[-100, 100]^D$	0	f_{18}	Alpine	$[-10, 10]^D$	0
f_8	Exponential	$[-10, 10]^D$	0	f_{19}	Levy	$[-10, 10]^D$	0
f_9	Quartic	$[-1.28, 1.28]^D$	0	f_{20}	Weierstrass	$[-1, 1]^D$	0
f_{10}	Rosenbrock	$[-5, 10]^D$	0	f_{21}	Himmelblau	$[-5, 5]^D$	−78.33236
f_{11}	Rastrigin	$[-5.12, 5.12]^D$	0	f_{22}	Michalewicz	$[0, \pi]^D$	−500, −100

functions. In the search range, the optimal global value of each function is shown in Table 1, and their definitions are found in the literature [15].

The mean value and standard deviation (SD) of the best objective function value are calculated by each algorithm to evaluate the quality or accuracy of the solutions obtained by different algorithms. The smaller the value of this metric is, the higher the quality/accuracy of the solution has. According to [15], the maximum function evaluation (*max_FEs*) is used as the termination condition and set to $5000 \cdot D$; *SN* set to 50 and D set to 50 for all algorithms, note that D represents the number of decision variables. Other parameters are set following the original literature, as shown in Table 2. For each function, all algorithms have a minimum of 30 independent execution runs. Experiment results when $D=50$ and $D=100$ are depicted in Tables 3 and 4, respectively.

In this text, f_1–f_9 are unimodal functions. From Table 3, when solving the unimodal functions f_1, f_2, f_3, f_5, and f_6, the solution accuracy and robustness of the EABC_elite are better than other algorithms except for ECABC, and all algorithms show similar performance on the unimodal functions f_7 and f_8. f_7 is a discontinuous step function which can be easily solved [14], since its optimal global solution is a region rather than a point. Therefore, all algorithms can find the optimal global solution on f_7. Since f_9 is a quartic function with noise, its optimal global solution is complicated to be found. All algorithms can approximate the global optimal solution though cannot find out the real global optimum, despite EABC_elite, ECABC, and DGABC exhibit better solution quality than other competitors. That is, EABC_elite is the second-best algorithm on the unimodal functions f_1–f_9, whereas ECABC performs best among all algorithms. Additionally, the solution quality of EABC_elite on unimodal function is approximately optimal to ECABC. The main reason why EABC_elite and ECABC get the best results on most unimodal functions lies in the search Eqs. (19) and (13) because they utilize the information of gbest to guide the whole population; thus, the convergence speed of EABC_elite and ECABC is enhanced.

Table 2 Parameters setting

Algorithm	Year	Parameters setting	Algorithm	Year	Parameters setting
ABCLGII	2018	$limit = SN \cdot D0$, $r = 1$, $q = 0.2$	ABC_elite	2016	$limit = SN \cdot D0$, $p = 0.10$, $r = 1/p$
ECABC	2018	$limit = SN \cdot D0$, $p = 0.1$, $Dim = 2$	DFSABC_elite	2016	$limit = SN \cdot D0$, $p = 0.10$, $r = 1/p$
DGABC	2016	$limit = SN \cdot D0$, $C = 1.50$, $F = 0.20$, $CR = 0.3$	EABC_elite	–	$limit = SN \cdot D0$, $p = 0.10$, $r = 1/p$

Table 3 The comparison results of ABC variants on 22 test functions at $D=50$

Alg	ABCLGII [24] Mean (SD)	ECABC [23] Mean (SD)	DGABC [31] Mean (SD)	ABC_elite [15] Mean (SD)	DFSABC_elite [15] Mean (SD)	EABC_elite Mean (SD)
f_1	2.13e−92 (5.24e−92)−	**1.19e−99** (5.34e−99)+	5.83e−69 (6.16e−69)−	2.39e−80 (8.28e−80)−	1.71e−83 (5.12e−83)−	8.53e−98 (1.18e−97)
f_2	6.51e−90 (5.82e−90)−	**5.39e−97** (3.92e−97)+	4.38e−65 (1.22e−64)−	1.07e−76 (2.91e−76)−	8.55e−79 (3.56e−79)−	4.39e−95 (3.36e−95)
f_3	3.84e−93 (4.89e−93)−	**3.24e−101** (3.8e−101)+	1.22e−68 (3.85e−68)−	4.61e−80 (7.15e−82)−	2.58e−83 (3.45e−83)−	2.64e−99 (2.32e−98)
f_4	1.34e−145 (3.9e−122)+	**3.67e−239** (4.0e−239)+	8.21e−25 (3.55e−24)−	2.64e−108 (1.1e−108)−	4.66e−110 (8.3e−110)−	3.11e−130 (7.6e−129)
f_5	1.25e−47 (9.22e−47)−	**2.57e−53** (4.82e−53)+	1.23e−41 (1.42e−41)−	1.28e−40 (1.07e−40)−	1.53e−42 (4.81e−42)−	5.74e−51 (3.84e−51)
f_6	1.31e+00 (1.82e+00)−	**1.21e−02** (3.03e−02)+	9.23e+00 (3.56e+00)−	7.20e−01 (9.22e−02)−	7.44e−01 (3.23e−01)−	3.30e−01 (5.09e−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)
f_8	**2.67e−109** (8.1e−125)=	**2.67e−109** (2.2e−125)=	**2.67e−109** (4.8e−125)=	**2.67e−109** (9.6e−125)=	**2.67e−109** (8.3e−122)=	**2.67e−109** (2.6e−109)
f_9	3.06e−02 (2.82e−02)−	1.01e−02 (3.99e−02)+	**8.52e−03** (2.15e−03)+	2.52e−02 (4.96e−03)−	2.35e−02 (3.22e−03)−	1.89e−02 (4.70e−03)
f_{10}	**3.32e−02** (9.82e−02)+	7.82e+00 (4.92e+00)−	7.56e+01 (3.87e+01)−	1.83e+00 (1.02e+00)−	1.43e+00 (5.72e+00)−	5.21e−01 (1.12e+00)
f_{11}	**0.00e+00** (0.00e+00)=	4.31e−01 (3.28e−01)−	**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)=	6.23e−02 (5.80e−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)
f_{13}	**0.00e+00** (0.00e+00)=	8.82e−03 (3.37e−03)−	**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}	3.81e−11 (2.21e−11)−	2.42e−02 (3.40e−2)−	1.84e−11 (5.31e−11)+	5.79e−11 (6.56e−11)−	2.42e−11 (4.72e−11)=	**2.32e−11** (2.01e−11)
f_{15}	6.48e−14 (3.82e−15)−	6.22e−14 (7.91e−14)−	1.53e−14 (4.88e−14)+	4.73e−14 (4.46e−15)=	**4.68e−14** (5.98e−14)=	**4.68e−14** (4.03e−15)
f_{16}	**9.42e−33** (5.88e−33)=	**9.42e−33** (3.92e−48)=	**9.42e−33** (8.78e−48)=	**9.42e−33** (2.78e−48)=	**9.42e−33** (2.22e−48)=	**9.42e−33** (1.44e−48)
f_{17}	**1.50e−33** (3.82e−32)=	**1.50e−33** (4.93e−32)=	4.22e−32 (4.67e−32)−	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)
f_{18}	1.08e−15 (2.89e−16)+	3.17e−10 (6.37e−10)−	1.69e−15 (7.82e−15)−	5.55e−17 (1.61e−16)+	**2.17e−17** (4.43e−16)+	7.23e−17 (9.24e−16)
f_{19}	**1.35e−31** (2.82e−31)=	**1.35e−31** (4.89e−32)=	**1.35e−31** (1.64e−32)=	**1.35e−31** (6.68e−47)=	**1.35e−31** (5.80e−47)=	**1.35e−31** (4.34e−48)
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)
f_{21}	− **78.332** (2.83e−14)=	− **78.332** (3.80e−14)=	− **78.332** (1.84e−14)=	− **78.332** (1.65e−14)=	− **78.332** (4.32e−14)=	− **78.332** (2.89e−15)
f_{22}	− 49.61 (2.83e−02)+	− 49.20 (4.82e−02)−	− 49.37 (8.27e−02)−	− 49.87 (3.33e−02)+	− 49.92 (1.34e−02)+	− **49.53** (3.05e−02)
+/=/−	4/10/8	7/7/8	3/9/10	2/11/9	2/12/8	–

Table 4 The comparison results of ABC variants on 22 test functions at $D=100$

Alg	ABCLGII Mean (SD)	ECABC Mean (SD)	DGABC Mean (SD)	ABC_elite Mean (SD)	DFSABC_elite Mean (SD)	EABC_elite Mean (SD)
f_1	7.85e−88 (2.99e−88)−	**3.59e−97** (3.73e−97)+	8.24e−52 (3.32e−52)−	1.03e−79 (4.23e−79)−	1.50e−81 (1.72e−81)−	2.56e−96 (6.44e−96)
f_2	1.23e−84 (4.97e−84)−	**3.30e−94** (7.11e−94)+	7.78e−50 (8.23e−50)−	9.74e−75 (8.22e−75)−	3.33e−77 (9.54e−77)−	3.01e−92 (7.08e−92)
f_3	7.89e−88 (3.36e−88)−	**1.66e−92** (2.82e−92)+	8.22e−49 (2.82e−48)−	1.43e−79 (9.18e−79)−	5.04e−82 (4.89e−82)−	1.15e−88 (5.17e−88)
f_4	3.66e−140 (1.6e−140)+	**1.70e−230** (6.8e−230)+	3.88e−24 (8.8e−24)−	7.83e−106 (4.9e−106)−	1.65e−106 (5.2e−106)−	3.66e−130 (1.6e−129)
f_5	1.33e−46 (9.05e−47)−	**1.00e−53** (5.29e−53)+	8.23e−40 (7.89e−40)−	6.83e−39 (7.82e−39)−	5.10e−42 (3.56e−42)−	3.99e−50 (3.64e−50)
f_6	2.02e+00 (3.46e−01)+	**7.23e−01** (2.37e−01)+	2.03e+01 (3.83e+01)−	4.54e+00 (4.20e−01)−	4.50e+00 (3.63e−01)−	2.81e+00 (2.21e−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	**7.12e−218** (0.0e+00)=	**7.12e−218** (0.0e+00)=	**7.12e−218** (0.0e+00)=	**7.12e−218** (0.0e+00)=	**7.12e−218** (0.0e+00)=	**7.12e−218** (0.0e+00)
f_9	6.83e−02 (5.67e−03)−	4.07e−02 (3.92e−02)=	9.82e−02 (3.77e−02)−	5.55e−02 (4.81e−02)−	5.29e−02 (4.71e−03)−	**4.07e−02** (7.10e−03)
f_{10}	9.23e−01 (5.65e−01)−	3.73e+01 (1.24e+01)−	3.83e+02 (8.82e+02)−	4.56e+00 (3.13+01)−	8.65e+00 (2.22e+01)−	**1.63e−01** (3.08e−01)
f_{11}	**0.00e+00** (0.00e+00)=	9.92e−01 (7.84e−01)−	**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)=	8.31e−01 (3.02e−01)−	**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)=	8.72e−14 (2.78e−14)−	**0.00e+00** (0.00e+00)=	**0.00e+00** (0.00e+00)=	**0.00e+00** (0.00e+00)
f_{14}	7.32e−02 (1.35e−02)−	1.87e−01 (2.40e−01)−	6.45e−10 (3.81e−10)−	9.47e−10 (9.99e10)−	1.12e−10 (6.84e−12)−	**1.09e−10** (3.87e−12)
f_{15}	1.39e−13 (6.45e−14)−	4.35e−13 (1.83e−13)−	2.89e−07 (3.88e−07)−	1.10e−13 (8.23e−13)−	1.09e−13 (6.52e−15)−	**9.27e−14** (8.41e−15)
f_{16}	**4.71e−33** (3.81e−33)=	**4.71e−33** (4.27e−33)=	1.37e−02 (1.92e−02)−	**4.71e−33** (5.28e−33)=	**4.71e−33** (7.21e−49)=	**4.71e−33** (1.40e−48)
f_{17}	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)=	**1.50e−33** (0.00e+00)
f_{18}	4.45e−14 (4.34e−15)−	4.07e−07 (6.93e−07)−	7.33e−12 (4.29e−12)−	3.44e−15 (5.72e−15)−	**5.94e−16** (6.87e−15)+	1.16e−15 (3.22e−15)
f_{19}	**1.35e−31** (0.00e+00)=	**1.35e−31** (0.00e+00)=	4.72e−31 (4.18e−31)−	**1.35e−31** (0.00e+00)=	**1.35e−31** (0.00e+00)=	**1.35e−31** (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)
f_{21}	**−78.332** (7.21e−14)=	**−78.332** (7.71e−02)=	−78.27 (5.28e−02)−	**−78.332** (4.28e−14)=	**−78.332** (5.71e−14)=	**−78.332** (2.01e−14)
f_{22}	**−99.60** (5.23e−02)+	−98.22 (2.46e−01)−	−98.32 (1.07e−01)−	−99.54 (3.92e−02)+	−99.52 (1.42e−02)+	−99.41 (2.08e−02)
+/=/−	3/10/9	6/9/7	0/6/16	1/10/11	2/10/10	

Still, in this text, f_{10}–f_{22} are multimodal functions. As f_{10} is Rosenbrock function and its global optimum is inside of a long and parabolic shaped valley, the variables are strongly dependent, as the gradients do not point toward the optimum. Generally speaking, if the population evolves under the guidance of the global best solution or some other good solutions, the search is easy to get into hopeless areas. Therefore, except for ABCLGII and EABC_elite, all other algorithms perform poorly on f_{10}, since the search equation utilizes the information of gbest to guide search direction. In ABCLGII, only elite individuals have chances to be guided by the information of gbest at a probability P_{str} ($0 < P_{str} < 1$); hence, ABCLGII may obtain better results than BCABC, DGABC, ABC_elite, and DFS-ABC_elite on f_{10}. Although all individuals in EABC_elite are guided by gbest in the employed bee and onlooker bee phases, the ordinary solution x_k is also used to diminish the great lead ability of gbest (see Fig. 1) and help the algorithm escape the local optima. Therefore, the EABC_elite can achieve a better balance between exploration and exploitation and produce the second-best result on function f_{10}. Although ECABC performs very well on unimodal functions f_1–f_9, it performs poorly on multimodal function f_{10} since ECABC only searches around the elite individuals according to Eq. (13), so the exploitation ability of ECABC is too strong and easy to result in precocity problem.

Similarly, regarding the accuracy and reliability of the multimodal functions f_{11}–f_{22}, the EABC_elite is superior to or at least comparable to the compared EAs while ECABC performs poorly on most of the multimodal functions, owing to its excessive exploitation ability.

Overall, EABC_elite outperforms ABCLGII, ECABC, DGBAC, ABC_elite, and DFSABC_elite on 8, 8, 10, 9, and 8 out of 22 functions, respectively. EABC_elite is beaten by ABCLGII, ECABC, DGBAC, ABC_elite, and DFSABC_elite on 4, 7, 3, 2, and 2 functions, respectively. Although ECABC performs better on unimodal functions f_1–f_9, EABC_elite shows robust results on both unimodal and multimodal functions. Comparison results between EABC_elite and other ABC variants on 22 test functions at $D = 100$ are shown in Table 4, and a similar conclusion is sought. As overall, due to the superior design of search equations, the EABC_elite shows the best overall performance among all six ABC variants.

To further verify the performance of EABC_elite, we compare EABC_elite on aforementioned 22 benchmark functions at $D = 40$ with five most widely used DE and PSO variants, i.e., SRPSO [33], SLPSO [28], JADE [34], sinDE [35], and ABCADE [36]. As the comparison, the parameters of all DE and PSO methods are set following the corresponding original papers, and parameter setting details of all DE and PSO methods are tabulated in Table 5. Experimental results of "mean" and "SD" are given in Table 6, from which we can observe that EABC_elite outperforms SRPSO, SLPSO, JADE, sinDE, and ABCADE on 21, 14, 12, 13, and 7 out of 22 functions and is beaten solely by SRPSO, SLPSO, JADE, sinDE, and ABCADE on 1, 2, 2, 2, and 3 functions, respectively. Therefore, EABC_elite performs better than all other algorithms both on unimodal functions and on multimodal functions due to its excellent exploration–exploitation balance.

Table 5 The parameters of EABC_elite and non-ABC variants

Algorithm	Year	Parameters setting	Algorithm	Year	Parameters setting
SRPSO	2015	$N=400$, $w_{\text{initial}}=1.050$, $w_{\text{final}}=0.5$, $c_1=c_2=1.49445$, $V_{\max}=0.06708\times\text{Range}$	sinDE	2015	$NP=40$, $freq=0.25$
SLPSO	2015	All parameters depend on the function dimension D.	ABCADE	2017	$SN=50$, $limit=200$, $m=5$, $n=10$, $c_1=0.9$, $c_2=0.999$
JADE	2009	$NP=100$, $c=0.1$, $p=0.05$	EABC_elite	–	$SN=50$, $limit=200$, $p=0.10$, $r=1/p$

Table 6 Comparison of EABC_elite with non-ABC variants on 22 test functions at $D=40$

	SRPSO mean (SD)	SLPSO mean (SD)	JADE mean (SD)	SinDE mean (SD)	ABCADE mean (SD)	EABC_elite mean (SD)
f_1	3.91e−73 (1.22e−73)−	1.41e−71 (2.08e−71)−	1.11e−76 (3.98e−76)−	1.33e−54 (1.37e−54)−	4.30e−70 (1.59e−69)−	**1.65e−98 (4.05e−98)**
f_2	4.49e−77 (1.47e−77)−	2.68e−68 (2.35e−68)−	1.35e−65 (6.69e−65)−	1.66e−51 (1.61e−51)−	.54e−64 (1.84e−63)−	**8.35e−96 (1.61e−95)**
f_3	2.74e−75 (9.87e−75)−	1.04e−72 (1.07e−72)−	7.20e−74 (3.45e−73)−	2.30e−55 (2.12e−55)−	1.77e−71 (5.77e−71)−	**1.15e−99 (5.01e−99)**
f_4	**3.30e−206 (0.00e+00)+**	5.30e−163 (2.22e−162)+	1.80e−92 (7.39e−92)−	4.67e−86 (2.33e−85)−	8.86e−172 (0.00e+00)+	3.60e−122 (1.94e−121)
f_5	1.10e−22 (4.79e−22)−	2.35e−37 (2.93e−37)−	1.03e−30 (4.24e−30)−	1.87e−31 (1.02e−31)−	1.03e−42 (1.90e−42)−	**3.96e−51 (5.22e−51)**
f_6	1.27e+00 (2.04e+00)−	**8.32e−16 (1.23e−15)+**	3.62e−14 (4.90e−14)+	2.38e−02 (9.26e−02)+	1.02e−03 (1.95e−03)+	2.28e−01 (1.72e−01)
f_7	1.01e+01 (1.31e+01)−	**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	6.00e−71 (2.68e−71)−	**1.38e−87 (9.77e−103)=**	**1.38e−87 (4.56e−103)=**	**1.38e−87 (1.16e−93)=**	**1.38e−87 (4.56e−103)=**	**1.38e−87 (4.56e−103)**
f_9	5.74e−02 (4.26e−02)−	2.57e−02 (2.6e−03)−	**1.30e−03 (3.82e−04)+**	6.66e−03 (1.81e−03)+	1.33e−02 (2.57e−03)−	1.17e−02 (5.29e−03)
f_{10}	3.52e+01 (4.19e+01)−	2.61e+01 (2.63e+01)−	6.38e−01 (1.49e+00)−	3.27e+01 (5.98e−01)−	1.97e+01 (3.16e+01)−	**4.56e−01 (1.45e−01)**
f_{11}	4.63e+01 (1.03e+01)−	1.96e+01 (4.19e+00)−	3.33e−11 (2.94e−11)−	1.60e+02 (8.49e+00)−	**0.00e+00 (0.00e+00)=**	**0.00e+00 (0.00e+00)**
f_{12}	6.15e+01 (1.46e+01)−	3.31e+01 (5.30e+00)−	2.86e−08 (1.69e−08)−	1.23e+02 (1.06e+01)−	**0.00e+00 (0.00e+00)=**	**0.00e+00 (0.00e+00)**
f_{13}	1.74e−16 (2.64e−16)−	**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}	1.69e+03 (5.67e+02)−	2.00e+03 (3.74e+02)−	4.74e+00 (2.39e+01)−	1.46e+03 (1.60e+03)−	7.27e−12 (0.00e+00)−	**6.06e−13 (1.37e−12)**
f_{15}	3.47e−08 (5.11e−08)−	7.99e−14 (1.37e−15)−	5.40e−15 (1.81e−15)=	6.54e−15 (1.33e−15)=	5.25e−15 (1.81e−15)=	**2.22e−15 (1.15e−15)**
f_{16}	5.19e−30 (6.22e−30)−	**1.18e−32 (2.79e−48)=**	**1.18e−32 (2.79e−48)=**	**1.18e−32 (2.79e−48)=**	**1.18e−32 (2.79e−48)=**	**1.18e−32 (2.79e−48)**
f_{17}	8.22e−30 (7.38e−30)−	**1.50e−33 (0.00e+00)=**	**1.50e−33 (0.00e+00)=**	1.55e−33 (2.47e−34)=	1.55e−33 (2.47e−34)=	**1.50e−33 (0.00e+00)**
f_{18}	1.85e−13 (8.30e−13)−	8.32e−17 (1.97e−16)=	1.19e−04 (6.37e−05)−	1.87e−02 (3.68e−03)−	**4.04e−40 (1.40e−39)+**	5.75e−17 (3.15e−06)
f_{19}	6.17e−30 (4.22e−30)−	9.10e−31 (2.84e−31)−	**1.35e−31 (2.23e−47)=**	**1.35e−31 (2.23e−47)=**	**1.35e−31 (2.23e−47)=**	**1.35e−31 (2.23e−47)**
f_{20}	1.02e+00 (2.41e+00)−	3.30e−02 (3.33e−02)−	2.25e−01 (2.26e−02)−	9.19e−05 (4.59e−04)−	**0.00e+00 (0.00e+00)=**	**0.00e+00 (0.00e+00)**
f_{21}	−74.728 (2.19e+00)−	−74.56 (1.23e+00)−	**−78.332 (0.00e+00)=**	−78.304 (1.41e−01)−	**−78.332 (6.48e−15)=**	**−78.332 (2.38e−14)**
f_{22}	−22.38 (5.52e−01)−	−20.75 (4.20e−01)−	−39.882 (2.14e−02)−	−29.613 (8.76e−01)−	**−40.00 (0.00e+00)=**	**−40.00 (0.00e+00)**
+/=/−	1/0/21	2/6/14	2/8/12	2/7/13	3/12/7	

4.2 Experiment 2: comparison of state-of-the-art ABCs on CEC 2015 functions

In this subsection, the performance of the proposed algorithm EABC_elite is tested by solving a set of problems taken from the CEC2015 competition on learning-based real-parameter single objective optimization [25]. The CEC2015 benchmark contains 15 shifted or rotated problems, which are very difficult to solve when compared to basic functions. In this subsection, functions F_1–F_2 are unimodal, F_3–F_5 multimodal, F_6–F_8 hybrid and F_9–F_{15} are composite functions, and the search space of each problem is $[-100, 100]^D$. We evaluated the procedures of the CEC2015 benchmark competition, and results are obtained based on 51 independent runs with 10000.D function evaluations (*max_FEs*) as the termination criterion for each test function, the error value of the found solution is defined as $(f(x) - f(x^*))$, where x^* is the optimum value of the function. As a threshold, error values lower than 10^{-8} (zero-threshold) are approximated to zero.

The population size is set to 100, so the parameter $SN = 0.5 \times$ population size $= 50$. For all the algorithms, D is set to 30, and other parameters are shown in Table 2.

The mean error and standard deviation (SD) of the best objective function value are calculated by each algorithm to evaluate the quality or accuracy of the solutions obtained by different algorithms. The smaller the value of this metric is, the higher the quality/accuracy of the solution has. From Table 7, EABC_elite is the second-best algorithms on unimodal function F_1, and ECABC ranks first among all the algorithms. On function F_2, EABC_elite has significant advantages over all other algorithms. The reason is that Eq. (19) is guided by the gbest, and thus, the exploitation ability of ABC is enhanced, which is beneficial to unimodal functions.

F_3–F_{15} are complicated multimodal functions with numerous local minima. As known, an algorithm should own strong global search ability to produce good results; otherwise, the algorithm may fall fastly into a local minimum. From Table 7, the EABC_elite performs significantly better than all compared algorithms regarding solution accuracy and robustness on almost all the test functions. On all 15 functions, EABC_elite is beaten by ABCLGII, ECABC, DGABC, ABC_elite, and DFSABC_elite only on 2, 2, 2, 1, and 2 functions, respectively. The reason is that EABC_elite has no bias to any search directions and the global search ability of EABC_elite is relatively strong. Although ECABC performs well on 22 test functions above discussed, it performs poorly on CEC 2015 functions due to ECABC always searches around elite individuals so that the exploitation ability of ECABC is too strong and easy to result in precocity problem. Observing experiments 1 and 2, since the EABC_elite uses stronger heuristic information and better balance strategy simultaneously, the overall performance of EABC_elite is better than all other algorithms regarding solution quality and robustness. For the convenience and clearness of illustration, the convergence curves of six representative functions are plotted in Fig. 2, where EABC_elite exhibits faster convergence speed than most of the competitors.

Table 7 The mean error and standard deviation of six ABCs on CEC 2015 test function suite at $D=30$

Alg	ABCLGII Mean error (SD)	ECABC Mean error (SD)	DGABC Mean error (SD)	ABC_elite Mean error (SD)	DFSABC_elite Mean Error (SD)	EABC_elite MeanError (SD)
F_1	2.32e+06 (1.01e+06)−	**2.01e+06** (9.07e+05)+	3.10e+06 (1.53e+06)−	2.97e+06 (1.51e+06)−	3.20e+06 (1.08e+06)−	2.27e+06 (9.86e+05)
F_2	2.61e+03 (3.32e+03)−	2.44e+03 (3.20e+03)−	2.51e+03 (2.34e+03)−	2.67e+03 (3.59e+03)−	1.64e+03 (2.03e+03)−	**1.20e+03** (2.26e+03)
F_3	2.01e+01 (2.74e−01)−	2.03e+01 (3.52e−02)−	2.01e+01 (3.91e−02)−	2.01e+01 (3.96e−02)−	2.01e+01 (4.61e−02)−	**2.00e+01** (3.83e−02)
F_4	4.96e+01 (8.52e+00)−	3.88e+01 (5.47e+00)−	4.09e+01 (6.79e+00)−	4.24e+01 (7.09e+00)−	4.21e+01 (6.8e+00)−	**3.73e+01** (6.40e+00)
F_5	1.81e+03 (3.38e+02)−	1.83e+033.67e+02)−	1.83e+03 (2.06e+02)−	1.69e+03 (2.58e+02)−	1.68e+03 (2.47e+02)−	**1.67e+03** (2.40e+02)
F_6	1.03e+06 (5.54e+05)−	8.03e+05 (4.36e+05)−	8.01e+05 (5.80e+05)−	1.07e+06 (5.78e+05)−	1.03e+06 (5.58e+05)−	**6.34e+05** (4.65e+05)
F_7	8.46e+00 (1.30e+00)−	8.68e+00 (1.65e+00)−	1.01e+01 (7.93e−01)−	8.32e+00 (1.41e+00)−	8.52e+00 (1.31e+00)−	**7.96e+00** (1.30e+00)
F_8	1.73e+05 (1.11e+05)+	**1.55e+05** (1.01e+05)+	1.73e+05 (8.95e+04)+	2.74e+05 (1.69e+05)−	2.36e+05 (1.36e+05)+	2.50e+05 (1.33e+05)
F_9	**1.03e+02** (2.77e−01)=	**1.03e+02** (2.02e−01)=	**1.03e+02** (1.72e−01)=	1.04e+02 (2.24e−01)−	1.04e+02 (2.63e−01)−	**1.03e+02** (2.41e−01)
F_{10}	4.77e+05 (2.63e+05)−	4.20e+05 (2.43e+05)−	4.17e+05 (2.34e+05)−	6.31e+05 (4.50e+05)−	6.99e+05 (5.80e+05)−	**4.09e+05** (5.26e+05)
F_{11}	3.49e+02 (9.39e+01)−	3.93e+02 (1.21e+02)−	4.50e+02 (1.49e+02)−	3.43e+02 (8.09e+01)−	3.65e+02 (1.30e+02)−	**3.28e+02** (9.80e+01)
F_{12}	**1.04e+02** (4.08e−01)=	**1.04e+02** (3.22e−01)=	1.05e+2 (4.76e−01)−	1.05e+02 (3.37e−01)−	1.05e+02 (6.69e−01)−	**1.04e+02** (3.20e−1)
F_{13}	2.62e−02 (3.61e−04)+	2.65e−02 (4.31e−04)=	**2.61e−02** (7.67e−04)+	2.63e−02 (3.38e−04)+	2.63e−02 (3.87e−04)+	2.65e−02 (5.12e−04)
F_{14}	3.22e+04 (9.34e+02)−	3.28e+04 (8.50e+02)−	3.28e+04 (8.23e+02)−	3.21e+04 (8.92e+02)−	3.21e+04 (9.64e+02)−	**3.19e+04** (7.51e+02)
F_{15}	**1.00e+02** (3.11e−12)=	**1.00e+02** (4.22e−14)=	**1.00e+02** (7.22e−14)=	**1.00e+02** (6.83e−14)=	**1.00e+02** (4.25e−13)=	**1.00e+02** (3.23e−14)
+/=/−	2/3/10	2/4/9	2/2/11	1/1/13	2/1/12	–

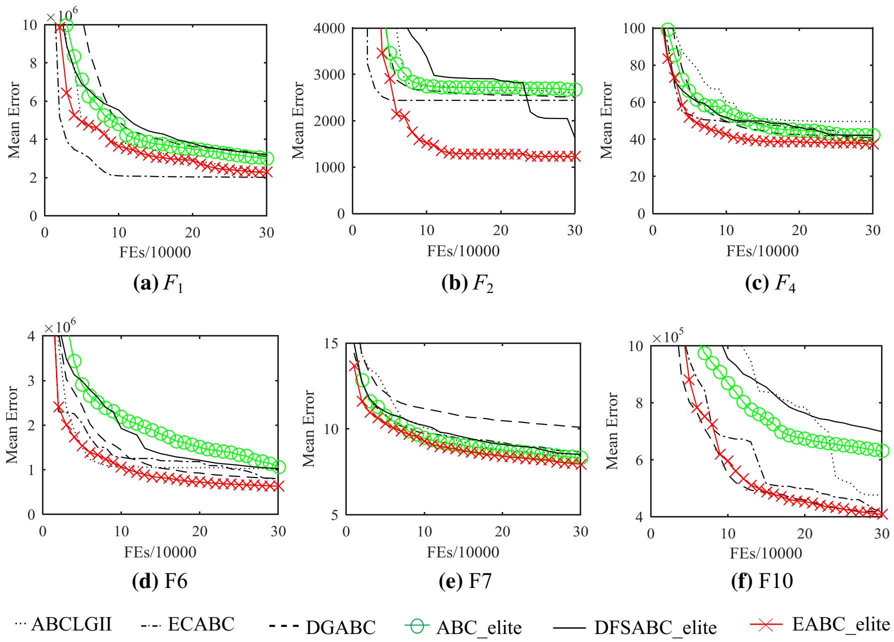

Fig. 2 Convergence curves of different ABCs on six CEC 2015 functions

4.3 Experiment 3: feature selection problem

Feature selection (FS) technology is an important step when extracting a subset of useful features subset and discarding irrelevant features of a given dataset [37]. It is a preprocessing step to solve the concerns of classification problems in recent years [38]. All features of a given data set may include noise, redundant, or misleading information, so exhaustive search strategy applied to all features should be a time-consuming process, that is unrealistic in the real world. Based on this consideration, we apply ABC variant that aims at the optimization algorithm to search the optimal subset d of related features from the original feature set D ($d < D$), to shorten the calculation time and obtain higher classification accuracy.

4.3.1 Individual encoding

Binary vectors are contemporary techniques in the feature selection problem [39], where 1 represents that the corresponding feature is selected, and 0 represents that the corresponding feature is not selected. According to the literature [39], each element of an individual is limited to [0, 1] that represents the probability of the related feature to be selected. Taking the dataset with D features as an example, an individual can be encoded as

$$X_i = (x_{i,1}, x_{i,2}, \dots x_{i,D}) \tag{21}$$

The corresponding feature subset S_i can be generated by

$$s_{i,j} = \begin{cases} 1 & rand < x_{i,j} \\ 0, & otherwise \end{cases} \tag{22}$$

where $j = 1, 2, \ldots, D$; *rand* denotes a randomly generated random number in the range of [0, 1].

4.3.2 Fitness evaluation

The K-nearest neighbor (KNN) is a simple yet efficient classifier used to evaluate the performance of each individual. In this section, the parameter k of KNN is set to 1.

The tenfold cross-validation method is used to train and test the KNN classifier, where the dataset is divided into ten un-duplicated subsets, and any nine of the ten subsets are used for training and the remaining one for testing.

Herein, the classifier will be trained and tested ten times. Note that, the satellite dataset cannot be tested under the tenfold cross-validation method since the dataset has been divided into testing and training dataset.

In EABC_elite for feature selection, the classification accuracy obtained by the i-th individual (food source) X_i is calculated as the proportion of correctly determined instances to all instances, shown as

$$\text{Accuracy}_i = \frac{\text{Number of correctly determined samples}}{\text{Total number of all the samples}} \tag{23}$$

$$f(X_i) = 1 - \text{Accuracy}_i \tag{24}$$

For each X_i, a subset of d relevant features from the original feature set D ($d < D$) is generated according to (21) and (22), then the KNN classification is used to classify the dataset with selected d features. Next, the classification accuracy is computed by Eq. (23), in which the higher the accuracy, the better is the selected subset performance. At last, since the EABC_elite algorithm is proposed to solve the minimization problem, though Eq. (23) is a maximization problem, and thus, Eq. (24) is used to transfer the maximization problem into minimization problem, and the value $f(X_i)$ is the objective function of the i-th food source X_i.

4.3.3 Experiments of feature selection problem

In this section, the EABC_elite-based feature selection method is evaluated and compared with DE [2], ABC [5], CBPSO1 [37], and NSABC [39] algorithms, and experimental results are taken from [39].

Three groups of datasets in this feature selection problem are applied, cited from the UCI repository. In this paper, we apply three well-known datasets in the UCI repository[1] to study the problem of feature selection. For the features between 10

[1] http://archive.ics.uci.edu/ml.

Table 8 The datasets used in feature selection problem

Dataset	No. of Samples	No. of classes	No. of features
Glass	214	7	10
Wine	178	3	13
Letter	20,000	26	16
Segmentation	2310	7	19
Ionosphere	351	2	34
Satellite	6435	6	36
Sonar	208	2	60

and 19, it is considered as a small group. This group contains glass, wine, letter, and segmentation. If the number of features is between 20 and 49, it is considered a medium size group. Say, the ionosphere and the satellite are in this group. Finally, if the number of features is higher than 50, it is looked on as a large group, e.g., sonar is in the large group. Table 8 gives a detailed description of these datasets.

The normalization is a favorite preprocessing step, as all features are normalized by projecting their feature value to the interval [0, 1] to diminish the significant impact of great numbers [40]. As a comparison, the population size of the EABC_elite algorithm is set to 20, which is the same as the literature [39], and the parameter p in EABC_elite is set to 0.3. Given that the maximum iteration number in most feature selection studies [37–39] is set to 100, this paper also utilizes the same maximum iteration number. Note that the maximum number of functions is related to the population size and the maximum iterations (MAX_ITER), and expressed as: population size $\times$ MAX_ITER $= 2 \times SN \times$ MAX_ITER. As shown in Table 9, we can observe that EABC_elite is the best feature selection method in all compared algorithms, and the classification accuracies of EABC_elite on the wine, letter, segmentation, satellite and sonar datasets are 99.85%, 85.67%, 98.23%, 91.59%, and 92.06%, respectively, which is better than other methods. On the glass dataset, the EABC_elite performs as well as other algorithms. The proposed EABC_elite ranks fourth on the ionosphere dataset and the ECABC ranks the first. As seen from the experimental results, the proposed EABC_elite is an efficient tool for feature selection.

5 Data clustering

In this section, the proposed EABC_elite is modified by embedding the K-means initialization strategy and chaotic parameters strategy to solve the clustering problem, to further verify its superiority.

5.1 Description of the clustering problem

Clustering is an essential tool for many applications such as data mining, statistical data analysis, data compression, and vector quantization [41, 42]. The purpose of clustering

Table 9 The results of feature selection (d: the selected feature numbers; Acc: the classification accuracy)

Dataset	DE d (Acc)	ABC d (Acc)	CBPSO1 d (Acc)	BBPSO d (Acc)	NSABC d (Acc)	ECABC d (Acc)	EABC_elite d (Acc)
Glass	4.6 (**100%**)	6 (**100%**)	4.2 (**100%**)	4 (**100%**)	4 (**100%**)	4 (**100%**)	4.3 (**100%**)
Wine	8 (99.10%)	8 (99.10%)	8 (99.33%)	8 (98.88%)	7.8 (99.78%)	7.6 (99.80%)	7.2 (**99.85%**)
Letter	11 (80.50%)	9.8 (79.84%)	9.6 (79.33%)	9.6 (79.80%)	10 (80.67%)	10.7 (81.16%)	10.4 (**85.67%**)
Segmentation	10.8 (98.11%)	13.8 (98.00%)	12.6 (98.04%)	10.8 (98.05%)	11.4 (98.11%)	10.3 (98.09%)	9.3 (**98.23%**)
Ionosphere	14 (94.81%)	16.6 (94.53%)	15 (94.64%)	11.6 (94.70%)	15 (94.87%)	12.7 (**94.90%**)	14.2 (94.71%)
Satellite	24 (90.51%)	22.2 (90.51%)	20.6 (90.26%)	18.2 (90.42%)	20.8 (90.54%)	19.6 (90.46%)	20.2 (**91.59%**)
Sonar	28.8 (91.83%)	28.8 (91.83%)	30.4 (91.63%)	25.6 (91.54%)	32 (91.92%)	32 (91.96%)	31.6 (**92.06%**)

is to gather data into clusters (or groups), so that the similarity of data in each cluster is highly similar while being very dissimilar to data from other clusters [43].

There are two main classes of clustering techniques: hierarchical clustering and partitioning clustering. The time complexity of the hierarchical clustering is quadratic, whereas it is almost linear in the partitioning approaches, the reason why the partitioning approaches are widely used rather than hierarchical ones [44]. In a partitional clustering problem [45], we need to divide a set of n objects into k clusters [46]. Let $O(o_1, o_2,\ldots o_n)$ be the set of n objects. Each object has q characters, and each character is quantified with a real value. Let $X_{n\times q}$ be the character data matrix. It has n rows and q columns. Each row represents data and $x_{i,j}$ represents the j-th feature of the i-th data ($i=1, 2,\ldots, n, j=1, 2,\ldots, q$).

Let $C = (C_1, C_2, \ldots C_k)$ be the k clusters. Then:

$$C_i \neq \emptyset,\ C_j \cap C_i \neq \emptyset,\ C_1 + C_2 + \cdots + C_k = O, \quad i,j = 1,2,\ldots,k, i \neq j$$

The goal of the clustering algorithm is to find such a C, so that objects in the same cluster can be as similar as possible, while objects in different clusters are different. These can be measured by some standards, such as total cluster variance or total mean square error (MSE) [47]:

$$Perf(O,C) = \sum_{i=1}^{n} Min\left\{ ||o_i - c_j||^2, j = 1,2,\ldots,k \right\} \tag{25}$$

where $||o_i - c_j||^2$ represents the similarity between the i-th object and the center of j-th cluster. The most popularly used similarity metric in clustering is Euclidean distance, which is derived from the Minkowski metric:

$$d(o_i, c_j) = \left(\sum_{m=1}^{p} (x_{im} - c_{jm})^r \right)^{1/r} \tag{26}$$

where c_j is the center of j-th cluster C_j and m is the dimension within q. In this study, we will use the Euclidean metric as a distance metric, i.e., $r=2$ in Eq. (26). K-means clustering is one of the most popular partitional clustering algorithms due to its simplicity and linear time complexity. The main steps of the K-means algorithm are given below.

Initialize the k number of cluster centers ($C_1, C_2,\ldots, C_k$) from the data points $\{X_1, X_2,\ldots X_N\}$ in random,

Assign the data points X_i, where $i=1, 2, 3,\ldots, N$ to cluster center $j=1, 2, 3,\ldots, k$, such that $X_i - C_j \leq X_i - C_l$, $l=1, 2, 3,\ldots k$ and $l\neq j$, where $X_i - C_j$ is the Euclidean distance between data points X_i and cluster center C_j.

Compute the new cluster centers $C'_1, C'_2, \ldots, C'_k$ as follows:

$$C'_j = \frac{1}{M_j} \sum_{X_i \in C_j} X_i, \quad j = 1,2,3,\ldots k \tag{27}$$

where M_j indicates the number of data points related to cluster C_j.

Replace each C_j with C_j', $j = 1, 2, \ldots, k$, until $C_j \neq C_j'$.

As a result for the multi-step K-means algorithm, k number of cluster centers positions are obtained and represented as the possible locations of the food source in D-dimensional search space for the employed bee phase of the ABC algorithm.

5.2 Traditional ABC-based clustering

From the view of optimization, clustering N objects to k clusters is a typical NP-hard problem [45], given that the swarm intelligent evolution algorithms have advantages in solving the NP-hard problem, a large number of EAs have been applied to the clustering problem [40, 45, 47]. It is easy to apply ABC variants for data clustering, as two changes are needed to be done for this approach according to the literature [46], as detailed in 5.2.1 and 5.2.2.

5.2.1 Solution presentation

In the numerical optimization of ABC, each food source represents a solution to the problem. When clustering in ABC, each food source represents a set of clusters, shown as

$$X_i = \{x_1, x_2, \ldots, x_q, x_{q+1}, \ldots, x_{k\times q}\} \tag{28}$$

$$c_m = \left\{x_{(m-1)\times q+1}, x_{(m-1)\times q+2}, \ldots x_{m\times q}\right\} \tag{29}$$

where X_i represents a food source in the ABC algorithm, k is the number of clusters, and q the number of features for the data clustering problem, for k centers clustering problem with q characters, the real dimension of ABC is $k \times q$.

There is no relationship between the population size of the ABC algorithm and the clustering problem. First, the upper and lower bounds on each feature are obtained by scanning the clustering data. At the initialization phase and scout bee phase, when the new food source is generated, the value on the j-th dimension should be restricted to the boundary of the l-th feature, where l is calculated as

$$l = \mathrm{mod}((j-1), q) + 1 \tag{30}$$

5.2.2 Fitness calculation

Unlike solving numerical optimization problems, the total within-cluster variance in Eq. (25) is employed to evaluate the quality of cluster partition when solving data clustering problems. The pseudocode of fitness calculation of ABC algorithm for solving cluster problems is shown in Algorithm 2, where each food source will be decoded to k clusters centers and the distances between objects and each center are calculated. Next, each of the objects will be assigned to the nearest cluster, and the total within-cluster variance will be calculated and taken as the food source's fitness [46].

5.3 Representative ABC-based clustering

Karaboga et al. [43] have applied the ABC algorithm for clustering analysis. Performance evaluation of the ABC algorithm shows that the ABC algorithm can efficiently be applied for data clustering. Yan et al. [46] have proposed a hybrid ABC (HABC) algorithm for data clustering by introducing the crossover operator of GA between the onlooker bee phase and scout bee phase of ABC:

Algorithm 2. Pseudo-code of fitness calculation of the ABC algorithm for clustering

```
Input: food sources X, data D
Output: fitness F
For each food source X_i
    Decode X_i to the k cluster centers following (29)
    Calculate the distance between all objects in D and each cluster center following (26)
     Assign objects to the nearest clusters centers
     Compute the total within-cluster variance V_i following (25)
     F_i=V_i
End for
Return F
```

$$child = rand(0,1) \times parent_1 + rand(0,1) \times parent_2 \tag{31}$$

where *a child* represents the newly produced offspring, while *parent1* and *parent2* are the two selected parents according to the binary tournament. Experiments indicate that the proposed HABC algorithm outperforms the original ABC and several other population-based clustering algorithms. Dang [48] et al. proposed an enhanced ABC and *K*-means (EABCK) to solve the clustering problem, where Eq. (5) of GABC instead of (3) ABC is used in employed bee phase and onlooker bee phase to improve the exploitation ability of ABC. Meanwhile, they proposed an improved information exchange mechanism as shown in

$$v_{i,j} = rand(0,1) \cdot (x_{i,j} - x_{k1,j}) + rand(0,1) \cdot (x_{best,j} - x_{k2,j}) \tag{32}$$

where $k1$ and $k2$ are two randomly selected individuals, and $x_{best,j}$ is the *j*-th dimension of the global best individual. We can see that the exploitation ability of EABCK is highly strong since the global best is used both in the employed bee phase and in the onlooker bee phase.

Algorithm 3. Pseudo-code of Two-step ABC clustering

1. Apply K-means algorithm to initialize *SN* food sources;
2. Employed bee phase: perform the same process as ABC, i.e., the search Equation (3) is used;
3. Like ABC, calculate the probability values according to (2);
4. Onlooker bee phase: perform the same process as ABC except that the (5) instead of (3) is used;
5. Scout bee phase: unlike ABC, the (33) is used to initialization the food source with maximum trial value.

Kumar et al. proposed an improved ABC (two-step ABC) to solve the clustering problem [49], and they also used Eq. (5) of GABC instead of Eq. (3) of ABC in the onlooker bee phase of two-step ABC to enhance the exploitation ability of ABC.

Nevertheless, to better balance the exploitation and exploration ability of two-step ABC, they still use Eq. (3) of ABC in the employed bee phase of two-step ABC. Another improvement in the two-step ABC is that the random initialization of the scout bee of ABC, i.e., (1) is modified as follows:

$$x_{new} = x_{best} + rand[0,1] \cdot (x_{best} - x_{curr}) \tag{33}$$

where x_{best} is the global best solution; x_{curr} is the position of the abandoned food; and $rand[0, 1]$ is a randomly generated number within [0, 1].

The procedure of two-step ABC clustering is shown in Algorithm 3. The EABCK and two-step ABC employ Eq. (5) of GABC instead of Eq. (3) to improve the exploitation ability of ABC. However, as mentioned above, it is pointed out that Eq. (5) of GABC used in EABCK and two-step ABC may cause oscillations [14, 15], so it may also reduce convergence, since the guidance of the last two terms may be in opposite directions. Therefore, the balance of EABCK and two-step ABC has not been well maintained and the performance of EABCK and two-step ABC can be improved.

Based on the above experiments, the proposed EABC_elite has shown to be very competitive with the optimization ability in complex test functions given its excellent balance ability between exploitation and exploration. It is anticipated that the EABC_elite achieves a better performance in the task of data clustering.

5.4 Proposed clustering algorithm

In the field of engineering, chaos theory is very useful in practical application. Chaos is a common nonlinear phenomenon, which is very complex and similar to randomness [50, 44]. Besides, it is susceptible to the initial value and can provide ergodicity, that is, the chaotic value has the opportunity to traverse all the domains within the specified range without repetition.

Recently, chaotic maps have been integrated with several meta-heuristic algorithms, such as the genetic algorithm [51] and cuckoo optimization [44]. In the field of ABC, Alatas [52] proposed a new ABC variant by combining the chaotic mapping into ABC (ChABC for short), but the chaotic maps are only used in the initialization phase and the scout bee phase, and most search behaviors of the bees have not been affected.

The clustering problem is a highly nonlinear complex problem with numerous local minima. In order to further enhance the global search ability of EABC_elite when solving the clustering problem, this paper incorporates chaotic mapping with ergodic, irregular, and stochastic properties in EABC_elite to further improve the global convergence. It is observed that the use of chaotic sequences in EABC_elite can further facilitate the escape from local minima, so sequences generated by the logistic map [53] replace the random parameter ϕ used in Eqs. (19) and (20) of EABC_elite. The parameter ϕ is replaced by the logistic sequence $\hat{c}$ shown in (35):

$$c_{t+1} = a \times c_t \times (1 - c_t),\ a = 4 \tag{34}$$

$$\hat{c}_{t+1} = 2 \times (c_{t+1} - 0.5) \tag{35}$$

Table 10 The summary of test datasets used in clustering experiments

Datasets	K	D	Number of data objects	Description
Iris	3	4	150 (50, 50, 50)	Fisher's iris data
Wine	3	13	178 (59, 71, 48)	Wine quality data
Glass	6	9	214 (70, 76, 17, 13, 9, 29)	Glass identification data
WBC	2	9	683 (444, 239)	Wisconsin breast cancer
CMC	3	9	1473 (629, 334, 510)	Contraceptive method choice

From Eq. (34), the chaotic value (c_{t+1}) at time $t+1$ depends only on the chaotic value at time t (c_t). Note that $c \in (0, 1)$ and $a=4$ were adopted in these experiments, as suggested in most research works. In Eq. (34), c_0 is generated randomly for each independent run, with $c_0 \neq \{0, 0.25, 0.5, 0.75\}$.

By using the new chaotic sequences shown in (35), Eqs. (19) and (20) can all be modified as follows:

$$v_{i,j} = \mu + \hat{c} \cdot \delta \tag{36}$$

where the meaning of $\hat{c}$ is the same as (35), $0 < \hat{c} < 1$, and μ and δ are the same as (19) and (20), respectively. Unlike [52], the chaotic sequence is used in the entire search process, so the global ability of EABC_elite is enhanced when solving the clustering problem. Hybridization of the algorithm is one of the active research areas used to enhance the performance of algorithms. In wto-step ABC, a multi-step K-means algorithm is embedded into the ABC algorithm to enhance the performance of the ABC algorithm in clustering. EABCK also employs K-means to enhance its performance. Thus, for fair comparison purposes, the proposed clustering also employs the K-means algorithm to initialize the food source.

By combining the chaotic parameter generated and K-means initialization strategy with EABC_elite, a novel two-step clustering algorithm, namely TEABC_elite, is proposed, as depicted in Algorithm 4.

Algorithm 4: The pseudo-code of the TEABC_elite for data clustering

Input: the original data used for clustering

```
For i=1 to SN   //i represent the ith employed bee and SN represents the total number of food sources
□   Initialize k cluster centers {C1,C2,...Ck} randomly from the data points within the boundary of a given dataset
      Apply the K-means to iterate the k clusters until the k centers no longer changed or termination criteria
        satisfied
      The ith food source is coded as xi={C1,C2,...Ck};//Initialization Phase
End
 Fes=0
while Fes<max_FEs
    Select the top T=p.SN solutions as elite solutions from population
    for i=1 to SN        //employed bee phase
      Generate a new candidate solution vi using equation (36), evaluate the candidate solution vi
     Adjust boundary according to equation (30);
     if f(vi)< f(xi),
        Replace xi by vi and counter(i)=0
     else
        counter(i)= counter(i)+1
      end if
    end for           //end employed bee phase
    for i=1 to SN     //onlooker bee phase
        Select a solution xe from elite solutions randomly to search
        Generate a new candidate solution ve using equation (36), evaluate the new candidate solution ve
      Adjust boundary according to equation (30);
        if f(ve)< f(xe)
          Replace xe by ve and counter(e)=0
        else
          counter(e)= counter(e)+1
        end if
    end for        //end onlooker bee phase
    Fes=Fes+SN×2
    Select the solution xmax with max counter value     //Scout bee phase
    if counter(max)>limit   //only one food source with max counter value can be initialized
      Replace xmax by a new solution generated according to (1), Fes=Fes+1,counter(max)=0;
      Adjust boundary of xmax according to equation (30);
    end if                 //end scout bee phase
end while
decode the xbest to k clusters center {C1,C2,...Ck} according to (29), get partitioned data according to Algorithm 2.
```

Output: ***k*** **clusters center** $\{C_1,C_2,\ldots C_k\}$; **the partitioned data**

5.5 Experiments of TEABC_elite for data clustering

To investigate the performance of TEABC_elite algorithm for data clustering, we make a comparison between TEABC_elite and two-step ABC [49], EABCK [48], HABC [46], ARABC [54], ECABC [23], ABCLGII [24], SLPSO [28], sinDE [35], and *K*-means [49] on five well-known datasets. These datasets are the benchmark datasets in the clustering field and widely used to analyze the performance of the newly developed algorithms, and they are iris, wine, CMC, glass, and WBC, available for download from the UCI repository.[2] They are listed briefly as Table 10, where the number of clusters of each cluster is denoted by k, and d specifies the number of attributes of each dataset. For the sake of fairness, the maximum number of fitness function evaluations (*max_FEs*) is set to 10,000 as recommended and presented in [46].

The values of the common control parameters in all algorithms are set as follows. For all ABC and variants, the population size is set to 100 [46], and *limit* set to 100 as well. Moreover, the number of employed bees and onlooker bees were set to be half of the total population, SN = employed bees = onlooker bees = 50. For PSO and DE variants such as sinDE and SLPSO, the population size is set to 50. Other algorithmic parameters of all algorithms being compared are as follows, set according to the original literature: two-step ABC, $limit = 10, \varphi = 1.5$; for EABK, $limit = 100$; for HABC, $limit = 100$; for ARABC, $limit = SN \cdot D$, $\Delta = 0.01$, $\alpha_{\min} = 0$, $\alpha_{\max} = 5$; for ABCLGII, $r = 1$, $q = 0.2$; and for sinDE, $freq = 0.25$.

Note that the *K*-means algorithm needs the initial cluster centers only, and no additional parameters are needed. In SLPSO, all parameters are set adaptively according to population size and dimension D. In [49], the population size of two-step ABC was set to 20, but we use 100 here instead of it for all algorithms to make a fair comparison. The outcome of the proposed method is described regarding average within-cluster distances and standard deviation.

Experimental results are given in Table 11 on the iris, wine, CMC, WBC, and glass datasets, where "mean" denotes the average total within-cluster variance for 30 executions and "SD" denotes the standard deviation. The symbol "Rank" denotes the performance order of all compared algorithms according to the total within-cluster variance criterion on five data sets.

On iris dataset, the performance order of the algorithms is TEABC_elite = Two-step ABC > ECABC = ABCLGII > EABCK = ABC_elite > sinDE > HABC >> DG ABC > ARABC > SLPSO > *K*-means. Results obtained by TEABC_elite and two-step ABC are close with each other since they employ the global best individual to guide the search process, meanwhile adopt mechanisms to avoid premature convergence. Specifically, two-step ABC only uses the global best solution in the onlooker bee phase, while the TEABC_elite uses the ordinary solution to balance the great lead ability of the global best solution. By comparison, the EABCK employs the global best solution to guide the search process, both in the employed bee phase and in the onlooker bee phase, so the algorithm is easy to get trapped in the local minimum. Therefore, EABCK achieves the biggest deviation except for *K*-means.

[2] http://archive.ics.uci.edu/ml.

Table 11 Average total within-cluster variance of 12 algorithms (Ave. Rank denotes average ranking)

Data sets	Mean (SD)	TEAB C_elite	ABC_ elite	Two-step ABC	DGA BC	EAB CK	HABC	ARA BC	ECA BC	ABCL GII	SLP SO	SinDE	K-m eans
Iris	Mean	**96.65**	**96.69**	**96.65**	96.80	96.69	96.71	96.83	96.66	96.66	97.20	96.70	104.32
	(SD)	0.0031	0.0548	0.0042	0.76	3.78	0.64	0.26	0.12	0.013	0.54	0.16	5.34
	Rank	**1**	3	1	6	3	5	7	2	2	8	4	9
Wine	Mean	**16293.2**	16308.1	16294.3	16303.1	16298.3	16301.9	16299.5	16294.4	16297.3	16306.1	16296.5	16555.0
	(SD)	3.27	5.78	4.18	4.37	6.19	3.18	5.22	4.27	9.34	21.84	10.78	473.82
	Rank	**1**	11	2	9	6	8	7	3	5	10	4	12
CMC	Mean	**5534.2**	5538.1	5633.3	5550.9	5672.4	5698.3	5567.7	5536.1	5553.3	5701.8	5569.3	5742.1
	(SD)	1.81	3.23	2.29	4.17	3.24	2.98	5.12	2.23	4.83	5.21	4.78	23.18
	Rank	**1**	3	8	4	9	10	6	2	5	11	7	12
WBC	Mean	2965.41	**3021.39**	**2964.82**	3025.18	2969.62	2973.72	2966.72	2988.32	2975.89	2987.72	2980.28	3217.72
	(SD)	9.76	0.07	11.87	12.53	23.25	22.29	12.82	11.76	8.98	38.46	29.82	126.87
	Rank	2	10	**1**	11	4	5	3	9	6	8	7	12
Glass	Mean	**210.75**	246.72	211.53	248.30	242.43	237.56	247.12	218.83	236.6012	244.78	264.28	241.79
	(SD)	3.14	5.64	3.20	5.37	4.78	7.23	11.85	4.49	4.81	4.42	8.82	10.03
	Rank	**1**	9	2	11	7	5	10	3	4	8	12	6
Ave. Rank		**1.2**	7.2	2.8	8.2	5.8	6.6	6.6	3.8	4.4	9.0	6.8	10.2
Total Rank		**1**	9	2	10	5	6	6	3	4	11	8	12

Bold indicates the best results

In other datasets, similar rank results are obtained. In Table 11, a rank function is used to determine the performance of all algorithms with corresponding datasets, and finally, an average rank is obtained using the individual rank of algorithms. To sum up, the proposed algorithm TEABC_elite obtains the best average rank among the compared ones of 1.2, while SLPSO and *K*-means obtain the worst two ranks, 9.0 and 10.2, respectively.

As can be seen from Table 11, the proposed algorithm TEABC_elite achieves the best clustering results on four datasets and ranks second on one dataset. The main reason is that the proposed TEABC_elite effectively utilizes ordinary solutions and has a better global search ability, so it avoids falling into the local optimal solution, achieving more stable performance.

6 Conclusions

In order to accelerate convergence and seeking for a better exploration–exploitation balance, an improved elite-guided ABC variant EABC_elite is proposed by using two novel search equations. The global best solution is used in the first equation on the employed bee phase to accelerate the convergence process, while the ordinary solution is used on the employed bee phase and onlooker bee phase to avert precocity. Comparing existing elite-guided ABC variants, such as ABC_elite, IABC_elite, and ABCLGII, each individual is guided by the global best individual to accelerate convergence in EABC_elite and ECABC, while EABC_elite uniquely has no bias to any search directions and show better global search ability by using novel balance strategy. Experiments on well-known test suites demonstrate that the proposed algorithm is significantly better than other ABC variants also some non-ABC variants on most of the functions tested regarding solution quality, robustness, and convergence speed. Additionally, the proposed EABC_elite can also be applied to solve the feature selection problems, where experimental results show that the performance of EABC_elite is superior to other feature selection methods.

Furthermore, TEABC_elite is designed to enhance the global search ability to solve data clustering, where the chaos parameter and *K*-means initialization strategies are integrated into EABC_elite. Experimental results executed on well-known datasets show that TEABC_elite has superior performance than other existing clustering methods, confirming that it is a competitive clustering tool.

Acknowledgements Authors of this manuscript are grateful to the valuable comments provided by external reviewers and international experts for the improvement in technical and organization sections.

Funding This research was supported in part by the National Natural Science Foundation of China (Nos. 61672338 and 61673160), in part by the Chaozhou Science and Technology Project (No. 2018GY45).

References

1. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
2. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
3. Zhan Z, Zhang J, Li Y (2009) Adaptive particle swarm optimization. IEEE Trans Cybern 39(6):1362–1381
4. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell M 1(4):28–39
5. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
6. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
7. Du Z, Han D, Liu G, Bi K (2017) An improved artificial bee colony algorithm with elite-guided search equations. Comput Sci Inf Syst 14(3):751–767
8. Xiang W, Meng X, Li Y (2018) An improved artificial bee colony algorithm based on the gravity model. Inf Sci 429:49–71
9. Pan Q, Wang L, Li J (2014) A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimization. Omega 45:42–56
10. Bose D, Biswas S, Vasilakos AV (2014) Optimal filter design using an improved artificial bee colony algorithm. Inf Sci 281:443–461
11. Szeto W, Wu Y, Ho SC (2011) An artificial bee colony algorithm for the capacitated vehicle routing problem. Eur J Oper Res 215(1):126–135
12. Gao W, Chan F, Huang L (2015) Bare bones artificial bee colony algorithm with parameter adaptation and fitness-based neighborhood. Inf Sci 316:180–200
13. Zhu G, Kwong S (2010) Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math Comput 217(7):3166–3173
14. Gao W, Liu S, Huang L (2013) A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans. Cybern. 43(3):1011–1024
15. Cui L, Li G, Lin Q (2016) A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. Inf Sci 367(22):1012–1044
16. Kiran MS, Hakli H, Gunduz M (2015) Artificial bee colony algorithm with variable search strategy for continuous optimization. Inf Sci 300(8):140–157
17. Banharnsakun A, Achalakul T, Sirinaovakul B (2011) The best-so-far selection in artificial bee colony algorithm. Appl Soft Comput 11(2):2888–2901
18. Gao W, Liu S (2012) A modified artificial bee colony algorithm. Comput Oper Res 39(3):687–697
19. Karaboga D, Gorkemli B (2014) A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl Soft Comput 23(10):227–238
20. Gao W, Liu S, Huang L (2014) Enhancing artificial bee colony algorithm using more information-based search equations. Inf Sci 270(12):112–133
21. Gao W, Liu S (2011) Improved artificial bee colony algorithm for global optimization. Inf Sci 111(17):871–882
22. Cui L, Zhang K, Li G (2017) Modified Gbest-guided artificial bee colony algorithm with new probability model. Soft Comput 22(7):1–27
23. Kong D, Chang T, Dai W (2018) An improved artificial bee colony algorithm based on elite group guidance and combined breadth-depth search strategy. Inf Sci 442:54–71
24. Lin Q, Zhu M, Li G (2018) A novel artificial bee colony algorithm with local and global information interaction. Appl Soft Comput 62:702–705
25. Liang J, Qu B, Suganthan PN (2014) Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China, Tech. Rep. 201411A
26. Xiang W, Li Y, Mengl X (2017) A grey artificial bee colony algorithm. App Soft Comput 60(11):1–17
27. Li Y, Zhan Z (2015) Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. Inf Sci 293(4):370–382
28. Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291(1):43–60
29. Li C, Yang S, Nguyen T (2012) A self-learning particle swarm optimizer for global optimization problems. IEEE Trans Man Cybern 42(33):627–646

30. Kiran MS, Hakli H, Gunduz M (2015) Artificial bee colony algorithm with variable search strategy for continuous optimization. Inf Sci 300(8):140–157
31. Gao W, Huang L, Wang J (2016) Enhanced artificial bee colony algorithm through differential evolution. Appl Soft Comput 48(11):137–150
32. Derrac J, Garcia S, Molina D (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18
33. Tanweer MR, Suresh S, Sundararajan N (2015) Self regulating particle swarm optimization algorithm. Inf Sci 294(4):182–202
34. Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–948
35. Draa A, Bouzoubia S, Boukhalfa I (2015) A sinusoidal differential evolution algorithm for numerical optimization. Appl Soft Comput 27:99–126
36. Liang Z, Hu K, Zhu Q (2017) An enhanced artificial bee colony algorithm with adaptive differential operators. Appl Soft Comput 58(9):480–494
37. Chuang LY, Yang CH, Li JC (2014) Chaotic maps based on binary particle swarm optimization for feature selection. App Soft Comput 11(1):239–248
38. Ghamary M, Mobasheri MR, Mojaradi B (2014) Unsupervised feature selection using geometrical measures in prototype space for hyperspectral imagery. IEEE Trans Geosci Remote 52(7):3774–3787
39. Shi Y, Pun CM, Hu H (2016) An improved artificial bee colony and its application. Knowl Based Syst 107:14–31
40. Guo C, Zhou Y, Ping Y (2014) A distance sum-based hybrid method for intrusion detection. Appl Intell 40(1):178–188
41. Liu R, Chen Y, Jiao L (2014) A particle swarm optimization based simultaneous learning framework for clustering and classification. Pattern Recogn 47(6):2143–2152
42. Farnad B, Jafarian A, Baleanu D (2018) A new hybrid algorithm for continuous optimization problem. Appl Math Model 55(3):652–673
43. Karaboga D, Ozturk C (2011) A novel clustering approach: Artificial Bee Colony (ABC) algorithm. Appl Soft Comput 11(1):652–657
44. Boushaki SI, Kamel N, Bendjeghaba O (2017) A new quantum chaotic cuckoo search algorithm for data clustering. Expert Syst Appl 96(4):358–372
45. Xiang W, Zhu N, Ma S (2015) A dynamic shuffled differential evolution algorithm for data clustering. Neurocomputing 158(6):144–154
46. Yan X, Zhu Y, Zou W (2012) A new approach for data clustering using hybrid artificial bee colony algorithm. Neurocomputing 97(1):241–250
47. Gungor Z, Unler A (2007) K-harmonic means data clustering with simulated annealing heuristic. Appl Math Comput 184(2):199–209
48. Dang CT, Wu Z, Wang Z (2015) A novel hybrid data clustering algorithm based on artificial bee colony algorithm and k-means. Chin J Electron 24(4):694–702
49. Kumar Y, Sahoo G (2017) A Two-step artificial bee colony algorithm for clustering. Neural Comput Appl 28(3):537–551
50. Lu H, Zhang H, Ma H (2006) A new optimization algorithm based on chaos. J Zhejiang Univ-Sci A 7(4):539–542
51. Ebrahimzadeh R, Jampour M (2013) Chaotic genetic algorithm based on lorenz chaotic system for optimization problems. Int J Intell Syst 5(5):19–24
52. Alatas B (2010) Chaotic bee colony algorithms for global numerical optimization. Expert Syst Appl 37(8):5682–5687
53. May RM (1976) Simple mathematical models with very complicated dynamics. Nature 261(5560):459–467
54. Cui L, Li G, Wang X (2017) A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. Inf Sci 417:169–185

Affiliations

Zhenxin Du[1,2] · Dezhi Han[2] · Kuan-Ching Li[3]

Zhenxin Du
duzhenxinmail@163.com

Dezhi Han
dzhan@shmtu.edu.cn

1 School of Computer Information Engineering, Hanshan Normal University, Chaozhou 521041, China

2 College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

3 Department of Computer Science and Information Engineering, Providence University, Taichung 43301, Taiwan