Soft Computing
https://doi.org/10.1007/s00500-022-06826-1

APPLICATION OF SOFT COMPUTING

A self-adaptive level-based learning artificial bee colony algorithm for feature selection on high-dimensional classification

Jing Wang[1] · Yuanzi Zhang[1] · Minglin Hong[1] · Haiyang He[1] · Shiguo Huang[1]

Accepted: 25 January 2022

Abstract
Feature selection is an important data preprocessing method in data mining and machine learning, yet it faces the challenge of "curse of dimensionality" when dealing with high-dimensional data. In this paper, a self-adaptive level-based learning artificial bee colony (SLLABC) algorithm is proposed for high-dimensional feature selection problem. The SLLABC algorithm includes three new mechanisms: (1) A novel level-based learning mechanism is introduced to accelerate the convergence of the basic artificial bee colony algorithm, which divides the population into several levels and the individuals on each level learn from the individuals on higher levels, especially, the individuals on the highest level learn from each other. (2) A self-adaptive method is proposed to keep the balance between exploration and exploitation abilities, which takes the diversity of population into account to determine the number of levels. The lower the diversity is, the fewer the levels are divided. (3) A new update mechanism is proposed to reduce the number of selected features. In this mechanism, if the error rate of an offspring is higher than or is equal to that of its parent but selects more features, then the offspring is discarded and the parent is retained, otherwise, the offspring replaces its parent. Further, we discuss and analyze the contribution of these novelties to the diversity of population and the performance of classification. Finally, the results, compared with 8 state-of-the-art algorithms on 12 high-dimensional datasets, confirm the competitive performance of the proposed SLLABC on both classification accuracy and the size of the feature subset.

Keywords Artificial bee colony algorithm · Feature selection · Classification · High dimensionality

1 Introduction

Feature selection (FS) is an important data preprocessing method in data mining. With the rapid development of data acquisition technology, the number of features collected in many applications increases greatly. However, not all of them are related to the given target. Irrelevant and redundant features may even reduce the performance of classification. This curse of dimensionality is a major obstacle in machine learning and data mining (Gheyas and Smith 2010). Hence, FS aims to choose a small number of relevant and non-redundant features to achieve similar or even better classification performance than using all features. FS methods can be categorized into three types: wrapper, filter and embedded. Filter methods analyze intrinsic properties of features and rank their importance, ignoring objective function (Rakshit et al. 2013). Wrapper methods utilize objective function to evaluate the quality of different feature subsets and select the best one (Grande et al. 2007). Compared with filter methods, wrapper methods usually obtain higher accuracy. However, they have high computational cost, especially for high-dimensional, large datasets. Embedded methods attempt to reduce the computation cost of reclassifying different subsets that are performed in wrapper methods. They perform feature selection in the process of training. Therefore, the performance of an embedded method is usually specific to the given learning algorithms, which results in poor robustness with respect to the change of learning algorithms (Rao et al. 2019). In this study, we focus on the wrapper method.

In principle, FS is an NP-hard combination problem. Given a feature set with n features, the number of all possible feature subsets is 2^n, which makes it too costly and

✉ Shiguo Huang
fjhsg25@126.com

[1] College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China

restrictive practically to specify the best feature subset with traditional exhaustive search approaches since the size of the search space increases exponentially as the number of features increases. Therefore, evolutionary computation (EC) techniques attract attention due to their global search strategies and have been proven to be effective in dealing with the FS problems (Oh et al. 2004). There are many typical EC techniques, such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO) and differential evolution (DE). All these EC techniques show the effectiveness in dealing with feature selection. Details will be described in the next section.

In this paper, we concentrated on a recent EC technique, i.e., artificial bee colony (ABC) Algorithm, which was proposed by Karaboga in 2005 (Karaboga 2005) and simulates the intelligent foraging behaviors of a honey bee swarm. Compared with other evolutionary algorithms, ABC algorithm has many advantages, such as strong robustness, high flexibility, simple structure, few control parameters and strong ability to explore a wide space. Accordingly, ABC is chosen for feature selection in this paper. However, when dealing with FS problem, the basic ABC algorithm has some drawbacks, including a slow convergence, insufficient exploitation and the best solution in the bee population is memorized but is not used to guide the population update.

To address the above issues, many studies have been developed, however, few of them are developed for high-dimensional FS, and most of them aim to improve the classification performance by combining ABC algorithm with other algorithms, which have more complex parameters to control than a single algorithm (reviewed at length in Sect. 2). Therefore, we propose a self-adaptive level-based learning artificial bee colony algorithm (SLLABC) for feature selection given as follows:

(1) Introducing a novel level-based learning mechanism into ABC algorithm. The bees are divided into high to low levels according to their fitness. Each bee "emulates those better than itself" from higher levels, except for the bees in the highest-level learning from each other. This novel mechanism effectively improves the exploitation abilities of the ABC algorithm, and accelerates the convergence of the ABC algorithm.

(2) Proposing a self-adaptive method for determining the number of levels. The number of levels is dynamically and automatically adjusted by diversity of individuals during evolution. The higher the diversity of individuals, the more the levels and the lower the probability of dominant individuals being learned, and vice versa. This method preserves high exploration in early evolution and promotes high exploitation in later evolution.

(3) Proposing a new update strategy for optimal individuals. Different from some single-objective methods that only consider the accuracy, as well as the multi-objective methods that adopt multi-objective fitness functions, this strategy takes both error rate and subset size into consideration but gives priority to the lower error rate, which impels the population to approach the optimal subset with both the minimum error rate and the smallest size.

The rest of this paper is organized as follows. Section 2 reviews the relevant literature on EC-based feature selection algorithms. Section 3 introduces the basic artificial bee colony algorithm. In Sect. 4, the SLLABC algorithm is proposed based on three new strategies. The experimental results are shown in Sect. 5. Finally, conclusions are given in Sect. 6.

2 Related works

To eliminate the negative impact of the irrelevant and redundant features, a variety of feature selection methods have been proposed. However, due to the inefficiency of traditional search approaches in feature selection which is a complex combinatorial optimization problem, various EC-based feature selection algorithms have been proposed. The forerunner is Siedlecki and Sklansky who prove that genetic algorithm (GA) is a powerful tool for feature selection when the dimensionality of the given feature set is greater than 20 (Siedlecki and Sklansky 1993). After that, to further improve the performance of GA for feature selection, many different improvements have been proposed in search mechanisms (Demirekler and Haydar 1999; Jeong et al. 2015; Wang et al. 2020) and fitness function (Canuto and Nascimento 2012; Sousa et al. 2013). A feature selection method (Derrac et al. 2009) based on GA utilizes the cooperative coevolution (CC) framework (Potter and Jong 2000) but is not investigated in the large datasets. With the same idea of "divide and conquer", particle swarm optimization (PSO) achieves better performance than GA algorithm (Song et al. 2020; Van den Bergh and Engelbrecht 2004). However, the traditional personal best and global best updating mechanism in PSO limits its performance for feature selection and the potential of PSO for feature selection has not been fully investigated. Therefore, Xue et al. have proposed new initialization strategies and new personal best and global best updating mechanisms (Xue et al. 2014b). As the dimension of the problem increases, most PSO-based FS methods consume a significant amount of memory and require a high computational cost. A variable-length PSO (Tran et al. 2018) has been developed to deal with high-dimensional data. Compared with fixed-length PSO methods, the proposed variable-length PSO can achieve much smaller feature subsets with significantly higher classification performance in much shorter time. A PSO variant namely competitive swarm optimizer (CSO) directly adopts predominant particles in the current swarm to guide the update of particles (Cheng and Jin 2015), the diversity of this optimizer is greatly promoted and thus shows good performance in dealing with large scale optimization. In Ref. (Gu et al. 2018), the CSO is applied to high dimensional FS problem. Recently, a hybrid CSO (Ding et al. 2020) algorithm has been proposed to improve the drawbacks of the original CSO with low computational efficiency and avoid falling into local optimum. The CC framework has been applied to differential evolution (DE) as well, namely DECC (Shi et al. 2005). DE is also a very popular evolutionary algorithm used in FS problems. Xue et al. designed a DE-based multi-objective feature selection algorithm (Xue et al. 2014a). After that, a binary DE variant with self-learning (MOFS-BDE) is employed to improve the classification performance of DE (Zhang et al. 2020). Since FS is a discrete combinatorial problem, it has been widely achieved by ant colony optimization (ACO) which was designed to solve discrete optimization problems originally. Interestingly, the CC framework has also been applied to ACO algorithm (Vieira et al. 2010). However, ACO usually represents FS as a graph problem that restricts the scalability of the algorithm and the interaction between features. Hence, some hybrid algorithms of ACO and other EC algorithms have been proposed to improve its scalability, such as a hybrid algorithm of ACO and GAs (Hamamoto et al. 2015), three mechanisms for hybrid ACO-PSO based approaches for feature selection (Menghour and Souici-Meslati 2016). Recently, Meenachi and Ramakrishnan have hybridized ACO and fuzzy rough set to select global optimal features (Meenachi and Ramakrishnan 2020). Except for the above algorithms, many other EC-based feature selection algorithms have also been developed. Since too many studies exist, we cannot review them all. Here, to save space, we only list the above typical works. For a comprehensive review of EC-based algorithms, readers can refer to Ref. (Xue et al. 2016) and Ref. (Nguyen et al. 2020).

This paper concentrated on the ABC-based FS algorithm. More and more papers have developed ABC variants for FS in the past decade. Gao et al. claimed that the ABC is good at exploration but poor at exploitation. In order to overcome this issue, opposition-based learning method and chaotic maps are used in the initialization of the algorithm, and the update rule of basic algorithm is replaced with a new update rule by considering the best solution in the population (Gao et al. 2011). A discrete binary ABC variant (DisABC) substantially modifies the search mechanism by using the Jaccard similarity coefficient for feature selection problems (Hancer et al. 2015). An ABC variant namely CLABC (Cooperative learning ABC) utilizes a cooperative learning strategy with modified search mechanisms and multiple search equation (Harfouchi and Habbi 2015). In another study, the population of ABC is divided into two groups, called diversity population-DP and convergence population-CP. In CP, the promising food sources are stored to improve convergence ability of the algorithm, and historical unpromising food sources are stored in DP to maintain diversity of the population (Cui et al. 2018). In order to improve the local search capability of the basic algorithm, a quick ABC algorithm (Karaboga and Gorkemli 2014) and some update strategies based on normal distribution have been proposed in Ref. (Babaoglu 2015). To overcome slow convergence issue and tendency to local optimum of basic ABC, opposition-based learning and generalized opposition-based learning have been integrated with the basic ABC (Zhou et al. 2015).

Although many ABC variants have been proposed, few of them are developed for high-dimensional FS. In other words, the most existing ABC variants are at the cost of higher computational complexity and more complex algorithm implementation.

Comparing the ABC algorithm with other swarm intelligence algorithms, one of its advantages is that exploration and development are clearly separated. Particularly, the employed bees and the onlooker bees that focus on neighboring solutions perform exploitation, while the scout bees that focus on renewing random solutions perform exploration. However, many studies combine ABC with other optimization algorithms to keep the balance between exploration and exploitation. ABC has been combined with DE to perform FS for classification in ABC-DE (Zorarpacı and Özel 2016). This algorithm contains a new binary neighborhood search mechanism and a modified onlooker bee process for the ABC algorithm; it also has a new binary mutation phase for the DE algorithm. In another study, a hybrid study based on ABC and quantum evolutionary algorithm has been proposed and ABC is used for improving the local search capability of the hybrid algorithm (Duan et al. 2010). By combing the characteristics of ACO and ABC algorithms, a novel hybrid algorithm AC-ABC has been proposed to optimize feature selection (Shunmugapriya and Kanmani 2017). In this variant, ants are committed to exploiting the optimal feature subset, and the bees use the feature subset generated by ants as their food source.

Although the above hybrid algorithms achieve promising results, they also have more parameters than a single algorithm, including the parameters from both basic algorithms and the parameters to control the hybridization. Given the clear separation between exploration and exploitation, a novel search mechanism was introduced into ABC algorithm in this study to control them instead of hybridizing the ABC algorithm with other algorithms.

3 Artificial bee colony algorithm

In the basic ABC algorithm, the colony of artificial bees is divided into three types: employed bees, onlooker bees and scout bees. The employed bees search for food sources and share their information with onlooker bees. The onlooker bees select one of the food sources according to the information from employed bees, and exploit the neighborhood space of the food sources to produce a new food source. An employed bee whose food source has been improved through a predetermined number of trials becomes a scout bee and starts to randomly search for a new food source. The basic implementation of ABC comprises four phases:

(1) Initialization phase: The algorithm randomly produces food sources. Each food source is described as a vector in the search space: $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$, and is generated by Eq. (1):

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

where $i = \{1, 2, \ldots, \mathrm{SN}\}$ and SN is the number of the food source and is equal to the number of employed bees or onlooker bees. $j = \{1, 2, \ldots, D\}$ and D is the dimensionality of the search space. $x_{i,j}$ is the j th dimension of x_i. $U(0,1)$ is a random variable which distributed uniformly, $x_j^{\min}$ and $x_j^{\max}$ are the maximum and minimum boundary value, respectively.

(2) Employed bee phase: Each employed bee is associated with a food source. Employed bees need to modify the position of their food source to find new better ones. Thereby, they learn from a neighbor source x_k which is selected randomly among all sources except for itself. The new food source is produced by Eq. (2):

$$x'_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,j}\right) \tag{2}$$

where x_i is the current food source, $\phi_{i,j}$ is a uniformly distributed random value within [− 1, 1]. After $x_i\prime$ is produced, its fitness value is evaluated and compared with x_i. If $x_i\prime$ is better than x_i, $x_i\prime$ replaces x_i to enter next iteration and its counter holding the number of trials is reset to 0. Otherwise, the current source x_i is kept into next iteration and its counter holding the number of trials is increased by 1.

(3) Onlooker bee phase: After each iteration, all employed bees give information about their sources to onlooker bees. Each onlooker bee selects a food source according to the fitness values by roulette-wheel scheme, where the better the fitness value of the source, the higher the probability of being selected. The probability value is calculated by Eq. (3):

$$p_{\mathrm{i}} = \frac{\mathrm{fit_i}}{\sum_{n=1}^{SN} \mathrm{fit_n}} \tag{3}$$

where $\mathrm{fit_i}$ is the fitness value of source x_i, which is calculated by Eq. (10). After calculating the probability value of each source, a random number $\mathrm{rand}(0,1)$ is generated to determine which source to choose. If $p_{\mathrm{i}} > \mathrm{rand}(0,1)$,$x_{\mathrm{i}}$ is chosen to update just as in the employed bee phase.

(4) Scout bee phase: Each source has a counter which is zero at the beginning. If the counter holding the number of trials exceeds the predefined threshold value, its corresponding food source will be abandoned and replaced by a new food source, which is generated by Eq. (1).

When the ABC algorithm is applied to feature selection, we abstract each food source into a feature subset, and the quality of the food source is the fitness value of the feature subset. Each individual is represented with a binary string. "1" in the string means the feature is selected and "0" means the feature is not selected.

4 The proposed method

4.1 A novel level-based learning strategy

Yang et al. (Yang et al. 2017) proposed a level-based learning (LL) mechanism for PSO, namely level-based learning swarm optimizer (LLSO), and apply it to continuous space optimization problems. In LLSO, two particles are selected from two different levels to instruct the current particle to update the position, and the particles in the highest level directly enter the next iteration without update. Due to the difference between ABC and PSO, a novel LL mechanism is proposed for ABC-based feature selection to solve the problem in discrete space.

The novel LL mechanism is described as follows:

(1) The individuals in the population are sorted in ascending order of the fitness values firstly.

(2) The sorted population is divided into NL levels, the individuals with good fitness values are assigned to

high levels, and the individuals with poor fitness values are assigned to low levels. The number of individuals in each level is the same. Note that the whole swarm may not be equally partitioned by NP/NL. In this situation, we assign the last NP%NL individuals into the lowest level.

(3) Each individual "emulates those better than itself" by learning from a random individual in higher levels, and the individuals in the highest level learn from a random individual in the same level to update their positions.

To have a better understanding of the novel LL mechanism, we take an example in Fig. 1, in which the population is divided into four levels. Level1 is the highest level, and Level4 is the lowest level. According to the above process, each individual in Level4 needs to learn from an individual randomly selected in Level3 or Level2 or Level1, each individual in Level3 needs to learn from one individual randomly selected in Level2 or Level1, each individual in Level2 needs to learn from one individual randomly selected in Level1, and each individual in Level1 learns from another individual randomly selected in the same level. In short, x_k in Eq. (2) is selected randomly from many individuals in the selected level which is one of the higher levels.

Obviously, the number of candidate exemplars for individuals in different levels is different. In other words, as the level that an individual belongs to goes higher, this individual has fewer exemplars in the higher levels in total to learn from. This level-based learning strategy encourages more exploration among individuals in lower levels and more exploitation among those in higher levels.

4.2 A self-adaptive method for determining the number of levels

The number of levels NL has an important impact on the performance of feature selection. Yang et al. (Yang et al. 2017) believe that different data sets have different numbers of levels, and then proposed a dynamic version of LLSO (DLLSO) to determine the number of levels. The description of DLLSO is as follows: Firstly, to input a set containing different integers that represent the candidate numbers of levels. Then, at each generation, LLSO selects an integer from the set based on their probabilities, and the performance of LLSO with this level number is recorded at the end of the generation. The greater the improvement in the fitness value of the current optimal solution is, the more likely it is to choose this integer as NL in the next iteration.

The method of determining the number of levels by DLLSO needs to give a predefined set of candidate integers according to the given problem, which is hard to be given successfully. Therefore, a self-adaptive method to determine *NL* is proposed in this study. The basic idea of this method is to use the diversity of the population, i.e., the similarity of individuals within the population, to obtain the series NL. The details are as follows:

(1) The distance between individuals is used to measure the similarity between individuals. In feature selection, the value of each dimension of individuals is 0 or 1, so Hamming distance is used to measure the similarity between individuals, which is defined as follows:

$$H_{i,j} = \sum_{d=1}^{D} \left| x_{i,d} - x_{j,d} \right| \tag{4}$$

where $H_{i,j}$ is the Hamming distance between x_i and x_j, x_i and x_j are two different individuals in the population, D is the dimensionality of the search space.

Then the average diversity Ave_t of the population in the t th iteration is calculated as Eq. (5).

$$\mathrm{Ave}_t = \mathrm{mean}\left(\sum_{i=1}^{i=NP-1} \sum_{j=i+1}^{NP} H_{i,j} \right) \tag{5}$$

where mean() denotes the mean function, and NP is the population size.

(2) Due to mutual learning, the individuals in the population tend to be similar during evolution, which leads to a decrease in the average diversity of the population. To avoid a sharp decline in the exploration performance in the early iteration, we divide the population into more levels to enhance the diversity of level selection and reduce the probability of individuals in dominant levels to be learned owing to the small number of individuals in each level, which can avoid a sharp decline in population diversity, while in later iterations, the diversity in individual selection becomes more important. Increasing the probability of dominant individuals being learned is beneficial for the colony to exploit the search space more intensively. Therefore, the decline rate of the average diversity is used to determine the NL in the current iteration. The decline

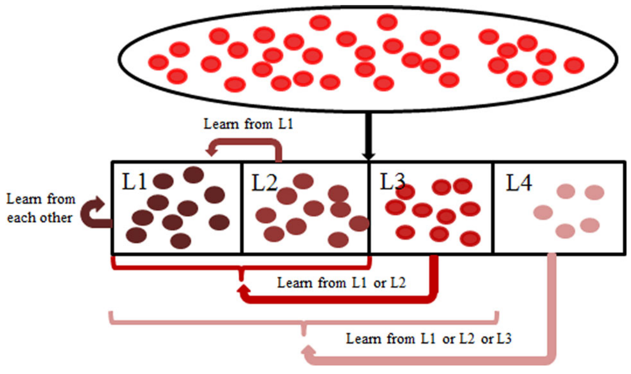


Fig. 1 The framework of the novel level-based learning strategy

rate of the average diversity can be calculated from the following formula:

$$\mathrm{NL} = \begin{cases} 2, & \mathrm{temp} < 2 \\ \mathrm{fix}(\mathrm{temp}), & 2 \le \mathrm{temp} \le NP \\ \mathrm{NP}, & \mathrm{temp} > NP \end{cases} \tag{6}$$

where the lower and upper bounds of NL are 2 and NP, respectively. fix(temp) returns the maximum integer which is not more than temp. temp is calculated as follows:

$$\mathrm{temp} = \frac{\mathrm{Ave_t}}{\mathrm{Ave_0}} \times \mathrm{NP} \tag{7}$$

where $\mathrm{Ave_t}$ is the average diversity of the population at t th iteration, and $\mathrm{Ave_0}$ is the average diversity after initializing.

Compared with DLLSO that gives a predefined set, our method of determining NL by using the average diversity of the population is simple and easy to implement. It not only saves a lot of parameter trial work but also has good performance in balancing exploration and exploitation. The experiments in Sect. 4 will show the comparative results of the two methods.

4.3 A new mechanism for updating the individuals

In ABC algorithm, the update mechanism is very important to the performance of the algorithm. If the food source in the employed bee or onlooker bee phase is not updated for a long time, it will be abandoned and reinitialized to produce a new food source, which reduces the convergence speed of the algorithm. Meanwhile, FS mainly contains two objectives, i.e., minimizing the classification error rate and minimizing the number of selected features, simultaneously. However, in most feature selection algorithms, the fact that the individuals with the same error rates may select different features is not taken into consideration.

On the one hand, for two individuals with the same fitness value, the number of features they select may be different. For example, there are two individuals $x_1 = [10011]$ and $x_2 = [11111]$ with the same fitness value, and x_2 selects five features, which is more than x_1. If x_2 is chosen to instruct the current individual to update, it increases the dimension of the solution, which is not conducive to minimize the number of selected features.

On the other hand, for two individuals with both the same error rates and the same number of selected features, the features they select may be different. For example, there are two individuals $x_3 = [01011]$ and $x_4 = [11100]$ with the same fitness value and the same number of selected features. However, they select different features. Specifically, x_3 selects the second, the fourth and the fifth features, while x_4 selects the first, the second and the third features.

Therefore, a new update mechanism is proposed to increase the update frequency of food sources and minimize the number of selected features. Let $\mathrm{sum}(x)$ represent the number of features selected by individual x, and $\mathrm{fitness}(x)$ be the fitness value of the individual (the smaller the better). Therefore, individuals are updated in the following way:

(1) If $\mathrm{fitness}(x^{t\prime}) < \mathrm{fitness}(x^t)$, then $x^{t+1} = x^{t\prime}$;
(2) If $\mathrm{fitness}(x^{t\prime}) = \mathrm{fitness}(x^t)$ and $\mathrm{sum}(x^{t\prime}) < \mathrm{sum}(x^t)$, then $x^{t+1} = x^{t\prime}$;
(3) If $\mathrm{fitness}(x^{t\prime}) = \mathrm{fitness}(x^t)$ and $\mathrm{sum}(x^{t\prime}) = \mathrm{sum}(x^t)$, then $x^{t+1} = x^{t\prime}$;
(4) Else, $x^{t+1} = x^t$.where x^t and $x^{t\prime}$ denote the current individual and its offspring (i.e., the candidate individual), respectively. x^{t+1} denotes the individual entering the next iteration. To give readers a clearer understanding of this mechanism, we give an example, as shown in Fig. 2:

There are nine cases of whether an individual updates or not in Fig. 2. The first three cases indicate that if the fitness value of x^t is larger than $x^{t\prime}$, it will be replaced by $x^{t\prime}$ regardless of the number of features it selects, which is shown as case 3–6. If x^t has the same fitness value with $x^{t\prime}$ and its number of selected features is the same as or more than $x^{t\prime}$, it will also be replaced by $x^{t\prime}$, otherwise it will enter the next iteration directly. In another three cases, x^t with smaller fitness value than $x^{t\prime}$ will enter the next iteration without update. The impact of the new update mechanism on the performance of the algorithm will be analyzed in the section of experimental studies.

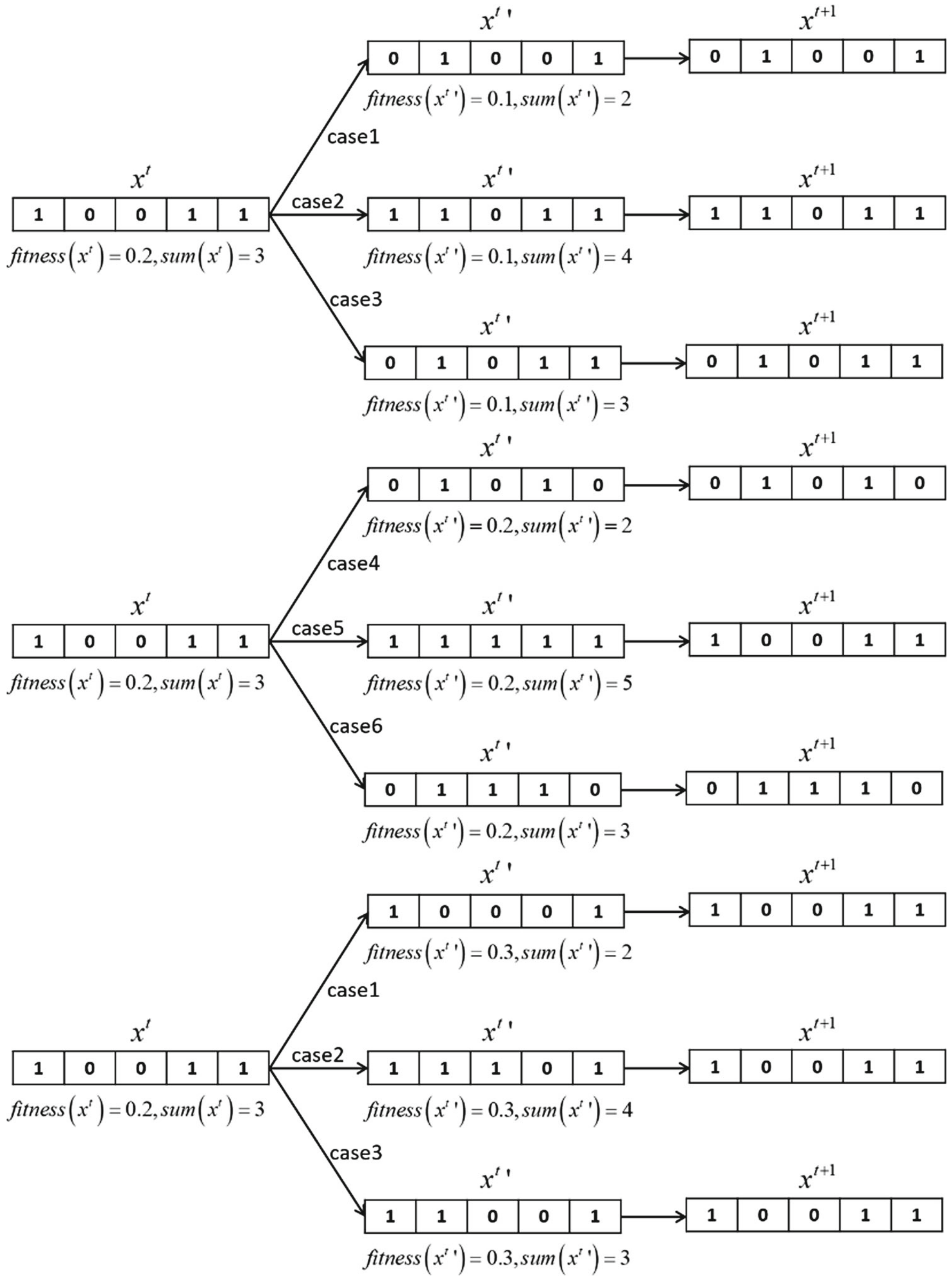
4.4 A self-adaptive level-based learning artificial bee colony algorithm

Notably, ABC algorithm was originally proposed for continuous optimization. To make the ABC algorithm be suited for feature selection, we introduce a transfer function to convert the population from continuous values into discrete values. This idea has been showed to be effective in some feature selection methods (Ghamisi et al. 2014; Mafarja et al. 2018; Mohammadi and Abadeh 2014; Zhang et al. 2015). The transfer function is given as follows:

$$x_{i,d} = \begin{cases} 1, & \mathrm{sigmoid}(x_{i,d}) > \mathrm{rand} \\ 0, & \mathrm{otherwise} \end{cases} \tag{8}$$

where $x_{i,d}$ is the position of the ith bee in dimension d. rand is a random number in [0, 1]. The function of sigmoid() is formulated as

Fig. 2 An example of an individual updating by using the new mechanism



case1
case2
case3
case4
case5
case6
$fitness(x^t) = 0.2, sum(x^t) = 3$
$fitness(x^{t}{}') = 0.1, sum(x^{t}{}') = 2$
$fitness(x^{t}{}') = 0.1, sum(x^{t}{}') = 4$
$fitness(x^{t}{}') = 0.1, sum(x^{t}{}') = 3$
$fitness(x^{t}{}') = 0.2, sum(x^{t}{}') = 2$
$fitness(x^{t}{}') = 0.2, sum(x^{t}{}') = 5$
$fitness(x^{t}{}') = 0.2, sum(x^{t}{}') = 3$
$fitness(x^{t}{}') = 0.3, sum(x^{t}{}') = 2$
$fitness(x^{t}{}') = 0.3, sum(x^{t}{}') = 4$
$fitness(x^{t}{}') = 0.3, sum(x^{t}{}') = 3$

$$\text{sigmoid}(\alpha) = \frac{1}{1 + e^{-10(\alpha - 0.5)}} \tag{9}$$

We combined the novel level-based learning mechanism, the self-adaptive method for determining the number of levels and the new update mechanism to develop an improved artificial bee colony algorithm, namely self-adaptive level-based learning artificial bee colony algorithm (SLLABC) for feature selection. The pseudo code of the SLLABC algorithm is shown by Algorithm 1.

Algorithm 1: The pseudo code of Self-adaptive Level-based Learning Artificial Bee Colony Algorithm

```
Input: Population size $NP$, Maximum number of iterations $MaxIt$, Abandonment limit $L$, $counter$=0, $t$=0.
Output: The optimal food source $x_{best}$, the best fitness value $f(x_{best})$.
Initialize the population $x_i$,($i=1,2,...,NP$) by using Eq. (1).
Evaluate the fitness value of each individual by using Eq. (10).
Calculate the average diversity of the population $Ave_0$ by using Eq. (5).
While $t < MaxIt$ do
    Calculate the number of levels $NL$ by using Eq. (6).
    Sort all individuals in ascending order of fitness values and divide them into $NL$ levels.
    % Employed bee phase
    For each individual $x_i$ in Level 1 do
        Randomly select one different individual $x_k$ in this level.
        Generate a new individual according to Eq. (2) and convert it into discrete values by using Eq. (8).
        Evaluate the fitness value of the new individual by using Eq. (10).
        Update $x_i$ according to the new update mechanism, and increase its counter by 1 if not update.
    End
    For each individual $x_i$ in other levels do
        Randomly select one different individual $x_k$ in higher levels.
        Generate a new individual according to Eq. (2) and convert it into discrete values by using Eq. (8).
        Evaluate the fitness value of the new individual by using Eq. (10).
        Update $x_i$ according to the new update mechanism, and increase its counter by 1 if not update.
    End
    Calculate the selection probability of each individual by using Eq. (3).
    % Onlooker bee phase
    For each onlooker bee do
        Select a food source $x_i$ according to the selection probability by roulette-wheel scheme.
        Randomly select one different individual $x_k$ in the whole population.
        Generate a new individual according to Eq. (2) and convert it into discrete values by using Eq. (8).
        Evaluate the fitness value of the new individual by using Eq. (10).
        Update $x_i$ according to the new update mechanism, and increase its counter by 1 if not update.
    End
    % Scout bee phase
    For each individual in the population do
        If $counter >= L$ then
            Initialize a new individual to replace the current one.
            Evaluate the fitness value of the new individual by using Eq. (10).
        End
    End
End
Output $x_{best}$ and $f(x_{best})$.
```

4.5 Complexity analysis

According to Algorithm1, the SLLABC can be divided into four phases: initialization, employed bee phase, onlooker bee phase and scout bee phase. The time complexity of SLLABC is analyzed as follows.

(1) Initialization can be finished in linear time scale, i.e., $O(\mathrm{NP} \times D)$.

(2) Employed bee phase includes four parts: sort and division, selection, update, and evaluation. In fact, it takes $O(\mathrm{NP} \times \log(\mathrm{NP}) + \mathrm{NP})$ to sort all individuals and divide them into NL levels in each iteration. The selection operator executes $O(\mathrm{NP})$ basic operations. During the update of individuals in all levels, it takes $O(\mathrm{NP} \times D)$ Evaluation operator needs $O(\mathrm{NP})$ basic operations.

(3) Onlooker bee phase includes four parts: probability calculation, selection, update and evaluation. Update operator has the complexity of $O(\mathrm{NP} \times D)$ and other parts can be implemented with linear time scale, i.e., $O(\mathrm{NP})$.

(4) Scout bee phase includes two parts: initialization and evaluation. The computational complexity of initialization and evaluation are both $O(\mathrm{NP})$ at worst.

Note that for evaluating the individuals, the above analysis only considers the number of the individuals evaluated in each iteration. Overall, SLLABC only takes extra $O(\mathrm{NP} \times \log(\mathrm{NP}) + \mathrm{NP})$ in each iteration compared with the basic ABC algorithm, which takes $O(\mathrm{NP} \times D)$ in each iteration.

As for the space complexity, SLLABC have the same space as basic ABC, i.e., $O(\mathrm{NP} \times D)$.

Therefore, the computational complexity of SLLABC is competitive compared with the basic ABC algorithm.

5 Experimental studies

5.1 Experimental design

To verify the performance of the proposed feature selection algorithm, a series of experiments are conducted on a total of 12 standard datasets. These datasets together with their number of features, instances and classes are listed in Table 1. They were obtained from http://featureselection.asu.edu/datasets.php and http://archive.ics.uci.edu/ml/datasets.php. Among them, there are microarray gene expression data, image (face) detection data and email text data etc. These datasets have been processed by the providers in advance. In addition, they not only come from different applications fields, but also the number of features varies from 310 to 22,283, instances vary from 62 to 165, and classes vary from 2 to 15, which makes our experiments universal.

Table 1 Properties of the datasets

Dataset	Features	Samples	Classes
LSVT	310	126	2
Yale	1024	165	15
Colon	2000	62	2
SRBCT	2308	83	4
DBWorld	4702	64	2
Leukemia1	5327	72	3
DLBCL	5469	77	2
ALLAML	7129	72	2
Pixraw10P	10,000	100	10
Prostate	10,509	102	2
Leukemia2	11,225	72	3
GLI_85	22,283	85	2

A suitable classifier is important to evaluate the feature subsets. Because of its effectiveness, the well-known K-Nearest Neighbor (KNN) (Liao and Vemuri 2002) is adopted to evaluate the performance of all algorithms in our experimental studies, where $K = 5$. In order to reduce the risk of overfitting, the average classification error rate by tenfold Cross Validation is taken as the fitness value. The fitness function is shown as follows:

$$\mathrm{fitness} = \frac{\sum_{i=1}^{10} \mathrm{error}_i}{10} \tag{10}$$

where, error_i is calculated as follows:

$$\mathrm{error}_i = \frac{\text{Number of Misclassified Samples}}{\text{Total Number of Samples}} \tag{11}$$

The proposed algorithm and other algorithms in comparison are in MATLAB language. The population size of all algorithms is 50, the maximum number of iterations is 100. For fair comparison, each algorithm runs 10 times independently. All experiments are executed on a computer with Intel(R) Core (TM) i5-7500 CPU and 16 GB RAM.

5.2 Experimental results and discussions

As previously discussed, this paper employed three new strategies. This section performs an extensive analysis on these three new strategies. After that, the proposed SLLABC algorithm is compared with the other eight algorithms for feature selection to evaluate its comprehensive performance.

5.2.1 Discussion on the diversity of two level-based learning mechanisms and its influence

First of all, we investigate the potency of individuals in level 1 learning from each other, and compare the diversity curve, convergence curve and the average feature subset size of the two mechanisms, denoted as SLLABC and SLLABC-1, respectively. Where, the individuals in level 1 are directly passed to the next iteration without updating in the SLLABC-1 algorithm, while in the SLLABC algorithm, they learn from each other in level 1 for updating.

Figure 3 shows diversity curve of SLLABC and SLLABC-1. To avoid contingency, we run these two algorithms 10 times independently to obtain the mean diversity in each iteration.

In terms of the value of diversity, the diversities of SLLABC are higher than that of SLLABC-1 on the dataset LSVT, Yale, and Colon in the later iterations. However, the diversity of SLLABC is lower on the dataset Leukemia1. On the most datasets, the diversity value of SLLABC algorithm is not significantly different from that of SLLABC-2. In terms of the coincidence degree of the

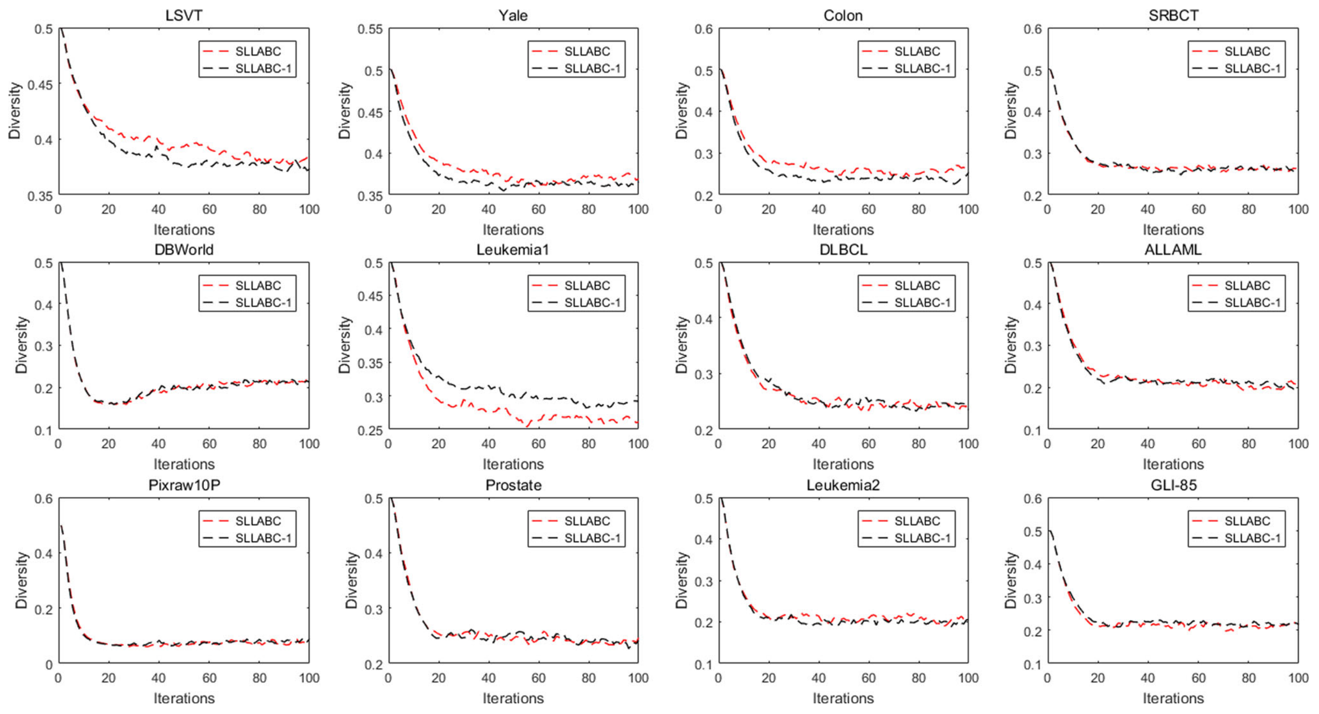


Fig. 3 The Diversity Curve of SLLABC and SLLABC-1

curve, the diversity curve on all data sets of SLLABC almost coincides with that of SLLABC-1 in the first 10 iterations. In addition, for datasets with a small number of features, LSVT, Colon and Leukemia1, the curves of the two algorithms are significantly different, while there is almost no difference between the curves of the two algorithms on the datasets with higher dimensions such as ALLAML, GLI_85 and Pixraw10P. The reason for this phenomenon may be that the number of individuals in level 1 is only a very small part of the whole population, especially in the high-dimensional feature selection, and in late iterations when the diversity of the population is very low, the individuals learning from each other does not play an important role in diversity.

Figure 4 shows the feature subset size of SLLABC and SLLABC-1. We can see that SLLABC selected less features than SLLABC-1 on 6 datasets, On the other 6 datasets, i.e., LSVT, SRBCT, Leukemia1, ALLAML,

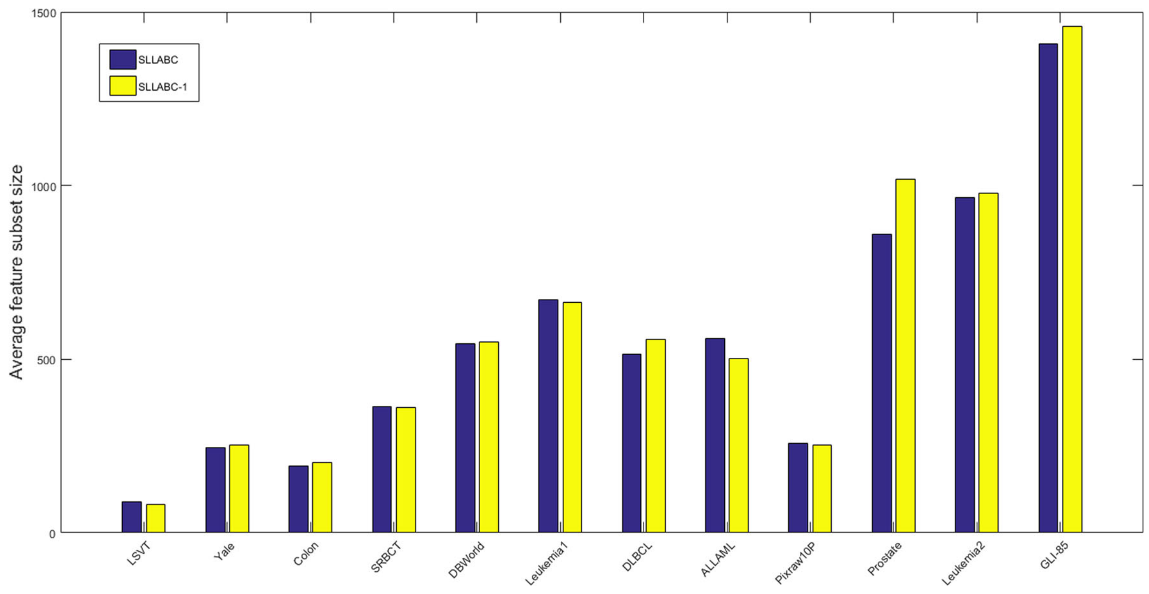


Fig. 4 The Average Feature Subset Size of SLLABC and SLLABC-1

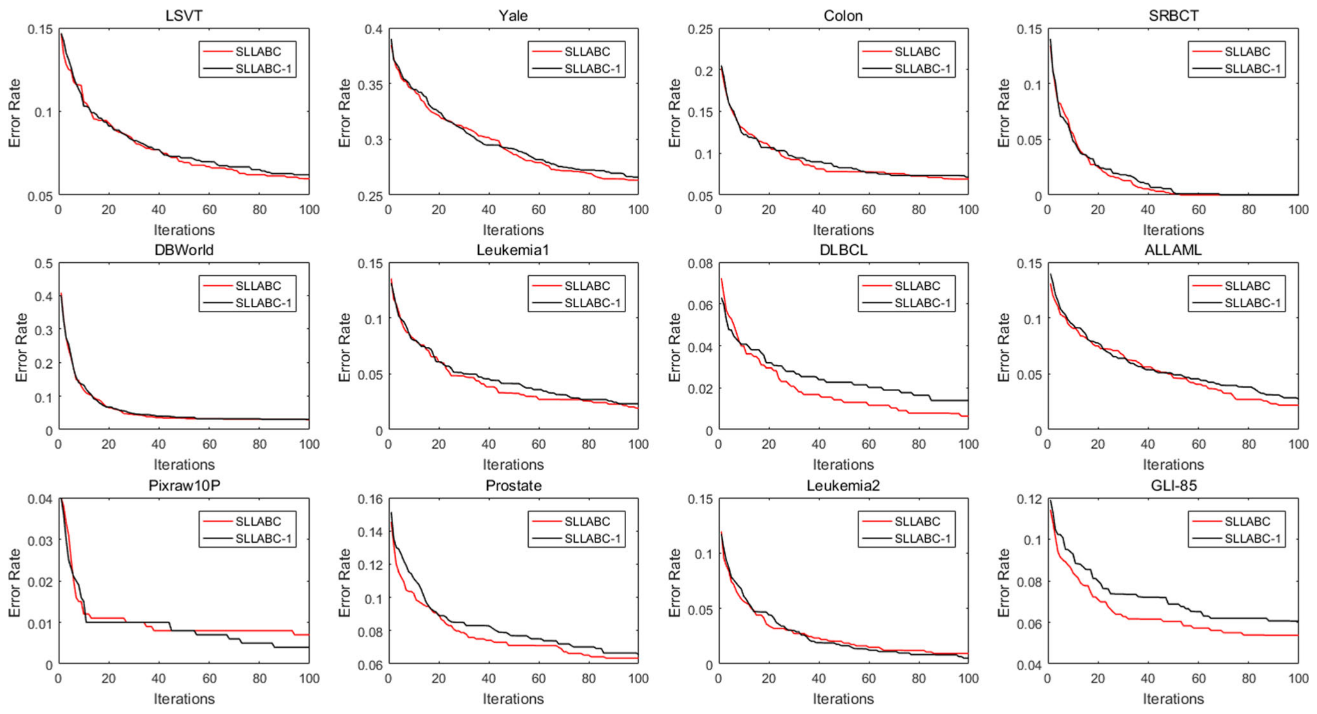


Fig. 5 The Convergence Curve of SLLABC and SLLABC-1

Pixraw10P and GLI_85, SLLABC selected more features than SLLABC-1.

The convergence curves of the two algorithms are plotted in Fig. 5. It shows the decline of the error rate. On most datasets, the effect of SLLABC on reducing the error rate in the early iterations is not obvious, but it is always able to achieve a lower error rate than SLLABC-1 in the middle-and-later iterations. Specially, although SLLABC-1 obtained smaller feature subsets on the above six datasets in Fig. 4, the corresponding performance on error rate in Fig. 5 is not better than SLLABC. This may be due to that the individuals in level 1 only account for a small proportion of the whole population at the early stage of the evolution, and learning from each other cannot enhance the diversity significantly, but can make the individuals in the level 1 jump out of the local optima, so then the possibility of achieving a lower error rate was improved.

It is generally accepted that the exploration and exploitation ability of EAs can be implied by the diversity of the population, the lower diversity is, the less the exploration potential of individuals is and the more on exploiting potential is, and vice versa. The error rate convergence curve verifies this view as well. Therefore, according to the discussion above, we draw such a conclusion that the mechanism of individuals learning from each other in the first level does not play an important role in improving the exploration ability, but can enhance the exploitation performance of the population in most cases.

5.2.2 Discussion on the diversity of two methods of determining level number and its influence

Secondly, in order to study the potency of the self-adaptive method for determining the number of levels, Figs. 6, 7, 8 show the comparison between SLLABC and SLLABC-2, the former using the self-adaptive level-based learning method and the latter using the original dynamic level-based learning (DLL) method. The DLL method needs to give a definite set of level numbers S. In order to study the influence of different S on classification performance, this study uses two sets, one is the empirical value $S_1 = [4, 6, 8, 10, 20, 50]$ obtained by Yang, and the other $S_2 = [5, 10, 20, 30, 40, 50]$ is listed at random. The algorithms using S_1 and S_2 are referred here as SLLABC-2-1 and SLLABC-2-2, respectively. Noted that, SLLABC-2 contains SLLABC-2-1 and SLLABC-2-2.

AS shown in Fig. 6, in the first 10 iterations, the diversity curves of the three algorithms are not significantly different on all data sets. However, at the middle and the end of run, the diversity curves of SLLABC are lower than those of SLLABC-2 on Yale, SRBCT, DLBCL and GLI_85. In addition, there are significant differences in diversity curves between SLLABC-2-1 and SLLABC-2-2, especially on the datasets Yale, ALLAML and Prostate.

Figure 7 shows the number of the features selected by the three algorithms. The feature subset of SLLABC is smaller than that of SLLABC-2-1 on six datasets, and smaller than that of SLLABC-2-2 on five data sets. Only on

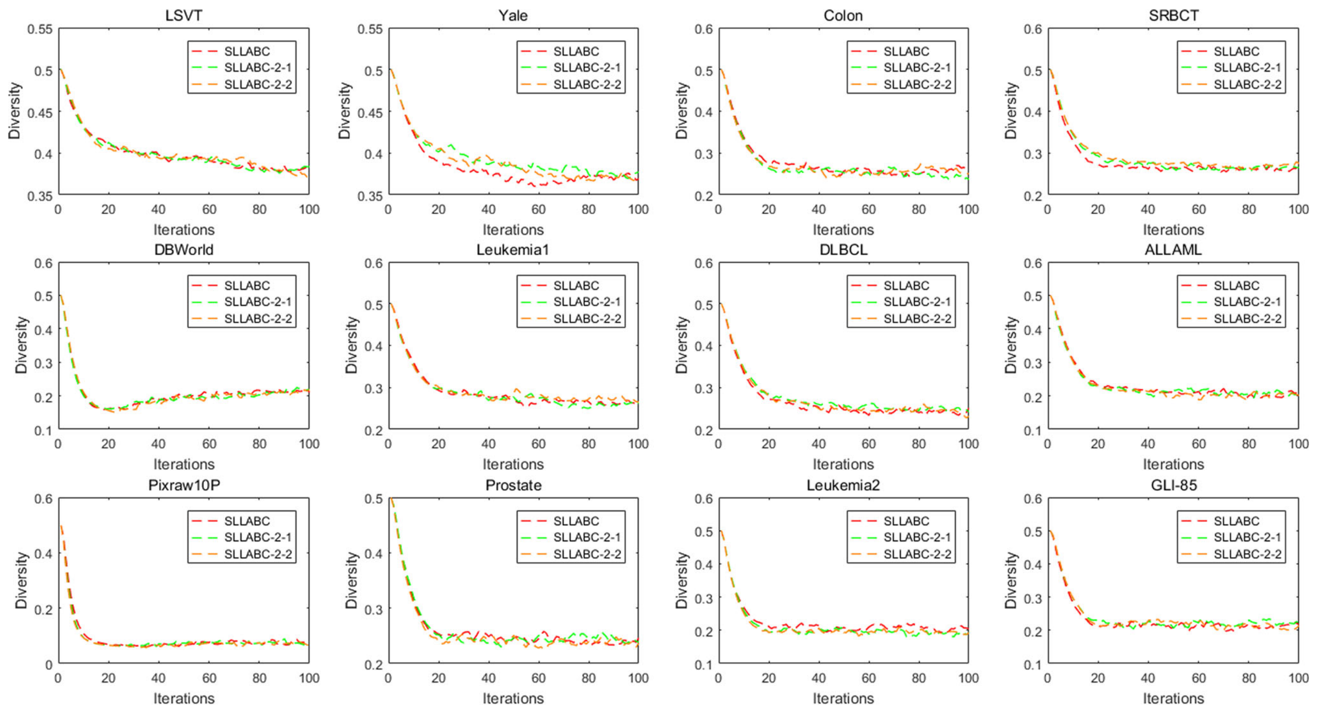


Fig. 6 The Diversity Curve of SLLABC, SLLABC-2-1 and SLLABC-2-2

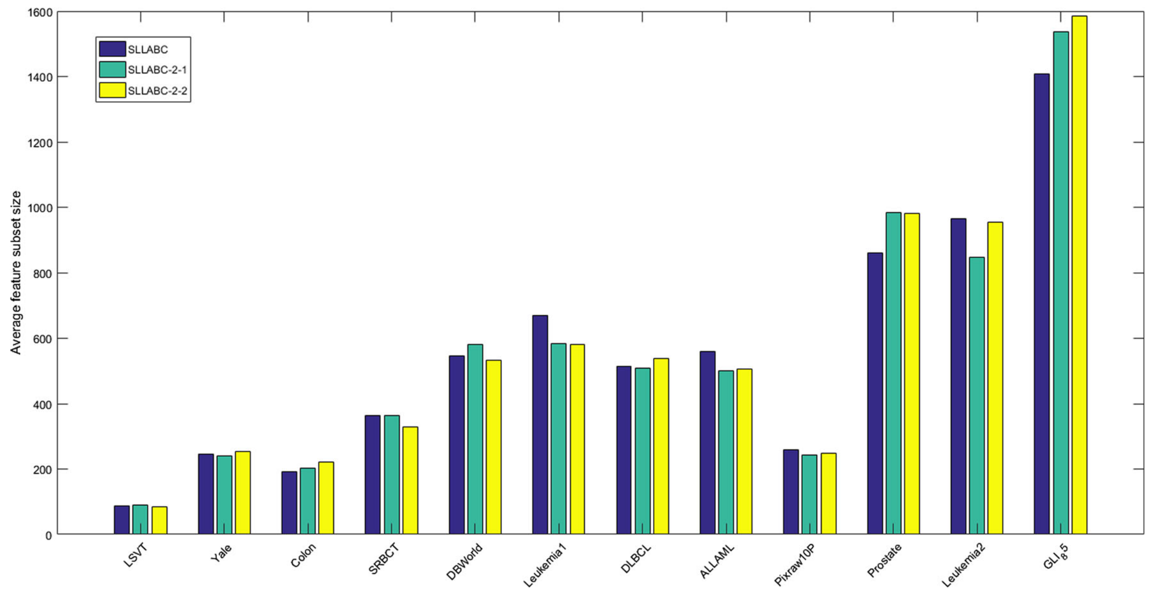


Fig. 7 The Average Feature Subset Size of SLLABC, SLLABC-2-1 and SLLABC-2-2

a few data sets, i.e., Leukemia1 and ALLAML, the feature subset of SLLABC is larger than that of both SLLABC-2-1 and SLLABC-2-2. Figure 8 plots the decline of the error rate by the three algorithms. We find that the decline rates of the three methods are similar in the early stage, however, the convergence rate of SLLABC-2-1 and SLLABC-2-2 are slower than the SLLABC at the middle and end of run on the most datasets, which indicates that the self-adaptive strategy for NL has a better exploitation ability. It is also worth mentioning that, on the datasets LSVT, ALLAML, Pixraw10P and Prostate, the final error rate of SLLABC is lower than that of SLLABC-2-1 but higher than that of SLLABC-2-2. On the dataset Leukemia2, the final error rate of SLLABC-2-2 is lower than that of the other two compared algorithms.

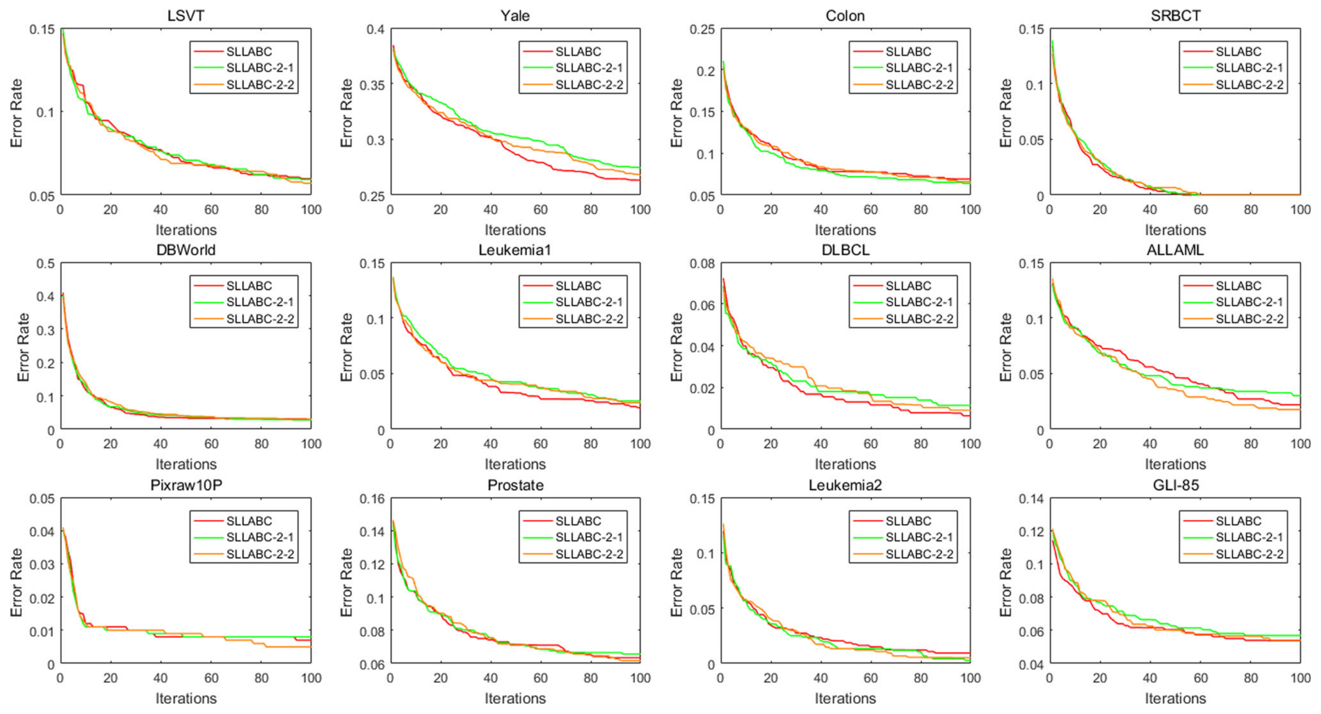


Fig. 8 The Convergence Curve of SLLABC, SLLABC-2-1 and SLLABC-2-2

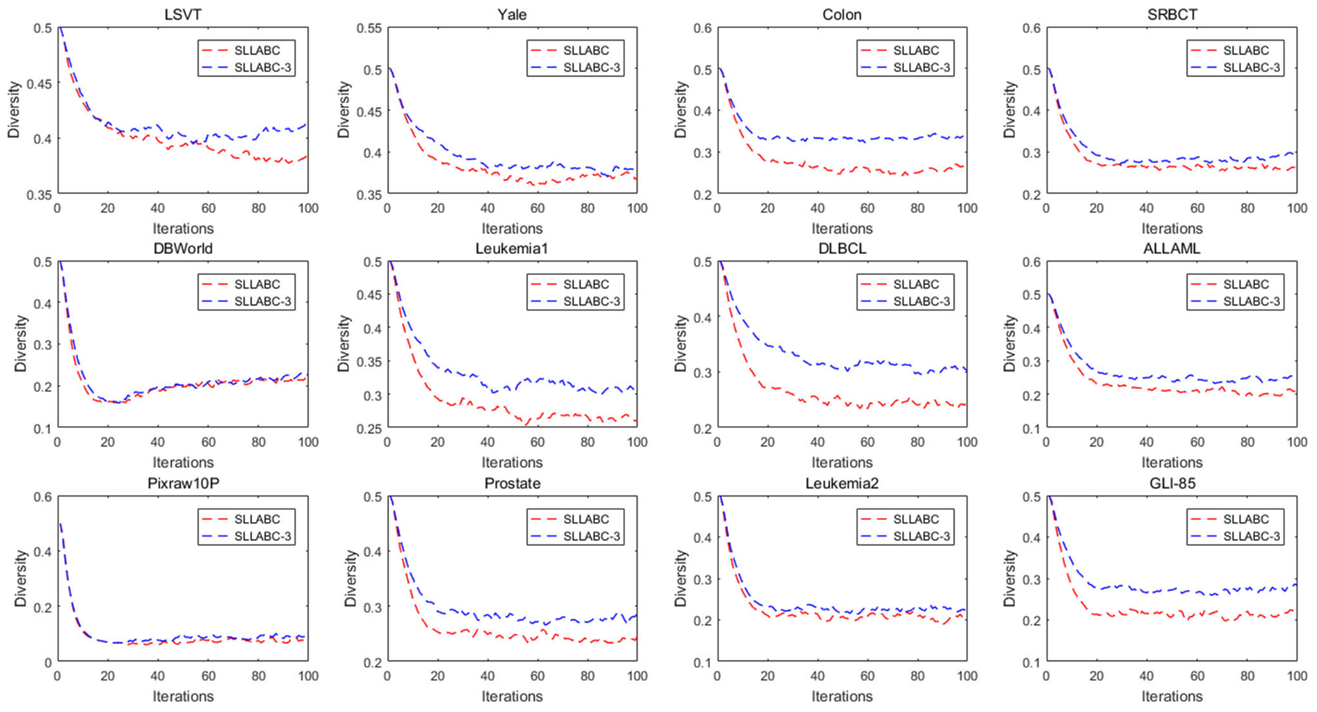


Fig. 9 The Diversity Curve of SLLABC and SLLABC-3

Comparing the three curves in Figs. 6 and 8, we find that a lower diversity may not result in a higher error rate. On the contrary, SLLABC achieves a lower diversity in Fig. 6 and a lower error rate as well in Fig. 8, which demonstrates that the exploration of SLLABC-2-1 and SLLABC-2-2 are overemphasized and thus resulting in a poor exploitation, while SLLABC is able to compromise exploration and exploitation better than SLLABC-2-1 and SLLABC-2-2. In

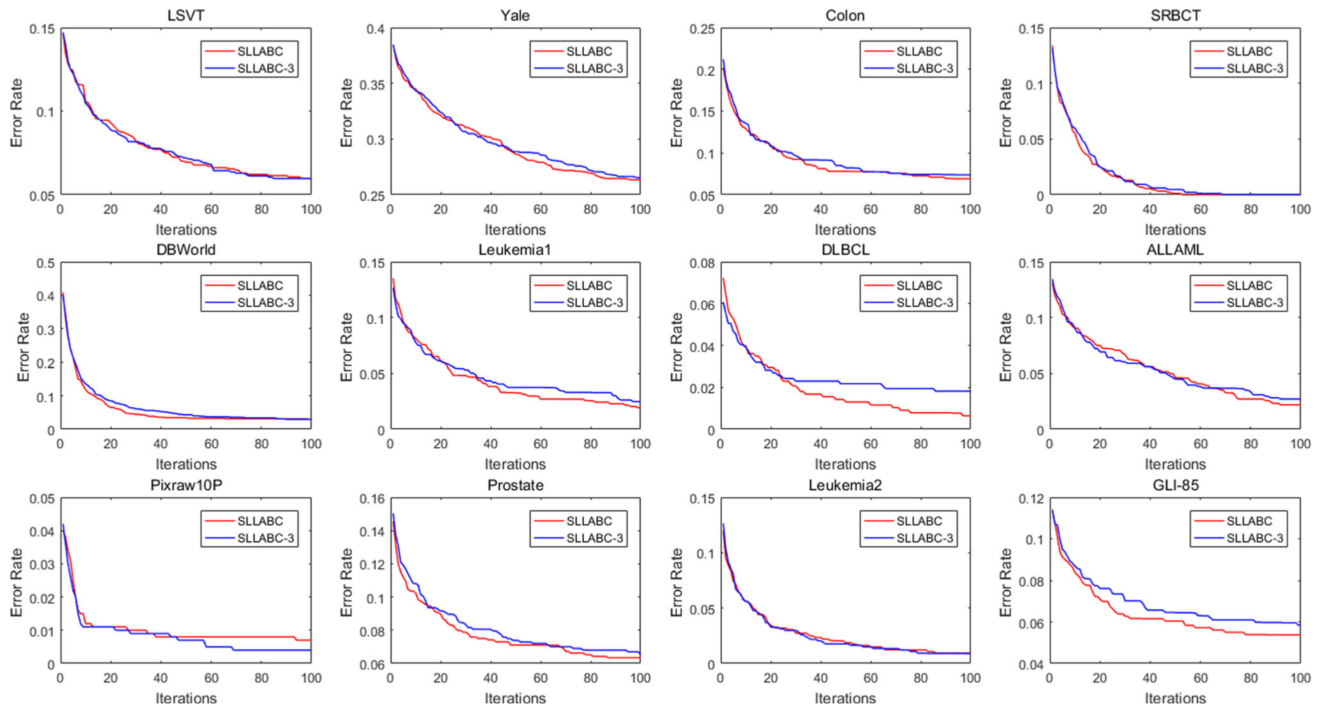


Fig. 10 The Convergence Curve of SLLABC and SLLABC-3

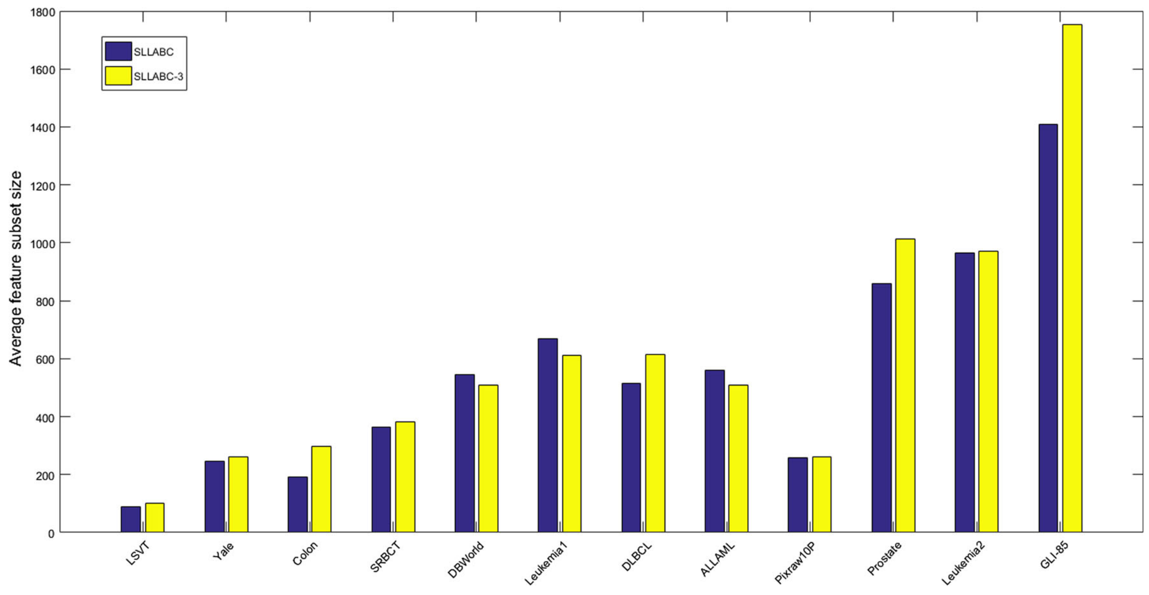


Fig. 11 The Average Feature Subset Size of SLLABC and SLLABC-3

addition, the SLLABC-2-1 and SLLABC-2-2 algorithms using different *S* also have great differences in the number of features and error rate. It indicates that how to determine the *S* in DLL strategy is very important, which may need to be summarized by different experiments according to different problems. Fortunately, the performance of the SLLABC using the adaptive strategy proposed in this paper is better than that of the other two algorithms using the original dynamic strategy in most cases. All in all, the self-adaptive strategy for *NL* is promising for the proposed algorithm.

Table 2 Experimental parameters and settings

Method	Parameter	Setting
All algorithms	Population size, NP	50
	Maximum number of iterations, $MaxIt$	100
	Number of runs	10
CSO	The influence factor of the mean position, φ	0.1
2D_UPSO	Inertia weight, ω	0.729
	Acceleration constant, c_1, c_2	1.49,1.49
	unification factor, μ	[0.2,0.4]
	Refresh Gap, RG	3
VS_CCPSO	The number of feature subspaces, M	2 (on LSVT), 4 (on Yale), 8 (on Colon, SRBCT), 10 (on other datasets)
	Convergence speed control factor, $prob$	0.7
ALO_GWO	Adjustment constant, ω	ω = 2 when $t > 0.1$ T, ω = 3 when $t > 0.5$ T, ω = 4 when $t > 0.75$ T, ω = 5 when $t > 0.9$ T, ω = 6 when $t > 0.95$ T
MbGWO-SFS	Mutation ratio	0.1
	Crossover ratio	0.9
	Maximum diffusion level	1
AC_ABC	Pheromone exponential weight, α	1
	Evaporation rate, ρ	0.7
	Heuristic exponential weight, β	0.8
	φ in ACO	0.3
	Initial Pheromone, τ_0	0.2
	φ in ABC	0.4
HLBDA	Personal learning rate,pl	0.4
	Global learning rate,gl	0.7
FRGA	/	/
SLLABC	Abandonment limit L	25

5.2.3 Discussion on the diversity of two update mechanisms and its influence

To investigate the influence of the new update mechanism on the performance of feature selection, SLLABC is compared with SLLABC-3, the former adopts the new update mechanism and the latter adopts the original update method, i.e., the size of feature subset is not taken into account.

As can be seen Fig. 9, SLLABC-3 maintains a higher diversity than SLLABC on each dataset, and its value of diversity remains high. It is worth mentioning that at the end of run, the diversity values of SLLABC-3 on the first three data sets are not less than 0.4, which is not conducive to the fast convergence of the algorithm. This is verified by the convergence curve in Fig. 10. In Fig. 10, SLLABC achieves a lower or the same error rate at the end of run on most datasets. The only exception is found on the Pix-raw10P dataset, which might be due to the internal attributes and samples of the dataset and we cannot confirm at present.

After that, to investigate the differences between the two update mechanisms, we draw a histogram of the average feature subset size shown as Fig. 11. We can see that SLLABC-3 outperforms SLLABC on only two datasets, i.e., DBWorld and ALLAML. Therefore, the new update mechanism is promising in dimensionality reduction.

Table 3 Comparison of classification error rate between SLLABC and other compared algorithms

Datasets	Index	CSO	2D_UPSO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC	SLLABC
LSVT	Worst	0.081	0.097	**0.063**	0.078	0.111	0.332	0.073	0.08	0.072
	Mean±std	0.063±0.008	0.083±0.014	**0.046±0.011**	0.065±0.009	0.099±0.012	0.277±0.046	0.059±0.006	0.065±0.008	0.060±0.006
	best	0.055	0.062	**0.032**	0.054	0.072	0.2224	0.054	0.056	0.054
Yale	Worst	0.32	0.333	**0.242**	0.291	0.357	0.304	0.3132	0.315	0.285
	Mean±std	0.295±0.015	0.310±0.015	**0.224±0.010**	0.263±0.018	0.337±0.012	0.285±0.016	0.298±0.009	0.288±0.015	0.263±0.012
	best	0.268	0.283	**0.212**	0.23	0.32	0.261	0.2842	0.266	0.247
Colon	Worst	0.176	0.126	0.097	0.088	0.155	0.148	0.1429	0.157	**0.081**
	Mean±std	0.113±0.027	0.091±0.024	0.071±0.016	0.0693±0.008	0.115±0.029	0.114±0.022	0.109±0.0232	0.119±0.024	**0.069±0.008**
	best	0.081	0.062	**0.048**	0.064	0.062	0.076	0.0667	0.095	0.06
SRBCT	Worst	0.049	0.036	0.036	0.025	0.096	0.039	0.069	0.063	**0**
	Mean±std	0.033±0.015	0.018±0.012	0.029±0.008	0.005±0.009	0.058±0.032	0.0039±0.0001	0.039±0.013	0.035±0.014	**0**
	best	0	0	0.012	0	0.011	0.004	0.024	0.022	**0**
DBWorld	Worst	0.255	0.107	0.063	0.062	0.167	0.1095	0.2214	0.3857	**0.033**
	Mean±std	0.126±0.054	0.08±0.019	0.048±0.005	0.034±0.012	0.113±0.035	0.075±0.026	0.156±0.039	0.361±0.013	**0.030±0.005**
	best	0.062	0.048	0.047	0.017	0.079	0.031	0.1071	0.345	**0.017**
Leukemia1	Worst	0.084	0.096	**0.028**	0.057	0.125	0.0313	0.098	0.125	0.043
	Mean±std	0.062±0.014	0.058±0.020	0.028±0.00	0.034±0.020	0.066±0.028	0.020±0.007	0.074±0.012	0.096±0.016	**0.019±0.012**
	best	0.039	0.027	0.028	0	0.029	0.0143	0.055	0.068	**0**
DLBCL	Worst	0.075	0.05	0.091	0.038	0.063	0.0432	0.079	0.077	**0.014**
	Mean±std	0.051±0.012	0.034±0.130	0.044±0.019	0.021±0.014	0.038±0.013	0.026± 0.013	0.043±0.014	0.054±0.014	**0.006±0.007**
	best	0.038	0.013	0.026	0	0.025	0.0044	0.025	0.038	**0**
ALLAML	Worst	0.113	0.084	**0.042**	0.082	0.109	0.128	0.125	0.134	0.043
	Mean±std	0.102±0.008	0.059±0.018	0.028±0.011	0.044±0.026	0.071±0.026	0.099±0.014	0.109±0.014	0.117±0.011	**0.022±0.015**
	best	0.093	0.029	0.014	0	0.025	0.084	0.082	0.095	**0**
Pixraw10P	Worst	0.05	0.01	0.01	0.04	0.04	0.044	0.06	0.04	**0.01**
	Mean±std	0.041±0.003	0.01±0	0.01±0	0.012±0.010	0.026±0.015	0.044±0	0.042±0.008	0.040±0	**0.007±0.005**
	best	0.04	0.01	0.01	0	0.01	0.044	0.03	0.04	**0**
Prostate	Worst	0.126	0.097	0.078	0.079	0.117	0.091	0.126	0.146	**0.076**
	Mean±std	0.112±0.013	0.083±0.012	0.063±0.011	0.066±0.011	0.088±0.021	0.080±0.009	0.113±0.013	0.132±0.012	**0.063±0.006**
	best	0.089	0.057	0.049	**0.049**	0.058	0.063	0.086	0.116	0.057
Leukemia2	Worst	0.082	0.052	0.097	0.029	0.055	0.029	0.084	0.125	**0.029**
	Mean±std	0.061±0.014	0.030±0.119	0.063±0.018	0.015±0.010	0.037±0.016	0.017±0.006	0.065±0.015	0.095±0.016	**0.010±0.009**
	best	0.041	0.014	0.028	0	0	0.013	0.043	0.07	**0**
GLI_85	Worst	0.129	0.083	0.106	**0.071**	0.106	0.087	0.119	0.15	0.074
	Mean±std	0.094±0.015	0.074±0.008	0.081±0.017	0.058±0.008	0.077±0.021	0.068±0.010	0.099±0.014	0.104±0.024	**0.054±0.009**
	best	0.081	0.06	0.047	0.047	0.047	0.052	0.083	0.082	**0.044**

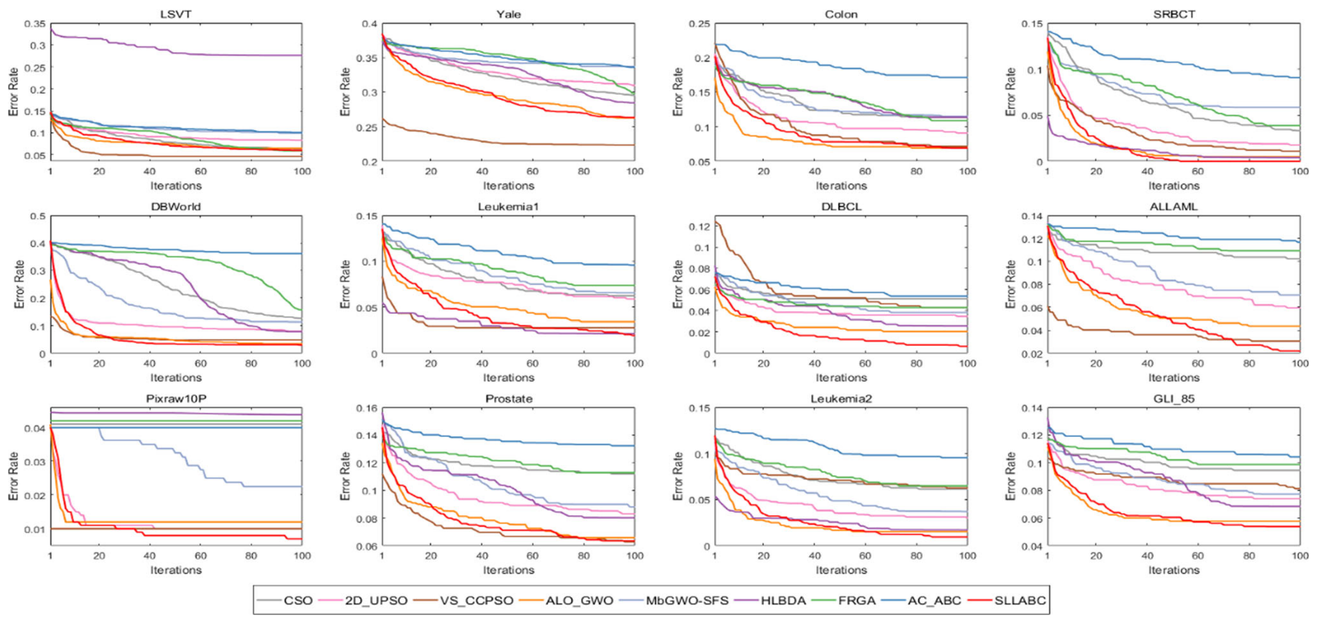


Fig. 12 The convergence curves of SLLABC and other compared algorithms

Table 4 Wilcoxon rank sum test on error rate of SLLABC and the compared algorithms

Datasets	CSO	2D_USPO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC
LSVT	0.70480(=)	0.00130(+)	0.01680(−)	0.23950(=)	0.00018(+)	0.00018(+)	0.73320(=)	0.00018(+)
Yale	0.00220(+)	0.00025(+)	0.00018(−)	0.96980(=)	0.00018(+)	0.01726(+)	0.00024(+)	0.00018(+)
Colon	0.00018(+)	0.04880(+)	0.51450(=)	0.70300(=)	0.00350(+)	0.00044(+)	0.00076(+)	0.00018(+)
SRBCT	0.00006(+)	0.00070(+)	0.00005(+)	0.07790(=)	0.00006(+)	0.00006(+)	0.00059(+)	0.00006(+)
DBWorld	0.00016(+)	0.00016(+)	0.00005(+)	0.53570(=)	0.00016(+)	0.00120(+)	0.00016(+)	0.00016(+)
Leukemia1	0.00020(+)	0.00048(+)	0.01670(+)	0.10930(=)	0.00048(+)	0.84898(+)	0.00016(+)	0.00017(+)
DLBCL	0.00014(+)	0.00034(+)	0.00013(+)	0.02530(+)	0.00046(+)	0.00334(+)	0.00014(+)	0.00015(+)
ALLAML	0.00018(+)	0.00110(+)	0.34050(=)	0.04400(=)	0.00110(+)	0.00018(+)	0.00017(+)	0.00018(+)
Pixraw10P	0.00011(+)	0.07670(=)	0.10580(=)	0.19360(=)	0.01510(+)	0.00004(+)	0.00012(+)	0.00010(+)
Prostate	0.00018(+)	0.00360(+)	0.03340(+)	0.62160(=)	0.00890(+)	0.00356(+)	0.00018(+)	0.00018(+)
Leukemia2	0.00017(+)	0.00170(+)	0.00022(+)	0.18570(=)	0.00220(+)	0.01072(+)	0.00017(+)	0.00017(+)
GLI_85	0.00017(+)	0.00070(+)	0.00690(+)	0.12880(=)	0.01040(+)	0.03020(+)	0.00017(+)	0.00025(+)

‘ ± ’: The result obtained by SLLABC is significantly better/worse than the compared algorithm. ‘ = ’: There is no statistically significant difference e between the results obtained by SLLABC and the compared algorithm

5.2.4 Comparison of the performance between SLLABC and other algorithms

In this section, we further investigate the performance of SLLABC algorithm by comparing it with eight feature selection methods, including the popular PSO variants, i.e., CSO (Gu et al. 2018), 2D_UPSO (Hafiz et al. 2018) and VS_CCPSO (Song et al. 2020), the novel GWO variants, i.e., ALO_GWO (Zawbaa et al. 2018) and MbGWO-SFS (El-Kenawy et al. 2020), a variant of Dragonfly Algorithm, i.e., HLBDA (Too and Mirjalili 2021), a GA variant, i.e., FRGA (Too and Abdullah 2021), and an ABC variant, i.e., AC_ABC (Shunmugapriya and Kanmani 2017). In particular, CSO, VS_CCPSO and ALO_GWO have performed well in dealing with high-dimensional problems. Moreover, for the compared algorithms, the parameters are set as recommended in the corresponding papers. Table 2 gives the parameter settings of the above algorithms.

First of all, we plot the convergence curves of the nine algorithms, as shown in Fig. 12. Evidently, in terms of

Table 5 Comparison of the size of subsets selected by SLLABC and other compared algorithms

Datasets	Index	CSO	2D_UPSO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC	SLLABC
LSVT	Worst	176	167	**43**	134	175	83	164	170	116
	Mean±std	151.5±11.18	120.9±32.33	**32.4±6.54**	102.5±16.13	114±35.05	63.7±12.45	148.3±8.18	150.9±9.15	88.1±24.84
	best	137	60	**22**	80	80	51	139	140	40
Yale	Worst	524	494	**210**	304	514	503	529	543	283
	Mean±std	504.1±16.48	265.9±130.20	**133±31.85**	236.4±40.42	349.4±131.51	478.3±17.76	500.5±14.71	506.3±23.16	246.5±23.31
	best	474	**69**	103	164	155	449	474	476	213
Colon	Worst	1036	409	**241**	468	841	852	1036	1013	272
	Mean±std	983±24.57	**155.3±128.09**	207.8±17.99	261.2±111.75	316±248.93	811.3±34.03	989.5±27.50	995.6±14.06	192.6±59.23
	best	951	38	178	122	**16**	762	957	975	107
SRBCT	Worst	1160	462	**342**	615	1043	943	1178	1178	453
	Mean±std	1127.1±20.12	288.2±99.15	**258.8±36.10**	408.3±112.17	504.5±326.15	899.6±37.23	1145.1±23.69	1136.2±23.35	364.4±57.78
	best	1092	175	211	190	**93**	846	1086	1092	262
DBWorld	Worst	2354	640	793	689	**603**	2272	2398	2384	686
	Mean±std	2316.2±25.27	303.6±175.46	670.6±50.42	446.5±161.63	**230.1±181.22**	2212±44.13	2306.8±36.44	2338±27.91	545.5±88.75
	best	2283	85	623	129	**39**	2157	2267	2301	395
Leukemia1	Worst	2733	1969	**862**	1220	1718	2431	2731	2705	996
	Mean±std	2664.6±45.62	826.9±602.85	757.8±70.67	867.4±216.70	1000.6±494.98	2341.4±60.99	2654.8±43.94	2645.6±43.31	**670.3±139.32**
	best	2580	**116**	668	594	301	2223	2578	2588	513
DLBCL	Worst	2788	3169	1108	2067	2234	2496	2795	2770	**733**
	Mean±std	2737.6±29.06	1282±945.43	768.7±156.55	1097.8±518.60	1146.1±554.51	2379.9±72.54	2732.4±39.51	2733.1±27.78	**515.4±152.96**
	best	2697	**84**	546	593	500	2256	2666	2676	325
ALLAML	Worst	3588	880	1788	1401	3556	3289	3668	3660	**787**
	Mean±std	3563.3±29.63	306.8±240.62	1324.6±211.14	846.5±355.27	985.8±1221.53	3149.5±69.91	3551.3±53.23	3537±47.15	**559.6±167.54**
	best	3488	**55**	1012	219	10	3045	3488	3477	261
Pixraw10P	Worst	5102	617	2601	5087	5090	4230	5083	5120	**345**
	Mean±std	5015.6±44.26	260.4±159.31	2549.4±32.26	882.1±1494.17	3142.8±1726.43	4201.3±33.95	5014.2±44.08	5006.7±57.34	**257.8±47.35**
	best	4956	**87**	2518	128	843	4121	4945	4941	195
Prostate	Worst	5314	1573	2724	2600	5788	5027	5330	5268	**1221**
	Mean±std	5246.2±34.33	861.7±497.88	1675.9±416.34	1440.3±601.09	1396.3±1798.33	4933.3±69.17	5266.3±54.01	5194.9±50.11	**859.9±224.18**
	best	5190	**165**	1248	737	188	4796	5164	5128	423
Leukemia2	Worst	5706	3812	2613	1842	2852	5191	5661	5676	**1408**
	Mean±std	5627.1±51.48	1160.5±1129.14	1999.7±379.97	1336.5±353.31	1274.4±737.39	5069±81.75	5606.5±45.78	5608.7±44.51	**965.9±292.32**
	best	5551	**109**	1375	503	326	4941	5500	5554	493
GLI_85	Worst	11461	2266	6564	4728	7105	10946	11301	11898	**1011**
	Mean±std	11157.5±123.77	1439.3±666.99	4836.4±1226.11	2971.2±1173.92	3859.5±2095.64	10746.2±94.40	11134±72.322	11682.5±169.07	**1408.2±279.81**
	best	11054	**265**	2557	1587	1231	10640	11049	11327	2028

Table 6 Wilcoxon rank sum test on feature subset size of SLLABC and the compared algorithms

Datasets	CSO	2D_USPO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC
LSVT	0.00018(+)	0.0309(+)	0.00033(−)	0.34430(=)	0.25650(=)	0.00018(−)	0.00018(+)	0.00018(+)
Yale	0.00018(+)	0.9397(=)	0.00018(−)	0.52050(=)	0.14050(=)	0.00018(+)	0.00018(+)	0.00018(+)
Colon	0.00018(+)	0.2123(=)	0.67760(=)	0.24130(=)	0.16200(=)	0.00018(+)	0.00018(+)	0.00018(+)
SRBCT	0.00018(+)	0.0640(=)	0.00058(−)	0.27290(=)	0.57080(=)	0.00018(+)	0.00018(+)	0.00018(+)
DBWorld	0.00018(+)	0.0036(−)	0.00460(+)	0.0890(=)	0.00220(−)	0.00018(+)	0.00018(+)	0.00018(+)
Leukemia1	0.00018(+)	0.8501(=)	0.05850(=)	0.03760(+)	0.14030(=)	0.00018(+)	0.00018(+)	0.00018(+)
DLBCL	0.00018(+)	0.0091(+)	0.00220(+)	0.00100(+)	0.00830(+)	0.00018(+)	0.00018(+)	0.00018(+)
ALLAML	0.00018(+)	0.0113(−)	0.00019(+)	0.02570(+)	0.90970(=)	0.00018(+)	0.00018(+)	0.00018(+)
Pixraw10P	0.00018(+)	0.6774(=)	0.00018(+)	0.12120(=)	0.00005(+)	0.00018(+)	0.00018(+)	0.00018(+)
Prostate	0.00018(+)	0.6776(=)	0.00018(+)	0.01730(+)	0.47270(=)	0.00018(+)	0.00018(+)	0.00018(+)
Leukemia2	0.00018(+)	0.6775(=)	0.00024(+)	0.01130(+)	0.47250(=)	0.00018(+)	0.00018(+)	0.00018(+)
GLI_85	0.00018(+)	0.4274(=)	0.00018(+)	0.00044(+)	0.00280(+)	0.00018(+)	0.00018(+)	0.00025(+)

‘ ± ’: The result obtained by SLLABC is significantly better/worse than the compared algorithm. ‘ = ’: There is no statistically significant difference e between the results obtained by SLLABC and the compared algorithm

convergence behavior, AC_ABC, MbGWO-SFS, CSO, HLBDA and FRGA show unsatisfactory performance, while SLLABC algorithm can accelerate to detect the global best subset over the complex feature space. Even though ALO_GWO algorithm shows a very fast convergence speed in the early iterations, it starts to decelerate after 20 iterations. In contrast, SLLABC is able to maintain a fast convergence over the course of iterations and can always converge to a smaller error rate than any other algorithm at the end of each run. Moreover, on the dataset LSVT, Leukemia1, DLBCL, ALLAML, Prostate, Leukemia2 and GLI-85, the SLLABC still has a downward trend in the 100th iteration. In other words, SLLABC algorithm not only preserves good exploration and exploitation abilities and can particularly compromise these two well to search the space during the evolution.

Table 3 shows the worst, the best, the mean and the standard deviation of the error rate obtained by SLLABC and other compared algorithms, and the best ones obtained on each dataset are bold. Subsequently, the Wilcoxon rank sum test (Wilcoxon 1992) is adopted to compare the results obtained by the SLLABC algorithm and other compared algorithms at a significance level of 0.05. The result is given in Table 4, wherein the symbol ‘ + ’denotes the SLLABC algorithm outperforms the compared algorithm significantly, while ‘-’—indicates otherwise. On some datasets, the compared algorithms have similar performance with SLLABC, which is marked as ‘= ’.

The experimental results in Table 3 show that VS_CCPSO algorithm achieves lower error rate than other algorithms on datasets with a small number of features such as LSVT and Yale, while SLLABC outperforms other algorithms on datasets with a large number of features. It achieves the minimum mean error rate, the minimum best and the minimum worst error rate on ten datasets, ten datasets and eight datasets, respectively. In addition, the SLLABC was able to perform more stably in comparison with the state-of-the-art algorithms, as smaller standard deviation values supported these findings.

The result of Wilcoxon rank sum test indicates that, the error rate of SLLABC algorithm is significantly lower than that of most algorithms on most datasets. Specifically, compared with VS_CCPSO algorithm, SLLABC underperforms VS_CCPSO on the datasets with a small number of features such as LSVT and Yale. Nevertheless, SLLABC is superior to or level pegging with VS_CCPSO on higher-dimensional datasets. The difference in error rate is insignificant between SLLABC and ALO_GWO in all datasets except DLBCL. In a word, SLLABC algorithm can achieve high classification accuracy for feature selection, especially on very high-dimension datasets.

Since the number of the selected features determines the computational cost of a classification algorithm, it is also a key performance index of feature selection. We compare the size of the feature subset of SLLABC algorithm with that of other algorithms. Table 5 shows the results of the maximum (worst), the mean, the minimum value (best) and the standard deviation (std) value of features selected by SLLABC and other algorithms. Inspecting the results, VS_CCPSO are outstanding on LSVT, Yale and SRBCT. Combined with Table 3, VS_CCPSO gets lower error rates and fewer features on these datasets, but the performance on very high-dimensional datasets is not as good as SLLABC. 2D_UPSO obtains the smallest feature subsets on higher-dimensional datasets, but its standard deviation in each dataset is large, which indicates that 2D_UPSO is

Table 7 Comparison of the execution time of SLLABC and other compared algorithms

Datasets	Index	CSO	2D_UPSO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC	SLLABC
LSVT	Worst	63.371	**44.877**	91.4	104.99	103.748	290.1387	55.89	180.771	173.237
	Mean±std	55.479±3.99	**42.871±1.45**	83.365±4.22	89.583±6.15	82.59±12.61	273.60±10.47	53.87±1.15	172.112±4.14	161.725±8.76
	best	51.559	**40.934**	77.4	84.482	65.106	260.409	52.68	164.527	146.876
Yale	Worst	**149.053**	281.945	512.424	27731.33	259.051	358.774	181.9049	1156.99	327.613
	Mean±std	**147.222±1.70**	261.572±11.93	482.005±22.63	3046.772±8673.28	190.899±35.52	319.500±21.67	166.880±9.41	547.563±233.72	323.785±2.31
	best	**144.516**	243.148	454.417	270.011	138.69	303.378	153.1691	402.912	319.685
Colon	Worst	**56.685**	64.409	164.709	307.25	78.255	373.564	86.7477	272.698	185.095
	Mean±std	**55.833±0.64**	62.7±1.22	153.435±7.04	287.624±15.66	58.746±17.07	346.607±28.32	85.4007±0.6720	271.671±0.66	173.467±6.43
	best	**54.855**	60.785	140.173	251.879	22.212	300.571	84.6242	270.248	165.401
SRBCT	Worst	**65.577**	254.26	343.204	327.805	156.609	393.483	138.48	409.926	286.936
	Mean±std	**64.838±0.42**	240.938±10.01	274.998±26.58	310.947±14.73	109.007±29.12	387.267±5.937	129.49±3.48	381.792±10.78	276.089±7.26
	best	**63.989**	225.862	252.63	274.578	64.867	374.012	125.69	374.011	269.191
DBWorld	Worst	**281.358**	1159.546	464.986	3233.149	304.799	494.64	406.8648	930.697	435.142
	Mean±std	268.705±10.77	638.785±196.96	419.94±23.72	840.963±842.08	**117.015±81.38**	457.78±26.15	388.54±13.73	901.812±15.02	406.466±17.62
	best	249.771	491.703	400.598	493.194	**39.214**	427.11	362.0077	880.246	381.876
Leukemia1	Worst	431.972	3527.4	583.3	852.191	833.202	646.2008	718.59	1268.146	**366.042**
	Mean±std	417.733±10.54	1406.175±1019.12	548.949±23.25	774.579±51.32	466.616±246.93	604.89±15.80	589.83±70.64	1257.020±9.37	**351.099±9.81**
	best	395.618	810.926	510.37	661.116	**189.53**	588.5002	537.25	1243.446	335.278
DLBCL	Worst	**445.163**	5717.368	721.665	5198.543	1334.396	651.1005	949.5288	2463.556	660.04
	Mean±std	**436.211±4.25**	1518.828±1477.24	631.813±50.29	1259.770±1387.91	709.665±354.33	644.1225±3.4531	911.57±22.32	2409.589±23.93	596.838±41.63
	best	428.788	908.618	572.629	709.577	**248.444**	640.0396	883.5772	2385.534	534.197
ALLAML	Worst	662.023	870.531	898.862	949.384	1103.361	3269.92	1647.53	3013.38	**647.866**
	Mean±std	646.212±10.86	800.979±31.18	827.998±55.03	780.630±73.08	**468.714±341.35**	1012.09±794.42	1548.87±62.91	2954.409±41.94	629.844±11.60
	best	629.11	765.574	744.722	692.513	**179.502**	724.69	1437.23	2895.124	614.5
Pixraw10P	Worst	1694.86	30984.457	3472.743	3131.419	3728.951	**1025.376**	1995.26	4192.29	1030.314
	Mean±std	1657.939±35.83	6075.577±8752.85	3417.893±27.89	1428.083±600.28	2855.928±811.59	1010.468±7.3863	1733.92±165.313	4187.257±4.78	**983.82±35.51**
	best	1611.872	3134.863	3374	1154.445	952.366	997.993	1598.76	4177.647	**922.686**
Prostate	Worst	2017.032	31523.541	2709.83	3599.993	4961.877	1924.05	3865.52	8418.34	**1899.147**
	Mean±std	1929.810±33.30	6722.332±8734.11	2261.695±204.57	2435.872±740.75	2024.488±1510.35	1919.96±41.1396	2990.89±595.33	8370.281±33.96	**1849.568±32.75**
	best	1900.973	3556.431	2007.704	1589.21	**1409.162**	1782.46	2377.18	8328.043	1801.274
Leukemia2	Worst	877.073	33301.351	2107.527	7818.939	2569.846	1008.52	1242.01	2738.697	**769.983**
	Mean±std	852.612±10.87	5670.962±9735.36	1820.362±157.23	2342.933±2004.79	1051.291±590.92	997.62±8.81	1183.29±56.87	2722.613±8.012	**714.082±32.10**
	best	838.053	2290.118	1636.275	1393.611	**542.45**	986.23	1102.25	2710.739	666.025
GLI_85	Worst	4081.994	4962.481	4628.113	3752.538	6563.272	**1645.678**	2964.19	9283.474	2669.54
	Mean±std	3996.313±34.94	4846.751±76.51	3872.890±562.97	3013.793±494.32	3689.243±1706.51	**1422.544±104.0785**	2950.18±9.504	9229.960±44.18	2507.958±118.14
	best	3965.587	4759.073	2946.259	2165.611	**1203.425**	1363.649	2934.81	9142.98	2320.219

Table 8 Wilcoxon rank sum test on execution time of SLLABC and the compared algorithms

Datasets	CSO	2D_USPO	VS_CCPSO	ALO_GWO	MbGWO-SFS	HLBDA	FRGA	AC_ABC
LSVT	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)
Yale	0.00018(−)	0.00018(−)	0.00018(+)	0.00460(−)	0.00018(−)	0.14047(=)	0.00018(−)	0.00018(−)
Colon	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(+)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)
SRBCT	0.00018(−)	0.00018(−)	0.21230(=)	0.00058(+)	0.00018(−)	0.00018(−)	0.00018(−)	0.00018(−)
DBWorld	0.00018(−)	0.00018(+)	0.18590(=)	0.00018(+)	0.00018(−)	0.00077(+)	0.03764(−)	0.00018(−)
Leukemia1	0.00018(+)	0.00018(+)	0.00018(+)	0.00018(+)	0.62320(=)	0.00018(+)	0.00018(+)	0.62320(=)
DLBCL	0.00018(−)	0.00018(+)	0.16200(=)	0.00018(+)	0.51480(=)	0.02113(+)	0.00018(+)	0.51480(=)
ALLAML	0.01130(+)	0.00018(+)	0.00018(+)	0.00018(+)	0.14050(=)	0.00018(+)	0.00018(+)	0.14050(=)
Pixraw10P	0.00018(+)	0.00018(+)	0.00018(+)	0.00018(+)	0.00310(+)	0.12122(=)	0.00018(+)	0.00310(+)
Prostate	0.00018(+)	0.00018(+)	0.00018(+)	0.10410(=)	0.14470(=)	0.24132(=)	0.00018(+)	0.14470(=)
Leukemia2	0.00018(+)	0.00018(+)	0.00018(+)	0.00018(+)	0.10410(=)	0.00018(+)	0.00066(+)	0.10410(=)
GLI_85	0.00018(+)	0.00018(+)	0.00018(+)	0.01400(+)	0.14050(=)	0.00018(−)	0.00018(+)	0.14050(=)

‘ ± ’: The result obtained by SLLABC is significantly better/worse than the compared algorithm. ‘ = ’: There is no statistically significant difference e between the results obtained by SLLABC and the compared algorithm

unstable on reducing feature size. SLLABC algorithm does not obtain the smallest feature subset on each data set, however, it reserves fewer features on most data sets, and its standard deviation value is not large. On the whole, SLLABC has a good performance on dimensionality reduction for very high dimensional feature selection.

Table 6 shows the results of Wilcoxon rank sum test on feature subset size. The effect of dimensionality reduction by SLLABC algorithm is substantially better than other algorithms on most datasets. However, when a dataset only contains the key features that must be used by classifiers, removing any one of them may increase the error rate. Although the subset size of 2D_USPO, MbGWO-SFS and HLBDA can be reduced, their classification accuracy is not improved. In contrast, the features selected by SLLABC can usually provide key information, which is good at enhancing the classification performance. The summarized results concluded that the proposed SLLABC was more capable of selecting significant features compared to the state-of-the-art algorithms in high-dimensional datasets.

For a better evaluation of the proposed SLLABC algorithms, not only the accuracy and the size of feature subsets but also the computational complexity needs to be investigated. To provide a more intuitive representation of the time consumption, this paper calculates the CPU execution times of SLLABC and other algorithms under the same physical conditions. Table 7 summarizes the worst(-longest), the mean and the best(shortest) CPU execution time (in seconds) between SLLABC and other compared algorithms. Table 8 shows the result of the Wilcoxon rank sum test on execution time.

As can be seen from Table 7, SLLABC algorithm still performs well on very high-dimensional datasets, which takes less execution time, but is inferior to CSO on the data sets with a small number of features. However, what is interesting is that as the dimension of datasets increase, the advantage of the SLLABC in time consumption is becoming more and more obvious. Combining Tables 7, 8, SLLABC algorithm consumes less time than most algorithms and obtains higher accuracy and fewer features on the very high-dimensional datasets. Therefore, our proposed SLLABC algorithm is a valuable feature selection tool, and it can be implemented for other real-world applications.

6 Conclusions

This paper aims to propose a Self-adaptive Level-based Learning Artificial Bee Colony (SLLABC) algorithm to deal with the feature selection problem on high-dimensional classification.

First of all, we described a novel level-based learning (LL) mechanism for ABC algorithm in detail. In basic ABC algorithm, the current individual learns from an individual selected randomly from the whole population, while after introducing the novel level-based learning mechanism, it has to learn from a better individual, and the individuals in the first level have to learn from each other. This mechanism enhances the exploitation of ABC algorithm and makes algorithm obtain the optimal solution more quickly.

Secondly, a self-adaptive method was proposed to determine the number of levels. Compared with the dynamic method for the number of levels, our new method adaptively adjusts the level number according to the average diversity of the population instead of artificial empirical values. The experimental results show that the self-adaptive method improves the exploitation ability and dimensionality reduction of ABC algorithm.

Furthermore, to improve the performance of ABC algorithm, we proposed a new update mechanism. If a candidate individual has the same accuracy as the current individual, but its number of selected features is the same as or less than the current individual, then replaces the current individual to enter the next iteration. This strategy effectively not only reduces the number of selected features but also improves the update frequency of individuals to enhance the exploration ability of ABC algorithm.

Finally, the proposed algorithm SLLABC is compared with SLLABC-1, SLLABC-2, SLLABC-3, and the results show that SLLABC can effectively balance the exploration and exploitation during the evolution. We further compared SLLABC with eight state-of-the-art algorithms on classification error rate, size of subset, and execute time. The results corroborate that proposed SLLABC is indeed a competitive algorithm to feature selection problems, especially with high-dimension data.

Moreover, it is worth mentioning that the novel level-based learning mechanism and the new update mechanism proposed in our paper are universal to NP-hard problems. Like most EAs, however, SLLABC has high computational complexity due to the characteristics of random search and repeated evaluations. Therefore, how to reduce execution time by using sample reduction strategies or parallelization is one of the directions for future study. Additionally, minimizing the size of the feature subsets and maximizing the classification accuracy are both important indicators in feature selection, hence formulating feature selection as a multi-objective combinatory optimization problem to meet various requirements of decision-makers is a direction for our future study.

Acknowledgements This work was supported by Special Funds of Science and Technology Innovation Project of Fujian Agriculture and Forestry University (KFA17030A, KFA17181A), Natural Science Foundation of Fujian Province (2017J01607, 2018J01612), Forestry Science and Technology Projects in Fujian Province (Memorandums 26), Fund for Excellent Dissertation in Fujian Agriculture and Forestry University (1122YS01004), China.

Funding The authors have not disclosed any funding.

Data availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest Authors declare that they have no conflict of interest, financially or otherwise.

Ethical approval This article does not contain any studies with the participants of humans or animals.

References

Babaoglu I (2015) Artificial bee colony algorithm with distribution-based update rule. Appl Soft Comput 34:851–861

Canuto AM, Nascimento DS (2012) A genetic-based approach to features selection for ensembles using a hybrid and adaptive fitness function. In: The 2012 international joint conference on neural networks (IJCNN), pp 1–8. IEEE

Cheng R, Jin Y (2015) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204

Cui L, Li G, Luo Y, Chen F, Ming Z, Lu N, Lu J (2018) An enhanced artificial bee colony algorithm with dual-population framework. Swarm Evol Comput 43:184–206

Demirekler M, Haydar A (1999) Feature selection using genetics-based algorithm and its application to speaker identification. In: 1999 IEEE international conference on acoustics, speech, and signal processing. proceedings. ICASSP99 (Cat. No. 99CH36258), pp 329–332. IEEE

Derrac J, García S, Herrera F (2009) A first study on the use of coevolutionary algorithms for instance and feature selection. In: International conference on hybrid artificial intelligence systems. Springer, pp 557–564

Ding Y, Zhou K, Bi W (2020) Feature selection based on hybridization of genetic algorithm and competitive swarm optimizer. Soft Comput 24(15):11663–11672. https://doi.org/10.1007/s00500-019-04628-6

Duan H-b, Xu C-f, Xing Z-H (2010) A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems. Int J Neural Syst 20(01):39–50

El-Kenawy E-SM, Eid MM, Saber M, Ibrahim A (2020) MbGWO-SFS: modified binary grey wolf optimizer based on stochastic fractal search for feature selection. IEEE Access 8:107635–107649

Gao W-f, Liu S-y, Jiang F (2011) An improved artificial bee colony algorithm for directing orbits of chaotic systems. Appl Math Comput 218(8):3868–3879

Ghamisi P, Couceiro MS, Benediktsson JA (2014) A novel feature selection approach based on FODPSO and SVM. IEEE Trans Geosci Remote Sens 53(5):2935–2947

Gheyas IA, Smith LS (2010) Feature subset selection in large dimensionality domains. Pattern Recognit 43(1):5–13

Grande J, del Rosario Suárez M, Villar JR (2007) A feature selection method using a fuzzy mutual information measure. In: Innovations in hybrid intelligent systems. Springer, pp 56–63

Gu S, Cheng R, Jin Y (2018) Feature selection for high-dimensional classification using a competitive swarm optimizer. Soft Comput 22(3):811–822

Hafiz F, Swain A, Patel N, Naik C (2018) A two-dimensional (2-D) learning framework for particle swarm based feature selection. Pattern Recognit 76:416–433

Hamamoto AH, Carvalho LF, Proenca ML (2015) ACO and GA metaheuristics for anomaly detection. In: 2015 34th international conference of the Chilean Computer Science Society (SCCC), pp 1–6. IEEE

Hancer E, Xue B, Karaboga D, Zhang M (2015) A binary ABC algorithm based on advanced similarity scheme for feature selection. Appl Soft Comput 36:334–348

Harfouchi F, Habbi H (2015) A cooperative learning strategy with multiple search mechanisms for improved artificial bee colony optimization. In: 2015 15th International conference on intelligent systems design and applications (ISDA), pp 434–439. IEEE

Jeong Y-S, Shin KS, Jeong MK (2015) An evolutionary algorithm with the partial sequential forward floating search mutation for large-scale feature selection problems. J Oper Res Soc 66(4):529–538

Karaboga D, Gorkemli B (2014) A quick artificial bee colony (qABC) algorithm and its performance on optimization problems. Appl Soft Comput 23:227–238

Karaboga D (2005) An idea based on honey bee swarm for numerical optimization vol 200. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department

Liao Y, Vemuri VR (2002) Use of K-Nearest Neighbor classifier for intrusion detection. Comput Secur 21(5):439–448. https://doi.org/10.1016/S0167-4048(02)00514-X

Mafarja M, Jarrar R, Ahmad S, Abusnaina AA (2018) Feature selection using binary particle swarm optimization with time varying inertia weight strategies. In: Proceedings of the 2nd international conference on future networks and distributed systems. ACM, pp 1–9

Meenachi L, Ramakrishnan S (2020) Differential evolution and ACO based global optimal feature selection with fuzzy rough set for cancer data classification. Soft Comput. https://doi.org/10.1007/s00500-020-05070-9

Menghour K, Souici-Meslati L (2016) Hybrid ACO-PSO based approaches for feature selection. Int J Intell Eng Syst 9(3):65–79

Mohammadi FG, Abadeh MS (2014) Image steganalysis using a bee colony based feature selection algorithm. Eng Appl Artif Intell 31:35–43

Nguyen BH, Xue B, Zhang M (2020) A survey on swarm intelligence approaches to feature selection in data mining. Swarm Evol Comput 54:100663

Oh I-S, Lee J-S, Moon B-R (2004) Hybrid genetic algorithms for feature selection. IEEE Trans Pattern Anal Mach Intell 26(11):1424–1437

Potter MA, Jong KAD (2000) Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evol Comput 8(1):1–29

Rakshit P, Bhattacharyya S, Konar A, Khasnobish A, Tibarewala D, Janarthanan R (2013) Artificial bee colony based feature selection for motor imagery EEG data. In: Proceedings of seventh international conference on bio-inspired computing: theories and applications (BIC-TA 2012). Springer, pp 127–138

Rao H et al (2019) Feature selection based on artificial bee colony and gradient boosting decision tree. Appl Soft Comput 74:634–642

Shi Y-j, Teng H-f, Li Z-q (2005) Cooperative co-evolutionary differential evolution for function optimization. In: International conference on natural computation. Springer, pp 1080–1088

Shunmugapriya P, Kanmani S (2017) A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid). Swarm Evol Comput 36:27–36

Siedlecki W, Sklansky J (1993) A note on genetic algorithms for large-scale feature selection. In: Handbook of pattern recognition and computer vision. World Scientific, pp 88–107

Song X-f, Zhang Y, Guo Y-n, Sun X-y, Wang Y-l (2020) Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data. IEEE Trans Evol Comput 24(5):882–895. https://doi.org/10.1109/TEVC.2020.2968743

Sousa P, Cortez P, Vaz R, Rocha M, Rio M (2013) Email spam detection: a symbiotic feature selection approach fostered by evolutionary computation. Int J Inf Technol Decis Mak 12(04):863–884

Too J, Abdullah AR (2021) A new and fast rival genetic algorithm for feature selection. J Supercomput 77(3):2844–2874. https://doi.org/10.1007/s11227-020-03378-9

Too J, Mirjalili S (2021) A hyper learning binary dragonfly algorithm for feature selection: a COVID-19 case study. Knowl Based Syst 212:106553. https://doi.org/10.1016/j.knosys.2020.106553

Tran B, Xue B, Zhang M (2018) Variable-length particle swarm optimization for feature selection on high-dimensional classification. IEEE Trans Evol Comput 23(3):473–487

Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239

Vieira SM, Sousa JM, Runkler TA (2010) Two cooperative ant colonies for feature selection using fuzzy models. Expert Syst Appl 37(4):2714–2723

Wang H, He C, Li Z (2020) A new ensemble feature selection approach based on genetic algorithm. Soft Comput 24(20):15811–15820. https://doi.org/10.1007/s00500-020-04911-x

Wilcoxon F (1992) Individual comparisons by ranking methods. In: Breakthroughs in statistics. Springer, pp 196–202

Xue B, Fu W, Zhang M (2014a) Differential evolution (DE) for multi-objective feature selection in classification. In: Proceedings of the companion publication of the 2014a annual conference on genetic and evolutionary computation, pp 83–84

Xue B, Zhang M, Browne WN (2014b) Particle swarm optimisation for feature selection in classification: novel initialisation and updating mechanisms. Appl Soft Comput 18:261–276

Xue B, Zhang M, Browne WN, Yao X (2016) A survey on evolutionary computation approaches to feature selection. IEEE Trans Evol Comput 20(4):606–626. https://doi.org/10.1109/TEVC.2015.2504420

Yang Q, Chen W-N, Da Deng J, Li Y, Gu T, Zhang J (2017) A level-based learning swarm optimizer for large-scale optimization. IEEE Trans Evol Comput 22(4):578–594

Zawbaa HM, Emary E, Grosan C, Snasel V (2018) Large-dimensionality small-instance set feature selection: a hybrid bio-inspired heuristic approach. Swarm Evol Comput 42:29–42

Zhang Y, Gong D, Hu Y, Zhang W (2015) Feature selection algorithm based on bare bones particle swarm optimization. Neurocomputing 148:150–157

Zhang Y, Gong D-w, Gao X-z, Tian T, Sun X-y (2020) Binary differential evolution with self-learning for multi-objective feature selection. Inf Sci 507:67–85

Zhou X, Wu Z, Deng C, Peng H (2015) Enhancing artificial bee colony algorithm with generalised opposition-based learning. Int J Comput Sci Math 6(3):297–309

Zorarpacı E, Özel SA (2016) A hybrid approach of differential evolution and artificial bee colony for feature selection. Expert Syst Appl 62:91–103