Bi-Directional Feature Fixation-Based Particle Swarm Optimization for Large-Scale Feature Selection

Bi-Directional Feature Fixation-Based Particle Swarm Optimization for Large-Scale Feature Selection

Jia-Quan Yang, *Student Member, IEEE*, Qi-Te Yang, *Student Member, IEEE*, Ke-Jing Du, Chun-Hua Chen, *Member, IEEE*, Hua Wang, *Senior Member, IEEE*, Sang-Woon Jeon, *Member, IEEE*, Jun Zhang, *Fellow, IEEE*, and Zhi-Hui Zhan, *Senior Member, IEEE*

Abstract—Feature selection, which aims to improve the classification accuracy and reduce the size of the selected feature subset, is an important but challenging optimization problem in data mining. Particle swarm optimization (PSO) has shown promising performance in tackling feature selection problems, but still faces challenges in dealing with large-scale feature selection in Big Data environment because of the large search space. Hence, this article proposes a bi-directional feature fixation (BDFF) framework for PSO and provides a novel idea to reduce the search space in large-scale feature selection. BDFF uses two opposite search directions to guide particles to adequately search for feature subsets with different sizes. Based on the two different search directions, BDFF can fix the selection states of some features and then focus on the others when updating particles, thus narrowing the large search space. Besides, a self-adaptive strategy is designed to help the swarm concentrate on a more promising direction for search in different stages of evolution and achieve a balance between exploration and exploitation. Experimental results on 12 widely-used public datasets show that BDFF can improve the performance of PSO on large-scale feature selection and obtain smaller feature subsets with higher classification accuracy.

Index Terms—Bi-directional feature fixation (BDFF), evolutionary computation, feature selection, large-scale, particle swarm optimization (PSO)

1 Introduction

Feature selection is the process of selecting a portion of features relevant to the labels in classification problems and removing redundant or noisy features from the entire feature set. The goal of feature selection is to select as few features as possible while maximizing the discriminative capability of the selected features. As feature selection can reduce the data dimensionality and the difficulty of analyzing and solving problems, it has become an effective data preprocessing method in many fields [1], [2].

Feature selection is essentially a binary discrete optimization problem and has been proved to be NP-hard [3], [4]. Most existing feature selection methods can be roughly classified into three categories: filter methods, wrapper methods, and embedded methods [5], [6]. Filter methods analyze the features through some statistical or informatics methods and then select the features with high scores. Wrapper methods search for the optimal feature subset by evaluating all candidate subsets on a specific problem. Embedded methods usually embed feature selection into the training process of machine learning through regularization [7], [8]. In general, wrapper methods perform better than filter methods in classification accuracy and have a larger scope of application than embedded methods [9]. Evolutionary computation (EC) techniques are powerful in solving variants of NP-hard optimization problems [10], [11], [12], [13] and they have been widely applied as wrapper methods for feature selection due to their excellent global search ability [14]. Particle swarm optimization (PSO) is a representative EC technique first proposed by Kennedy and Eberhart [15]. Compared with other EC methods, PSO has the advantage of simple implementation and fast convergence [16], thus becoming an effective method for feature selection [9], [14]. Therefore, this paper focuses on PSO-based methods to solve feature selection problems.

- *Jia-Quan Yang, Qi-Te Yang, and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong 510006, China. E-mail: {jeremy.j.yang, qiteyang}@foxmail.com, zhanapollo@163.com.*
- *Ke-Jing Du and Hua Wang are with the Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC 8001, Australia. E-mail: dukejing@126.com, hua.wang@vu.edu.au.*
- *Chun-Hua Chen is with the School of Software Engineering, South China University of Technology, Guangzhou, Guangdong 510006, China. E-mail: chunhuachen@scut.edu.cn.*
- *Sang-Woon Jeon is with the Department of Electronics and Communication Engineering, Hanyang University, Ansan 15588, South Korea. E-mail: sangwoonjeon@hanyang.ac.kr.*
- *Jun Zhang is with the Zhejiang Normal University, Jinhua, Zhejiang 321004, China, and also with the Hanyang University, Ansan 15588, South Korea. E-mail: junzhang@ieee.org.*

Manuscript received 20 September 2022; revised 15 November 2022; accepted 19 December 2022. Date of publication 29 December 2022; date of current version 13 May 2023.
This work was supported in part by the National Natural Science Foundations of China under Grant 62176094, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, in part by the National Research Foundation of Korea under Grant NRF- 2022H1D3A2A01093478, and in part by the Korea Institute of Ocean Science and Technology under Grant PE99732.
(Co-First Authors: Jia-Quan Yang and Qi-Te Yang)
(Corresponding authors: Chun-Hua Chen and Zhi-Hui Zhan.)
Digital Object Identifier no. 10.1109/TBDATA.2022.3232761

Due to the development of the Big Data era, data from various application fields also grows with the increment of the number of features. In general, the potential search space for feature selection with D features is 2^D and it grows exponentially when the number of features increases. Under this circumstance, PSO methods face two challenges. Firstly, they require more computational resources for the evaluations of candidate feature subsets. Secondly, the search ability of these PSO methods drops sharply because of "the curse of dimensionality" [17], [18], [19], [20], [21].

Therefore, many PSO variants have been proposed for large-scale feature selection recently. They can be roughly divided into two categories. In the first category, PSO-based algorithms aim to design a more effective and reasonable evolutionary mechanism for particles to improve their performance on large-scale feature selection problems [22][23][24]. In the second category, PSO-based algorithms use some correlation measures to indicate the importance of each feature and focus on searching for solutions formed by features with more importance in the subsequent search process [25][26][27][28][29][30]. These correlation-based algorithms usually perform better because they can narrow the search space according to the correlation measures. However, they still face the following three challenges. Firstly, the implementation of these algorithms is more complicated than those in the first category because of additional correlation measurement methods or control mechanisms based on correlation. Secondly, the correlation information of features is sometimes difficult to obtain. For example, the cost of correlation computing is expensive in high-dimensional data, and the commonly used entropy-based correlation measures are difficult to estimate accurately on continuously distributed observations [31]. Thirdly, the correlation analysis is usually incomplete because it is time-consuming to calculate correlation values of all feature combinations and it is difficult to calculate a proper correlation value of multiple variables. Usually, only correlations between two single variables, e.g., correlations between two single features or correlations between a single feature and the label, are considered in the correlation analysis. In other words, the correlation information of feature subsets composed of multiple features is always missed, which affects the correctness of analyzing feature importance. Thus, over-reliance on incomplete correlation analysis results may mislead particles and prevent them from finding optimal solutions.

Nevertheless, correlation-based algorithms are effective methods for large-scale feature selection by focusing on a portion of important features specifically rather than on all features equally, even though there are still some challenges as mentioned above. To overcome the above challenges faced by traditional correlation-based algorithms, an algorithm that can narrow the search space but does not over-rely on correlation measures is greatly expected. Therefore, this paper proposes a novel framework named bi-directional feature fixation (BDFF) for PSO. The main novelties and contributions of BDFF can be summarized as follows.

(1) Using bi-direction guidance when updating particles to fully search for solutions with different numbers of selected features. Many existing feature selection algorithms consider the number of selected features in their fitness evaluation to find a solution with fewer features [26], [28], [32][33][34][35][36]. Different from those existing algorithms, BDFF does not need to consider the number of selected features in its fitness function, so it simplifies the fitness evaluation. Instead, it uses the information of the number of selected features to update particles and can also help particles find solutions with fewer features.

(2) Narrowing the search space by the feature fixation strategy to improve search efficiency. Therefore, BDFF can not only reduce the search space but also reduce the difficulty of searching for a better solution.

(3) Proposing a self-adaptive direction change (SADC) strategy for particles so that they can change the search direction adaptively with the information provided by the swarm. With the SADC strategy, particles can switch the state of each feature between being fixed and not being fixed dynamically, which increases the flexibility of feature fixation and further improves their search ability.

(4) Having advantages in simple implementation and fine-grained control. BDFF is not complex to be implemented and can be easily applied to many existing PSO-based feature selection algorithms. Besides, BDFF has a small control granularity because it takes the neighborhood containing only several features as the basic unit for fixation.

The rest of this paper is organized as follows. In Section 2, the related work of applying PSO to feature selection is presented. Section 3 introduces the detailed implementation of the proposed BDFF for PSO. The experimental results and analysis of BDFF are given in Section 4. Finally, Section 5 concludes this paper.

2 Related Work

2.1 Feature Selection Problem

The feature selection problem is a binary discrete optimization problem, which means that the optional values for each dimension are "0" or "1". Assuming that there are D features in the data, the solution for the feature selection problem can be represented by a D-dimension vector $\boldsymbol{x}$. The binary value of "1" or "0" at the dth dimension x_d represents the dth feature is selected or not selected, respectively. Then, the goal of the feature selection optimization problem is to select a feature subset from the D features to maximize the discriminative capability $f(\boldsymbol{x})$ of the data, as shown in Eq. (1). For example, in classification problems, $f(\boldsymbol{x})$ can be the classification accuracy of the selected feature subset, and the goal of feature selection is to maximize $f(\boldsymbol{x})$.

$$\begin{aligned} \max \quad & f(\boldsymbol{x}) \\ \text{s.t.} \quad & \boldsymbol{x} = (x_1, x_2, \ldots, x_D) \\ & x_d \in \{0,1\}, \quad d = 1,2,\ldots,D \end{aligned} \tag{1}$$

2.2 PSO-Based Algorithms for Large-scale Feature Selection

Most existing PSO-based algorithms for feature selection can be divided into two categories: one of the categories tries to use different mechanisms to help particles search effectively, while the other utilizes the correlation information to gain further improvement.

The first category of algorithms focuses on designing different evolutionary mechanisms for feature selection to improve their performance. For example, the earliest PSO-based algorithm that can be used for feature selection is binary PSO (BPSO) proposed by Kennedy and Eberhart [37], which used the velocity of particles to represent the probability of a feature being selected. After BPSO, many PSO-based algorithms attempted to improve the encoding representation, the initialization strategy, the updating mechanism, and the evaluation function of PSO to obtain a better performance on feature selection and 0/1 problems. Xue et al. [38] proposed a PSO-based approach with novel initialization strategies and updating mechanisms for feature selection. Shen et al. [39] developed a bi-velocity discrete PSO (BVDPSO) using the two velocities of particles to respectively represent the possibilities of being 1 and 0. Gu et al. [23] discretized the competitive swarm optimizer (CSO) [40] and applied CSO to large-scale feature selection problems. Later, the potential PSO proposed by Tran et al. [41] used potential entropy-based cut-points to discretize values of each feature and then encoded the particles with those discrete values. In self-adaptive PSO introduced by Xue et al. [24], multiple candidate solution generation strategies were applied simultaneously by a self-adaptive mechanism to increase the diversity of the swarm.

In the first category, although many different mechanisms have been proposed to enhance the search ability of particles, few of them can distinguish which features are worth further searching. Therefore, they always treat all features equally and keep searching for solutions by considering the entire feature set, which wastes computational resources and makes it difficult to find a better solution.

The second category of PSO-based algorithms uses correlation measures as an auxiliary tool to figure out the weight of importance of each feature, which shows more potential for large-scale feature selection and has attracted great attention in recent years. Commonly used correlation measures are based on similarity or information theory [31], [42], including Relief-F [43], mutual information (MI) [44], symmetric uncertainty (SU) [45], etc. Chuang et al. [46] first used the correlation-based feature selection as a filter method to select the important features and then used the BPSO with chaotic theory to search for the final optimal solution on those important features. After sorting features with SU measure, Tran et al. [25] designed the variable-length PSO with local search (VLPSO-LS) to dynamically shorten the length of particles thus narrowing the search space. Chen et al. [26] introduced the evolutionary multitasking framework into PSO, which identified a promising feature subset with high Relief-F values and generated two related tasks on the promising feature subset and the whole feature set, respectively. Song et al. [27] proposed the variable-size cooperative coevolutionary PSO, which employed a space division strategy based on SU measure and allocated a larger subswarm for those features that were more relevant to the label. Besides, the PSO variant developed by Chen et al. [28] generated new particles with a correlation-guided updating strategy and those features with higher correlation were more likely to be selected. To reduce the computational cost, Song et al. [29] proposed a hybrid feature selection algorithm named HFS-C-P that used SU measure to discard low-correlation features and to cluster features so that it could search in a small solution space. In [47], the SU measure was also used to distinguish relevant and redundant features in the local search strategy and to affect the mutation probability in the adaptive flip mutation strategy.

In the second category, the correlation-based PSO algorithms can narrow the search space because they can point out promising features and then focus on those promising features rather than all features. However, they still face the following challenges. First, the calculation of correlations is time-consuming. To obtain the correlations between D features and the label, the time complexity is O(D). If the correlations among features are also required, the time complexity will rise dramatically to O(D^2), which does matter in large-scale feature selection problems in Big Data environments. Second, due to the high computational consumption, correlations between groups of multiple features are rarely considered, so the correlation information used to assist PSO is incomplete in most cases. Third, the entropy-based correlation measures such as MI and SU can only be applied to datasets with continuous numerical features by discretization [31], [42] or non-parametric estimation methods [48], which increases the difficulty of their application and weakens the stability of their performance on different data.

2.3 Bare Bones PSO

Bare bones PSO (BBPSO) proposed by Kennedy [49] is a simple but efficient PSO variant. It drops the velocity part in the standard PSO and only uses the historical optimal position found by each particle and the global optimal position found by the swarm so far to update the position of each particle, as shown in:

$$x_{i,d} = \begin{cases} N\left(\frac{pbest_{i,d}+gbest_d}{2}, \left|pbest_{i,d} - gbest_d\right|\right), & \text{if } r < 0.5 \\ pbest_{i,d}, & \text{otherwise} \end{cases} \tag{2}$$

where $x_{i,d}$ is the dth dimension of the position of the ith particle, $pbest_{i,d}$ is the dth dimension of the historical optimal position found by the ith particle, $gbest_d$ is the dth dimension of the global optimal position found by the swarm so far, $N\ (\mu, \sigma)$ is a Gaussian distribution with a mean μ and a variance σ, and r is a random value uniformly sampled within [0, 1].

To help particles escape from the local attractor and perform better in feature selection, Qiu [22] introduced an adaptive chaotic jump (ACJ) strategy into BBPSO and then proposed BBPSO-ACJ. Chaos is a non-linear system and unpredictable. When BBPSO-ACJ is combined with a chaotic system, it can help the stagnated particles change their positions greatly and jump out of the trapped local optima. Therefore, swarm diversity can be promoted and the global search ability can be greatly enhanced. The position of each particle in BBPSO-ACJ is updated by:

$$x_{i,d} = \begin{cases} N\left(\frac{pbest_{i,d}+gbest_d}{2}, \left|pbest_{i,d} - gbest_d\right|\right), & \text{if } r > P_{cj,i} \\ pbest_{i,d}(1 + (2z_k - 1)), & \text{otherwise} \end{cases} \tag{3}$$

where $P_{cj,i}$ is the probability of the ith particle performing the chaotic jump, and $z_k \in (0, 1)$ is a value of a chaotic sequence generated with Eq. (4) every time it is used.

$$z_k = \begin{cases} 0.13, & k = 0 \\ 4z_{k-1}(1 - z_{k-1}), & k \geq 1 \end{cases} \tag{4}$$

To balance the convergence speed and the diverse swarm, BBPSO-ACJ designs a strategy to make $P_{cj,i}$ related to the stagnant generations s_i, i.e., generations without fitness improvement of the ith particle. Therefore, $P_{cj,i}$ can be calculated by:

$$P_{cj,i} = \frac{1}{1 + e^{2-s_i}} \tag{5}$$

Moreover, BBPSO-ACJ employs a method to decode the position of a particle into the representation of the selected feature subset. If the value of $x_{i,d}$ is greater than 0.5, then the dth feature will be selected into the feature subset; otherwise, the dth feature will be discarded.

3 Bi-Directional Feature Fixation Framework

In this section, the proposed BDFF for PSO to solve large-scale feature selection problems is introduced. First, the main idea of BDFF is given to illustrate how BDFF works with different search directions. After introducing the search direction initialization and the feature neighborhood representation, the details about feature fixation and the self-adaptive direction change strategy are discussed. Finally, the overall framework of BDFF is presented.

3.1 Main Idea of Design

A common initialization method for position $\mathbf{x}$ of a particle in PSO-based feature selection algorithms is to randomly select some features, as shown in:

$$x_{i,d} = \begin{cases} 1, & \text{if} \quad r > 0.5 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $x_{i,d}$ is the dth dimension of the ith particle. Since each feature has an equal probability of being selected and being not selected, the particles in the initial swarm have a mathematical expectation of the feature number which is equal to one-half of the total feature size. Focusing on the fact that the number of selected features contained in the global optimal solution can be greater or less than the mathematical expectation of the number of selected features contained in the initial particles, the BDFF framework is hence proposed.

The main idea of BDFF is to guide some particles to search for solutions with more features and the other particles to search for solutions with fewer features after initialization. As the particle swarm evolves and acquires new information, BDFF then changes the search direction of some particles adaptively to help the swarm approaches the global optimal solution in terms of the feature number as well as the fitness value.

Without loss of generality, BDFF also adopts Eq. (6) for particle initialization. Assuming that there is a dataset with D features for selection, and the global optimal solution contains fewer than $D/2$ selected features. Herein we give an example of the evolution process guided by BDFF, as shown in Fig. 1. After initialization, all particles in the swarm S select around

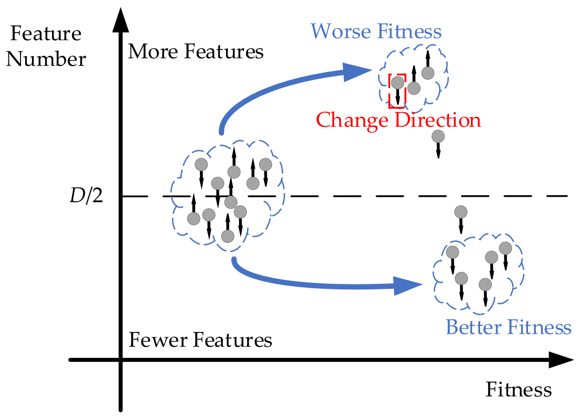


Fig. 1. Evolution process of the particles guided by BDFF.

$D/2$ features, and their fitness values are poor. Then, the particles are divided evenly into two subswarms with different initial search directions. One subswarm SS_u (composed of particles with upward arrows in Fig. 1) searches in the solution space with more than $D/2$ features. The other subswarm SS_l (composed of particles with downward arrows in Fig. 1) searches in the solution space with fewer than $D/2$ features. Each particle can judge whether it is in the correct direction, i.e., the direction that is more likely to guide it to find the global optimal solution, according to the information from the whole swarm. For example, in Fig. 1, the particles in SS_u with upward arrows are probably in the wrong direction, because their fitness values are worse than those of the particles searching in the opposite direction (i.e., particles with downward arrows). Therefore, it is necessary to adjust the search direction of some particles in SS_u in time to avoid a useless search, while the rest particles in SS_u are still reserved for exploration. Eventually, most of the particles gather together in one of the subswarms with similar fitness values and feature numbers to the global optimal solution.

In a feature selection problem, an optimal solution means that it should contain as fewer features as possible under the premise of optimal fitness (i.e., discriminative capability like Eq. (1)). To approach the optimal solution, most of the existing EC-based feature selection algorithms consider reducing the number of selected features and optimizing fitness at the same time in their evaluation. However, reducing the number of selected features is not the core goal of feature selection. Therefore, these EC-based algorithms may be misled by such consideration to focus on finding a feature subset with fewer features and ignore the importance of discriminative capability. Differently, the proposed BDFF framework adopts a novel technique named feature fixation to approach the optimal solution. With feature fixation, BDFF only needs to consider optimizing discriminative capability in its evaluation, and the size of the selected feature subset can be optimized automatically in its update process. For example, if the classification accuracy is adopted as the fitness function in the classification problem, then the only goal of BDFF is to maximize the classification accuracy.

3.2 Initialization of Search Direction

There are two search directions for particles: the direction of searching for solutions with more features and the direction

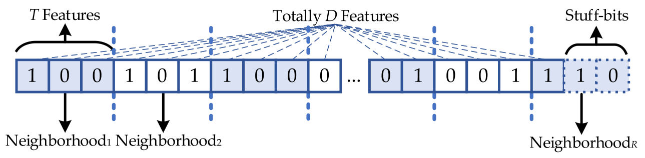


Fig. 2. An example of division for feature neighborhoods.

of searching for solutions with fewer features. Supposing that there are N particles in the swarm, the search direction of the ith particle p_i is initialized by:

$$dir(p_i) = \begin{cases} dir_l, & \text{if } i \leq \frac{N}{2} \\ dir_u, & \text{otherwise} \end{cases}, i = 1, 2, \ldots, N \tag{7}$$

where $dir(p_i)$ returns the search direction of p_i, and dir_l and dir_u are search directions representing that p_i searches for solutions with fewer features and searches for solutions with more features, respectively. After initialization, particles with direction dir_l are in subswarm SS_l, and the other particles with direction dir_u are in subswarm SS_u.

3.3 Representation of Feature Neighborhood

To ensure the subsequent feature fixation can be carried out, we introduce the representation of feature neighborhood here, which has two purposes. First, feature neighborhood can provide common information among adjacent features, and considering the features in the same neighborhood as a whole can make the search more efficient. Second, the use of feature neighborhood can gain a fine-grained control for the feature fixation and avoid fixing too many features at a time.

Assuming that there are D features in the data, each feature neighborhood consists of T adjacent features, then all features can be divided into R feature neighborhoods and R is calculated by:

$$R = \left\lceil \frac{D}{T} \right\rceil \tag{8}$$

where symbol $\lceil \cdot \rceil$ represents the ceiling function. If the number of features in the last neighborhood is less than T, extra features with random selection states will be added to the neighborhood as the stuff-bits. An example of neighborhood division with $T = 3$ is shown in Fig. 2, where value 1 indicates that the feature is selected, and value 0 indicates that the feature is not selected.

To strengthen the correlation between two features in the same neighborhood, SU values between features and the label of the dataset are used to sort all features before the neighborhood division. After being sorted by SU, features in the same neighborhood have a similar correlation for labels, thus they are more integrated and can be regarded as a whole in the feature fixation stage. The SU value between a feature F and the class label C can be calculated as:

$$\mathrm{SU}(F, C) = 2\frac{\mathrm{H}(F) - \mathrm{H}(F|C)}{\mathrm{H}(F) + \mathrm{H}(C)} \tag{9}$$

where H(F) and H(C) are the entropies of F and C, and H($F \mid C$) is the conditional entropy of F when C is given. Notice that SU is only used for the feature sorting at the beginning of BDFF, but does not play a dominant role in feature fixation. More importantly, BDFF still works well without the assistance of feature sorting by SU, which can be verified in Section 4.6.

Algorithm 1. Particle Position Update with Feature Fixation

```
Input: The ith particle $p_i$ to be updated, the total number of features $D$, the number of features $T$ in feature neighborhood, the historical optimal position $\boldsymbol{pbest}_i$ of $p_i$, the position $\boldsymbol{x}_i$ of $p_i$
Output: The updated particle $p_i$
BEGIN
 1: FOR $j = 1$ to $D$ DO
 2:   Calculate the neighborhood index $k = \lfloor j \,/\, T \rfloor + 1$;
 3:   IF $dir(p_i) == dir_l$ THEN
 4:     IF no feature in $neighborhood_k$ is selected in $\boldsymbol{pbest}_i$ THEN
 5:       $x_{i,j} \leftarrow pbest_{i,j}$;
 6:       CONTINUE;
 7:     END IF
 8:   ELSE
 9:     IF all features in $neighborhood_k$ are selected in $\boldsymbol{pbest}_i$ THEN
10:       $x_{i,j} \leftarrow pbest_{i,j}$;
11:       CONTINUE;
12:     END IF
13:   END IF
14:   Use any chosen position update mechanism to update $x_{i,j}$;
15: END FOR
16: RETURN $p_i$;
END
```

3.4 Feature Fixation Guided By Search Directions

Feature fixation, which is the core part of the BDFF framework, aims to fix some features and keep their selection states unchanged when the particle is being updated. The feature fixation process of each particle is guided by its current search direction and uses the feature neighborhood as the basic unit. When the condition of feature fixation meets, all features in the same neighborhood will be fixed as a whole until the search direction changes.

Supposing that there is a particle p_i in the subswarm SS_l, its condition of feature fixation can be described as follows. If none of the features in a feature neighborhood are selected in the historical optimal position of p_i (i.e., the $\boldsymbol{pbest}_i$), all these features in the neighborhood will be fixed as a whole. Then in the newly generated position $\boldsymbol{x}_i$ of p_i, the fixed features will be kept unselected and have the same selection states as what they have in the $\boldsymbol{pbest}_i$. As more feature neighborhoods are fixed with the guidance of the search direction dir_l, the number of selected features in the new $\boldsymbol{x}_i$ will become less. Thus, p_i can search for feature subsets with fewer features. On the contrary, if p_i is in the other subswarm SS_u, the features in the same neighborhood will be fixed and kept selected if all of these features are selected in the $\boldsymbol{pbest}_i$. Then more selected features will be contained in the new $\boldsymbol{x}_i$, and p_i can search for a feature subset with more features.

The pseudo-code of feature fixation in the particle update procedure is given in Algorithm 1. If the feature

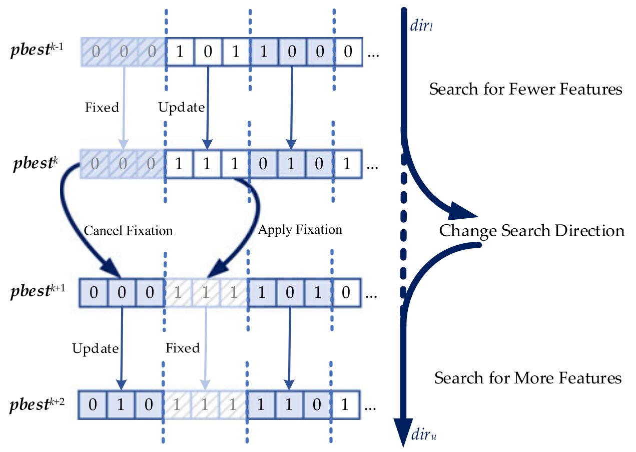


Fig. 3. An example of feature fixation guided by different search directions.

neighborhood is fixed, its corresponding dimension values of $\boldsymbol{x}_i$ will be the same as those in $\boldsymbol{pbest}_i$. Otherwise, the dimension values of $\boldsymbol{x}_i$ will be updated by the update mechanism of any chosen PSO-based algorithm in the first category mentioned in Section 2.2.

An example of a feature fixation procedure carried out by a particle is shown in Fig. 3. In the $k-1$ generation, the search direction of a particle is dir_l, i.e., searching for solutions with fewer selected features. Therefore, only feature neighborhoods with no features selected (i.e., with T zeros) in the $\boldsymbol{pbest}^{k\text{-}1}$ are fixed. Then in the k generation, this particle turns to the opposite direction dir_u, i.e., searching for solutions with more selected features. Therefore, the previously fixed neighborhoods are canceled fixation and can be updated, while those neighborhoods with all features selected (i.e., with T ones) are fixed and will be kept unchanged in the next generation until the search direction changes again.

Besides guiding particles to search for solutions with different numbers of features, feature fixation has three extra advantages. Firstly, no matter which search direction the particle is currently in, feature fixation can reduce the number of features that each particle needs to search, thus narrowing the search space. Secondly, the information of the fixed features is still retained in the swarm, so other particles can learn from the information when being updated. Thirdly, different features are fixed in different particles and they can be unfixed when the search direction of the particle is changed, which improves the diversity of the swarm.

3.5 Self-Adaptive Direction Change Strategy

Changing the search direction of each particle in time is the key to making full use of all particles in the swarm. It is expected that all particles search in the correct direction so that none of them do a meaningless search in the wrong direction. However, the swarm does not know which direction is correct at the beginning of the search. Therefore, the SADC strategy is proposed to help particles determine the correct direction according to the information gained during the evolution process. Two metrics, i.e., average improvement (AI) and average fitness (AF), are designed in the SADC strategy to balance the abilities of exploitation and exploration of the swarm.

Algorithm 2. SADC Strategy

```
Input: The swarm $S$ to be changed search direction
Output: The swarm $S$ after changing the search direction
BEGIN
 1: Count the number of particles $n_l$ and $n_u$ in $SS_l$ and $SS_u$;
 2: IF $n_l == 0$ OR $n_u == 0$ THEN
 3:    RETURN $S$;
 4: END IF
 5: Calculate $\mathrm{AI}_l$ and $\mathrm{AI}_u$ of $SS_l$ and $SS_u$ with Eq. (10);
 6: IF $\mathrm{AI}_l < \mathrm{AI}_u$ THEN
 7:    Randomly select a particle $p_r$ and $dir(p_r) == dir_l$;
 8:    $dir(p_r) \leftarrow dir_u$;
 9: END IF
10: IF $\mathrm{AI}_l > \mathrm{AI}_u$ THEN
11:    Randomly select a particle $p_r$ and $dir(p_r) == dir_u$;
12:    $dir(p_r) \leftarrow dir_l$;
13: END IF
14: IF $\mathrm{AI}_l == \mathrm{AI}_u$ THEN
15:    Calculate $\mathrm{AF}_l$ and $\mathrm{AF}_u$ of $SS_l$ and $SS_u$ with Eq. (11);
16:    IF $\mathrm{AF}_l < \mathrm{AF}_u$ THEN
17:       Randomly select a particle $p_r$ and $dir(p_r) == dir_l$;
18:       $dir(p_r) \leftarrow dir_u$;
19:    END IF
20:    IF $\mathrm{AF}_l > \mathrm{AF}_u$ THEN
21:       Randomly select a particle $p_r$ and $dir(p_r) == dir_u$;
22:       $dir(p_r) \leftarrow dir_l$;
23:    END IF
24: END IF
25: RETURN $S$;
END
```

The AI of a subswarm is the average boost fitness value of the particles in the subswarm within several generations. Considering that in the kth generation, the value of AI from generation $(k-W)$ to k is calculated by:

$$\mathrm{AI} = \frac{1}{|SS|} \sum_{p_i \in SS} \left(f(\boldsymbol{pbest}_i^k) - f(\boldsymbol{pbest}_i^{k-W}) \right) \tag{10}$$

where SS is the subswarm SS_l or SS_u, $|SS|$ is the size of SS, p_i is a particle of SS, $\boldsymbol{pbest}_i^k$ is the historical optimal position of p_i in the kth generation, W is the generation window for AI calculation, and f is the fitness function. A larger AI value means that the search direction is more promising and the particles searching in this direction are more likely to find a better solution.

The AF of a subswarm is the average fitness value of all $\boldsymbol{pbest}$s in the subswarm, which can be calculated by:

$$\mathrm{AF} = \frac{1}{|SS|} \sum_{p_i \in SS} f(\boldsymbol{pbest}_i^k) \tag{11}$$

The larger the value of AF is, the better the particles in the subswarm perform, and the more likely the optimal solution is to be found in this search direction.

The procedure of the SADC strategy is described in Algorithm 2. First, the number of particles in the two opposite directions is counted. If there is no particle in one of the directions, it means that all particles have the same direction and the correct direction has been determined by the swarm. Therefore, no particle will change its search direction. Otherwise, a random particle in the direction with a less AI value will change its direction. Furthermore, if the AI values of the two directions are the same, a random particle in the direction with a less AF value will change its direction. Notice that if the AI values and the AF values of both directions are the same, all particles will keep their direction unchanged.

Algorithm 3. Framework of BDFF

Input: The total number of features D, the number of features in a feature neighborhood T, the maximum number of fitness evaluations MAX_FE, the size of the swarm N, window of generations for changing direction W
Output: The global optimal solution found by the swarm $\boldsymbol{gbest}$

```
BEGIN
 1: Set the number of fitness evaluations $FEs \leftarrow 0$;
 2: Initialize the swarm with Eq. (6);
 3: Initialize the search direction of each particle with Eq. (7);
 4: Sort features with SU;
 5: Divide $D$ features into $R$ feature neighborhoods;
 6: $k \leftarrow 1$;
 7: WHILE $FEs < MAX\_FE$ DO
 8:     FOR $i = 1$ to $N$ DO
 9:         Update $\boldsymbol{x}_i$ of particle $p_i$ with Algorithm 1;
10:         Evaluate $\boldsymbol{x}_i$ and update $\boldsymbol{pbest}_i$;
11:     END FOR
12:     Update $\boldsymbol{gbest}$;
13:     IF $k \% W == 0$ THEN
14:         Change search direction of $p_i$ with Algorithm 2;
15:     END IF
16:     $k \leftarrow k + 1$;
17: END WHILE
18: RETURN $\boldsymbol{gbest}$;
END
```

The SADC strategy can be divided into two stages: early exploration and late exploitation. In the early stage, the SADC strategy prefers the search direction corresponding to the subswarm with a greater AI value even if its AF value is worse. Because the AF values of both SS_l and SS_u are at a poor level in the early stage, a subswarm with better AF may not indicate that its direction is correct. Instead, a more promising direction with better AI is worthy of greater efforts to search, which enhances the exploration ability of the swarm S. In the late stage, the AI values of both SS_l and SS_u are likely to be the same and equal to 0. Then the SADC strategy encourages S to search in the direction corresponding to the subswarm with better AF, which enhances the exploitation ability of S. With the two metrics to assess the search directions of particles, the SADC strategy can make the trade-off between the exploration and the exploitation in different stages.

3.6 Overall Framework

The overall framework is described in Algorithm 3. First, each particle is initialized and assigned a search direction. Then, the features are sorted by SU and divided into feature neighborhoods. In each generation, particles fix some feature neighborhoods according to their search directions and only update those features that are not fixed, thus reducing the combinations of features to search and narrowing their search space. After every W generations, BDFF adjusts the search directions of particles adaptively to balance the exploration ability and the exploitation ability of the swarm. Noting that there is no specific update mechanism for the position in the BDFF framework. Therefore, most PSO-based algorithms that mainly focus on designing different evolutionary mechanisms and have no mechanism to narrow the search space can be adopted in the framework of BDFF. In this paper, we adopt the update mechanism of BBPSO-ACJ mentioned in Section 2.3.

The time complexity of BDFF is O($MAX\_GEN \times N \times (D_{AU} + T_E)$), where MAX_GEN is the maximum number of generations, N is the size of the swarm, D_{AU} is the average number of unfixed features of the whole swarm, and T_E is the time complexity of evaluating a particle. Usually, D_{AU} is less than the total number of features D in the BDFF framework. In the worst case, the time complexity of BDFF is O $(MAX\_GEN \times N \times (D + T_E))$.

TABLE 1
Detailed Information of Datasets

Dataset	#Samples	#Features	#Classes	Data Type
Colon	62	2000	2	discrete
WarpAR10P	130	2400	10	continuous
GLIOMA	50	4434	4	continuous
Leukemia_1	72	5327	3	discrete
9_Tumor	60	5726	9	continuous
TOX_171	171	5748	4	continuous
Brain_Tumor_1	90	5920	5	continuous
Nci9	60	9712	9	discrete
Arcene	200	10000	2	continuous
CLL_SUB_111	111	11340	3	continuous
Lung_Cancer	203	12600	5	continuous
SMK_CAN_187	187	19993	2	continuous

4 Experimental Results and Analysis

In this section, experiments are carried out to evaluate the performance of BDFF on twelve public large-scale feature selection problems compared with other PSO-based algorithms.

4.1 Datasets

We used twelve public datasets for feature selection in the experiments and the detailed information is listed in Table 1, where symbol "#" means the number of corresponding items. All the used datasets are for classification problems and can be accessed from [26] and [31]. A common characteristic of the twelve datasets is that they have a small number of samples but a large number of features, which makes it difficult to solve the classification problems.

4.2 Algorithms for Comparison and Parameter Settings

All the used algorithms and their parameter settings are listed in Table 2. The proposed BDFF using the update mechanism of BBPSO-ACJ is named BBPSO-ACJ-BDFF. We used six PSO-based algorithms which can be used in feature selection for comparison. BPSO [37], BVDPSO [39], BBPSO-ACJ [22], and CSO [23] can be classified in the first category as mentioned in Section 2.2. BPSO and BVDPSO are two typical PSOs for binary optimization problems and they are

TABLE 2
Parameter Settings of Algorithms

Algorithm	Parameter Settings
BPSO [37]	$N = 20$, $v \in [-6, 6]$, $c_1 = c_2 = 2.01$, $w = 1$.
BVDPSO [39]	$N = 20$, $w \in [0.4, 0.9]$, $c_1 = c_2 = 2$, selected threshold $\alpha = 0.5$.
CSO [23]	$N = 100$, control factor $\phi = 0.1$, selected threshold $\lambda = 0.5$.
BBPSO-ACJ [22]	$N = 20$, selected threshold $\lambda = 0.5$.
VLPSO-LS [25]	$N = \min\{\text{features}/20, 300\}$, $c = 1.49445$, $w \in [0.4, 0.9]$, selected threshold $\lambda = 0.6$, max iterations to renew exemplar: 7, number of divisions: 12, max iterations for length changing: 9, local search tries: 100, local search flipping rate: 0.25.
HFS-C-P [29]	$N = 20$.
BBPSO-ACJ-BDFF	$N = 20$, $T = 3$, $W = 10$.

treated as the baseline methods. CSO and BBPSO-ACJ are algorithms specifically proposed for large-scale feature selection problems recently. VLPSO-LS [25] and HFS-C-P [29], both of which are representative and very new algorithms in the second category, use the correlation to get information on the importance of features and then focus on features with higher importance to narrow the search space. The parameter settings of the compared algorithms are the same as those in their corresponding papers. Since the swarm size N of each algorithm is different, we limited the maximum number of fitness evaluations (MAX_FE) for all algorithms to 5000.

For classification problems, we chose the k-nearest neighbor (k-NN) method as the classifier because k-NN has a stable classification performance on different datasets [50]. The parameter k of k-NN was set to be 5 in the experiments. First, 70% of samples in each dataset are randomly selected as the training dataset and the remaining 30% of samples are reserved as the test dataset. Then the classification accuracy obtained by k-NN with 5-fold cross-validation on the training dataset was adopted as the evaluation function for all algorithms in the training process. After training, the best feature subset found by each algorithm will be tested on the test dataset with the k-NN classifier to get its classification accuracy. Each algorithm was run 20 times on each dataset independently with different random seeds to reduce random statistical errors in the results. To verify the significant difference between different algorithms, we employ the Wilcoxon rank sum test [51] on the experimental results with a significance level of 0.05. Three symbols are used to indicate the Wilcoxon rank sum test results: symbols "+" and "–" indicate that our proposed algorithm is significantly superior to and inferior to the compared algorithm, respectively, while symbol "=" indicates that there is no significant difference between our algorithm and the compared algorithm.

All algorithms were implemented in C++ with an open-source library named Feature Selection Toolbox 3 [52]. In terms of the hardware environment, experiments were carried out on a platform with an Intel Core i7-10700F CPU @2.90GHz and a total memory of 8 GB.

4.3 Comparison Results and Discussion

The average classification accuracy, number of selected features, and running time obtained by the seven algorithms over 20 independent runs on the 12 datasets are compared in Tables 3, 4, and 5, respectively, where the value in **bold** represents the best result among all algorithms.

Compared with BPSO, BVDPSO, CSO, and BBPSO-ACJ, our BBPSO-ACJ-BDFF performs better on most datasets, obtaining a higher or similar classification accuracy but a much smaller subset of selected features. The classification accuracy of BBPSO-ACJ-BDFF is superior to or similar to those of the four algorithms on all datasets except datasets TOX_171 and Brain_Tumor_1. In terms of the number of selected features, BBPSO-ACJ-BDFF has a significantly better performance than BPSO, BVDPSO, and BBPSO-ACJ on most datasets. On 6 of the 12 datasets, BBPSO-ACJ-BDFF requires fewer features than CSO while achieving better or similar accuracy. In addition, the average running time of BBPSO-ACJ-BDFF is less than BPSO, BVDPSO, CSO, and BBPSO-ACJ on most datasets. Not only the feature fixation strategy but also the smaller feature subsets found by BDFF help BBPSO-ACJ-BDFF spend less time searching for the best solution.

Compared with VLPSO-LS, BBPSO-ACJ-BDFF obtains a higher classification accuracy on 3 datasets and a similar classification accuracy on 8 datasets. Considering all datasets, BBPSO-ACJ-BDFF with a rank sum of 38 outperforms VLPSO-LS with a rank sum of 45 on the classification accuracy. Though VLPSO-LS can find a smaller feature subset on some datasets, it always spends much more time searching for solutions than BBPSO-ACJ-BDFF, as shown in Table 5. This is because the local search strategy of VLPSO-LS requires the correlation result between each pair of features, which is time-consuming especially on datasets with a large number of features and samples. On the contrary, BBPSO-ACJ-BDFF only requires the correlation result between each feature and the class label, so it spends less time than VLPSO-LS.

Compared with HFS-C-P, BBPSO-ACJ-BDFF has a significantly better classification accuracy on 3 datasets and a similar accuracy on 8 datasets. BBPSO-ACJ-BDFF also obtains a smaller feature subset on 6 datasets than HFS-C-P. Overall, BBPSO-ACJ-BDFF has a more consistent performance than HFS-C-P on different datasets. For example, HFS-C-P achieves the highest accuracy on datasets WarpAR10P and 9_Tumor, while it gets the lowest accuracy on datasets Leukemia_1 and Arcene. A possible reason to explain the poor performance of HFS-C-P on some datasets is that it relies much on the correlation to filter irrelevant features and cluster relevant features, which greatly affects the final result. When sometimes the correlation measures cannot indicate the importance of features accurately, particles may be misled by such information and finally find poor solutions. On the contrary, the proposed BDFF framework reduces the search space according to the feature fixation mechanism instead of correlation measures, so it is more adaptable to different datasets.

4.4 Further Analysis of BDFF

In this subsection, we give a further discussion on BDFF and try to figure out how much search space the feature fixation mechanism can reduce.

TABLE 3
Average Classification Accuracies Obtained by Algorithms on the 12 Datasets

Dataset	BBPSO-ACJ-BDFF	BPSO	BVDPSO	CSO	BBPSO-ACJ	VLPSO-LS	HFS-C-P
Colon	**0.889**	0.861(+)	0.853(+)	0.866(=)	0.858(=)	0.858(+)	0.855(+)
WarpAR10P	0.583	0.444(+)	0.441(+)	0.498(+)	0.453(+)	0.546(+)	**0.621(–)**
GLIOMA	0.731	0.733(=)	0.731(=)	0.708(=)	0.736(=)	**0.750(=)**	0.719(=)
Leukemia_1	0.867	0.876(=)	0.857(=)	**0.878(=)**	0.872(=)	0.841(=)	0.841(=)
9_Tumor	0.532	0.534(=)	0.520(=)	0.534(=)	0.527(=)	0.516(=)	**0.561(=)**
TOX_171	0.618	0.631(=)	0.647(–)	0.638(=)	**0.650(–)**	0.628(=)	0.590(=)
Brain_Tumor_1	0.805	0.814(=)	**0.820(–)**	0.802(=)	0.807(=)	0.791(=)	0.816(=)
Nci9	0.370	0.305(+)	0.295(+)	0.345(=)	0.309(+)	**0.475(–)**	0.295(+)
Arcene	0.784	0.770(+)	0.769(=)	0.774(=)	0.780(=)	**0.796(=)**	0.769(=)
CLL_SUB_111	0.599	0.540(+)	0.541(+)	0.596(=)	0.571(+)	**0.617(=)**	0.576(=)
Lung_Cancer	0.772	0.771(=)	0.780(=)	**0.780(=)**	0.780(=)	0.752(+)	0.772(=)
SMK_CAN_187	0.692	0.673(+)	0.663(+)	0.678(=)	0.671(+)	**0.709(=)**	0.676(+)
+/=/–	NA	6/6/0	5/5/2	1/11/0	4/7/1	3/8/1	3/8/1
Rank Sum	**38**	51	60	40	44	45	56

"+", "=", and "–" indicate that BBPSO-ACJ-BDFF is significantly superior to, similar to, and significantly inferior to the compared algorithm, respectively.

TABLE 4
Average Number of Selected Features Obtained by Algorithms on the 12 Datasets

Dataset	BBPSO-ACJ-BDFF	BPSO	BVDPSO	CSO	BBPSO-ACJ	VLPSO-LS	HFS-C-P
Colon	420.8	1232.7(+)	1179.3(+)	538.5(=)	786.1(+)	349.3(=)	**170.4(–)**
WarpAR10P	**13.3**	1450.0(+)	1356.1(+)	204.2(+)	730.7(+)	464.8(+)	374.1(+)
GLIOMA	1637.8	2603.5(+)	2385.6(+)	626.9(–)	1924.7(+)	**425.8(–)**	674.7(–)
Leukemia_1	1436.9	3271.7(+)	3125.4(+)	1402.7(=)	2246.6(+)	541.9(–)	**203.8(–)**
9_Tumor	2009.8	3533.7(+)	3605.3(+)	3607.4(+)	2297.7(=)	**420.6(–)**	4008.0(+)
TOX_171	1623.4	3560.3(+)	3549.1(+)	2997.0(+)	2343.4(+)	735.9(–)	**197.9(–)**
Brain_Tumor_1	922.5	3599.7(+)	3385.9(+)	1547.9(+)	2224.6(+)	**350.9(–)**	1484.6(+)
Nci9	**417.3**	5853.3(+)	5329.3(+)	864.7(=)	3390.1(+)	692.6(=)	4944.2(+)
Arcene	1490.6	6176.4(+)	6062.5(+)	3013.6(+)	3324.3(+)	835.5(=)	**581.4(–)**
CLL_SUB_111	881.9	6993.1(+)	6801.9(+)	1342.3(=)	3907.5(+)	**627.6(=)**	2931.9(+)
Lung_Cancer	3569.0	7771.7(+)	8065.9(+)	6575.1(+)	4755.5(+)	1258.9(–)	**903.3(–)**
SMK_CAN_187	2315.4	12375.6(+)	12187.1(+)	2320.8(=)	7002.7(+)	**1401.7(–)**	4889.7(+)
+/=/–	NA	12/0/0	12/0/0	6/5/1	11/1/0	1/4/7	6/0/6
Rank Sum	30	80	72	44	55	**21**	34

"+", "=", and "–" indicate that BBPSO-ACJ-BDFF is significantly superior to, similar to, and significantly inferior to the compared algorithm, respectively.

First, we introduce an indicator named feature fixation rate (*FFR*) for further analysis, which can be calculated by:

$$FFR = \frac{D_F}{D_H} \tag{12}$$

where D_F is the total fixed dimensions that have been ignored in the update phase so far and D_H is the total dimensions that have been handled (including being ignored and being updated) in the update phase so far. The higher the value of *FFR* is, the more the search space is reduced. Therefore, the value of D_{AU} mentioned in Section 3.6 is equal to $(1 - FFR) \times D$.

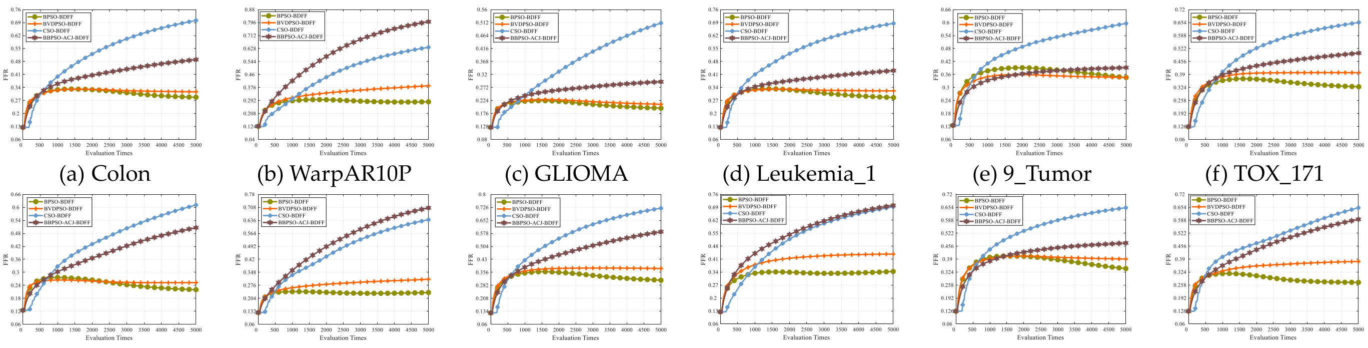
As BDFF is a general framework and can be applied to other PSO variants except BBPSO-ACJ, we also adopted the update mechanism of BPSO, BVDPSO, and CSO in the BDFF and used the same parameters as BBPSO-ACJ-BDFF so that the analysis can be more generalized and convincing. For the four PSO-based algorithms with BDFF, i.e., BPSO-BDFF, BVDPSO-BDFF, CSO-BDFF, and BBPSO-ACJ-BDFF, we recorded the total *FFR* values of all particles during the search and drew the change curves for further analysis, as shown in Fig. 4.

In the beginning, the four PSOs with BDFF have similar *FFR* values which are about $1/2^3$ because the parameter T of each algorithm is set to be 3. If a particle is guided by the same search direction and keeps updating its historical optimal position, it can usually fix more features, so its *FFR* value then increases. Among all algorithms, CSO-BDFF gets the highest *FFR* value on 10 datasets. The *FFR* value of CSO-BDFF is always over 50% and is over 70% on datasets Colon and Arcene. BBPSO-ACJ-BDFF ranks second among all algorithms and also achieves an *FFR* value of over 40% on all datasets except GLIOMA. On datasets WarpAR10P, Nci9, and CLL_SUB_111, the *FFR* value of BBPSO-ACJ-BDFF is over 60%. Though BPSO-BDFF and BVDPSO-BDFF often have lower *FFR* values than the other two algorithms, they still ignore over 20% of the dimensions in their search to narrow their search space. Noting that the *FFR* value of BPSO-BDFF and BVDPSO-BDFF does not always increase

TABLE 5
Average Running Time (minute) of the Algorithms on the 12 Datasets

Dataset	BBPSO-ACJ-BDFF	BPSO	BVDPSO	CSO	BBPSO-ACJ	VLPSO-LS	HFS-C-P
Colon	0.22	0.45(+)	0.43(+)	0.33(+)	0.34(+)	0.25(=)	**0.12(–)**
WarpAR10P	**0.58**	1.56(+)	1.48(+)	0.87(+)	1.11(+)	13.95(+)	0.75(+)
GLIOMA	0.95	1.52(+)	1.34(+)	0.90(–)	1.23(+)	6.88(+)	**0.67(–)**
Leukemia_1	1.32	2.48(+)	2.38(+)	1.90(+)	1.94(+)	6.14(+)	**0.45(–)**
9_Tumor	**1.80**	2.53(+)	2.61(+)	3.36(+)	2.05(+)	15.98(+)	3.43(+)
TOX_171	3.35	5.67(+)	5.73(+)	5.52(+)	4.43(+)	142.53(+)	**0.73(–)**
Brain_Tumor_1	**1.46**	3.22(+)	3.09(+)	2.87(+)	2.46(+)	41.31(+)	1.59(+)
Nci9	**1.61**	6.64(+)	6.26(+)	3.72(+)	4.93(+)	5.78(+)	6.76(+)
Arcene	6.80	14.56(+)	15.21(+)	11.78(+)	10.56(+)	432.93(+)	**4.73(–)**
CLL_SUB_111	**3.04**	11.02(+)	10.85(+)	4.41(+)	8.21(+)	232.74(+)	6.03(+)
Lung_Cancer	11.66	19.31(+)	20.65(+)	18.66(+)	15.69(+)	724.54(+)	**3.62(–)**
SMK_CAN_187	**15.69**	45.32(+)	43.91(+)	37.46(+)	33.91(+)	1967.77(+)	22.25(+)
+/=/–	NA	12/0/0	12/0/0	11/0/1	12/0/0	11/1/0	6/0/6
Rank Sum	**19**	67	63	41	41	77	28

"+", "=", and "–" indicate that BBPSO-ACJ-BDFF is significantly superior to, similar to, and significantly inferior to the compared algorithm, respectively.



(a) Colon (b) WarpAR10P (c) GLIOMA (d) Leukemia_1 (e) 9_Tumor (f) TOX_171
(g) Brain_Tumor_1 (h) Nci9 (i) Arcene (j) CLL_SUB_111 (k) Lung_Cancer (l) SMK_CAN_187

Fig. 4. The change curve of average *FFR* of PSO-based algorithms with BDFF from evaluation times 0 to 5000 on 12 datasets.

with time. If the search direction of a particle is changed frequently, the previously fixed dimensions will be shifted to be flexible and can be updated, thus the *FFR* value will probably decrease.

4.5 From the Perspective of Feature Subset Size

We recorded the average number of selected features of the global best particle in the swarm throughout the search process and drew the change curves of four PSOs and four PSOs with BDFF to describe how they vary with the increase in evaluation times, as shown in Fig. 5.

The number of features contained in the best particle of all algorithms at the beginning is about half of the total number of features because they adopt the same initialization method in Eq. (6). However, the change curves of PSOs with BDFF are different from the original PSOs and finally can achieve a smaller number of selected features. It can be known from the results in Tables 3 and 4 that solutions with the highest accuracy can be obtained with the number of features less than half on all datasets. BPSO and BVDPSO can improve their accuracy as the number of evaluations increases, but the number of their features cannot be effectively reduced. From the perspective of the number of selected features, BPSO and BVDPSO probably insist on searching in the wrong direction, so the accuracy they achieve finally is usually inferior to other algorithms. After being combined with the BDFF framework, BPSO-BDFF and BVDPSO-BDFF can search in the correct direction and finally achieve a higher classification accuracy with fewer features. CSO and BBPSO-ACJ can reduce the features of the best particle while improving the accuracy in most datasets with the increase in evaluation times. However, they cannot exploit the search space with fewer features and usually find a solution with more features than CSO-BDFF and BBPSO-ACJ-BDFF finally. Therefore, the BDFF framework can help particles explore and exploit more efficiently not only from the perspective of the fitness but also from the perspective of the number of selected features, and finally guide them to approach the global optimal solution in terms of the number of selected features.

4.6 Influence of Correlation Information

To investigate the effect of correlation information used in BDFF, we removed the feature sorting step via SU and then recorded the comparison results between BBPSO-ACJ-BDFF and its variant in Table 6. The BBPSO-ACJ-BDFF-w/o-SU has similar performances on the classification accuracy and the size of the feature subset to BBPSO-ACJ-BDFF and there is no significant difference between the two algorithms on most datasets. Without the help of correlation

TABLE 6
Comparison Results Between BDFF and BDFF Without SU

Dataset	BBPSO-ACJ-BDFF		BBPSO-ACJ-BDFF-w/o-SU	
	Accuracy	Features	Accuracy	Features
Colon	**0.889**	**420.8**	0.845(+)	514.6(=)
WarpAR10P	0.583	**13.3**	**0.599(=)**	16.7(+)
GLIOMA	**0.731**	**1637.8**	0.719(=)	1689.9(=)
Leukemia_1	0.867	1436.9	**0.870(=)**	**1329.9(=)**
9_Tumor	0.532	2009.8	**0.534(=)**	**1922.0(=)**
TOX_171	0.618	**1623.4**	**0.625(=)**	1795.7(=)
Brain_Tumor_1	0.805	**922.5**	**0.811(=)**	1117.6(=)
Nci9	**0.370**	417.3	0.361(=)	**303.7(=)**
Arcene	**0.784**	1490.6	0.780(=)	**1027.9(=)**
CLL_SUB_111	**0.599**	881.9	0.596(=)	**475.6(=)**
Lung_Cancer	**0.772**	3569.0	0.767(=)	**2685.7(=)**
SMK_CAN_187	**0.692**	**2315.4**	0.683(=)	2509.7(=)
+/=/–	NA	NA	1/11/0	1/11/0

"+", "=", and "–" indicate that BBPSO-ACJ-BDFF is significantly superior to, similar to, and significantly inferior to BBPSO-ACJ-BDFF-w/o-SU, respectively.

information, BBPSO-ACJ-BDFF-w/o-SU also finds a solution with high accuracy and a small number of features. However, BBPSO-ACJ-BDFF can perform slightly better and find a solution with higher accuracy and fewer features on some datasets after using the SU measure for feature sorting.

An important reason why BDFF can work well without using correlation information is that it only ranks features and discards none of the features according to the correlation information. Though the features in the same feature neighborhood are no longer strongly correlated without feature sorting, they can be reserved and still have the opportunity to be selected. Moreover, the small size of the feature neighborhood also reduces the correlation requirement for features in the same neighborhood because the feature fixation operation is fine-grained. However, some redundant features will be mixed in the same feature neighborhood when the feature sorting is missing, which makes it a bit more difficult to select a better feature subset with fewer features.

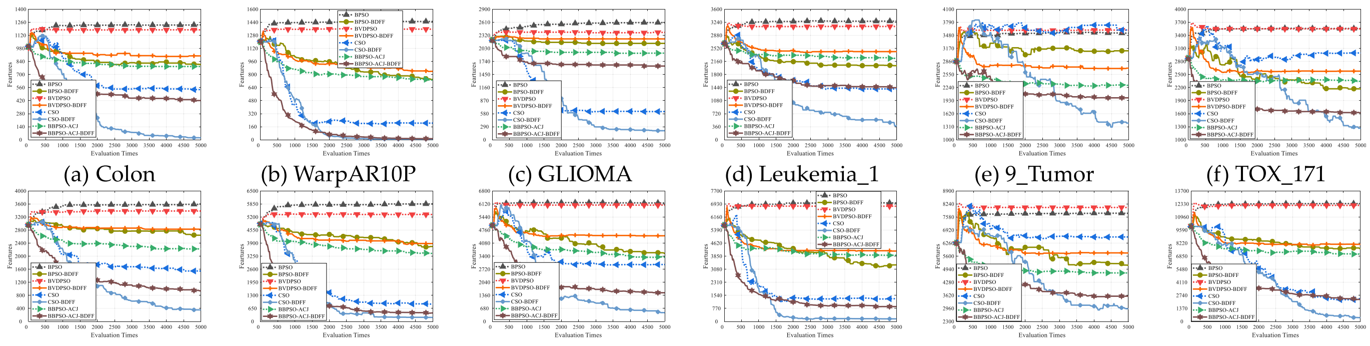
4.7 Influence of the Size of Feature Neighborhood

The parameter T represents that there are T features in a feature neighborhood. To investigate the influence of T, variants of BBPSO-ACJ-BDFF with different values of T are compared and the results are presented in Table 7. The performance of the accuracy is the best when T is 3 among all

TABLE 7
Average Results Among BBPSO-ACJ-BDFF Variants with Different Size T of Feature Neighborhood

Dataset	Accuracy					Features				
	$T=1$	$T=2$	$T=3$	$T=4$	$T=5$	$T=1$	$T=2$	$T=3$	$T=4$	$T=5$
Colon	0.829(+)	0.882(=)	**0.889**	0.855(+)	0.863(=)	**26.7(–)**	121.1(–)	420.8	646.5(+)	699.6(+)
WarpAR10P	0.544(+)	0.545(=)	**0.583**	0.578(=)	0.529(=)	**5.3(–)**	9.5(=)	13.3	90.0(+)	262.1(+)
GLIOMA	0.728(=)	0.722(=)	0.731	**0.739(=)**	0.733(=)	**220.6(–)**	933.7(–)	1637.8	1911.5(+)	2018.1(+)
Leukemia_1	0.839(=)	0.865(=)	0.867	**0.883(=)**	0.870(=)	**499.2(–)**	914.7(–)	1436.9	1475.4(=)	2107.2(+)
9_Tumor	0.484(+)	0.525(=)	**0.532**	0.523(=)	0.525(=)	**583.9(–)**	1539.1(=)	2009.8	2280.8(=)	2206.1(=)
TOX_171	0.630(=)	0.619(=)	0.618	0.638(=)	**0.642(=)**	**928.5(–)**	1185.9(–)	1623.4	1760.4(=)	2165.0(+)
Brain_Tumor_1	0.800(=)	0.800(=)	0.805	**0.813(=)**	0.813(=)	**164.7(–)**	391.8(–)	922.5	1630.1(+)	1858.1(+)
Nci9	0.386(=)	**0.386(=)**	0.370	0.332(+)	0.332(+)	**60.9(–)**	147.0(–)	417.3	1054.2(+)	1856.0(+)
Arcene	0.772(=)	0.783(=)	**0.784**	0.779(=)	0.780(=)	**192.0(–)**	600.9(–)	1490.6	2449.9(+)	2299.4(+)
CLL_SUB_111	0.577(=)	0.587(=)	**0.599**	0.574(=)	0.567(+)	**80.1(–)**	173.8(–)	881.9	1218.7(+)	2818.2(+)
Lung_Cancer	0.763(=)	0.777(=)	0.772	**0.778(=)**	0.778(=)	2525.9(–)	**2143.4(–)**	3569.0	4152.5(=)	3794.0(=)
SMK_CAN_187	0.681(=)	0.686(=)	**0.692**	0.671(=)	0.682(=)	**119.6(–)**	958.9(–)	2315.4	3154.5(=)	4915.6(+)
+/=/–	3/9/0	0/12/0	NA	2/10/0	2/10/0	0/0/12	0/2/10	NA	7/5/0	10/2/0
Rank Sum	49	34	**27**	33	36	**13**	23	36	51	57

"+", "=", and "–" mean that BBPSO-ACJ-BDFF with T=3 is significantly superior to, similar to, and significantly inferior to the compared variant, respectively.



(a) Colon (b) WarpAR10P (c) GLIOMA (d) Leukemia_1 (e) 9_Tumor (f) TOX_171
(g) Brain_Tumor_1 (h) Nci9 (i) Arcene (j) CLL_SUB_111 (k) Lung_Cancer (l) SMK_CAN_187

Fig. 5. The change curve of the average number of features contained in the global best particle from evaluation times 0 to 5000 on 12 datasets.

variants. As the value of T increases, the number of features also increases in most cases. A small T makes it easy to fix features and reduce most of the search space, so the particle is likely to find a small feature subset but miss the optimal solution. On the contrary, a large T makes it hard to fix features and can retain the optimal solution in the search space, but it also increases the difficulty to find the optimal solution because the search space is too large. Therefore, the value of T is usually recommended to be 3.

5 Conclusion

This paper proposes the BDFF framework for PSO to solve large-scale feature selection problems. In the BDFF framework, each particle owns a search direction and searches for solutions with different numbers of features. According to its search direction, each particle can fix some features and ignore them in the update stage with the feature fixation mechanism, thus narrowing its search space. BDFF also adopts the representation of the feature neighborhood, dividing all features into small neighborhoods. Then it uses the feature neighborhood as the basic unit to fix features and refine the granularity of operations. Moreover, BDFF designs the SADC strategy to change the search direction of particles adaptively, making a trade-off between exploration and exploitation in different search processes. Experimental results on 12 public feature selection datasets show that the proposed BDFF framework can help particles approach the optimal solution from the perspective of the fitness value and the number of features. Compared with the correlation-based algorithms, BDFF can reduce the search space effectively without over-relying on the correlation information and have a consistent performance on most classification problems. Besides, BDFF can be treated as a general framework and be combined with PSO-based feature selection algorithms to further improve their performances.

For future work, there still exist some challenges to be overcome for PSO-based feature selection algorithms, such as reducing the time of evaluation and dealing with some complicated real-world problems. Therefore, some promising techniques designed for expensive optimization, such as the data-driven method [53], [54], the scale-adaptive fitness evaluation method [55], [56], and parallel/distributed computing methods [57], [58], [59], [60], can be combined to further improve the performance of BDFF.

References

[1] V. Bolón-Canedo, N. Sánchez-Maroño, and A. Alonso-Betanzos, "Recent advances and emerging challenges of feature selection in the context of big data," *Knowl. Based Syst.*, vol. 86, pp. 33–45, Sep. 2015.

[2] E. Hancer, B. Xue, and M. Zhang, "A survey on feature selection approaches for clustering," *Artif. Intell. Rev.*, vol. 53, no. 6, pp. 4519–4545, Aug. 2020.

[3] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, no. 1, pp. 273–324, Dec. 1997.

[4] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theor. Comput. Sci.*, vol. 209, no. 1, pp. 237–260, Dec. 1998.

[5] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.

[6] Y. Zhang et al., "An embedded vertical-federated feature selection algorithm based on particle swarm optimisation," *CAAI Trans. Intell. Technol.*, to be published, 2022, doi: 10.1049/cit2.12122.

[7] H. Zhang, D. Wu, F. Nie, R. Wang, and X. Li, "Multilevel projections with adaptive neighbor graph for unsupervised multi-view feature selection," *Inf. Fusion*, vol. 70, pp. 129–140, Jun. 2021.

[8] X. Li, H. Zhang, R. Wang, and F. Nie, "Multiview clustering: A scalable and parameter-free bipartite graph fusion method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 330–344, Jan. 2022.

[9] B. H. Nguyen, B. Xue, and M. Zhang, "A survey on swarm intelligence approaches to feature selection in data mining," *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100663.

[10] Z. H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 59–110, Jan. 2022.

[11] J. Y. Li, Z. H. Zhan, and J. Zhang, "Evolutionary computation for expensive optimization: A survey," *Mach. Intell. Res.*, vol. 19, no. 1, pp. 3–23, Feb. 2022.

[12] Z. H. Zhan, J. Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, vol. 483, pp. 42–58, Apr. 2022.

[13] Z. G. Chen, Z. H. Zhan, S. Kwong, and J. Zhang, "Evolutionary computation for intelligent transportation in smart cities: A survey," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 83–102, May 2022.

[14] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[16] X. Xia et al., "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[17] J. R. Jian, Z. H. Zhan, and J. Zhang, "Large-scale evolutionary optimization: A survey and experimental comparative study," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 3, pp. 729–745, Mar. 2020.

[18] J. R. Jian, Z. G. Chen, Z. H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021.

[19] Z. J. Wang et al., "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2715–2729, Jun. 2020.

[20] Y. Hou, G. Hao, Y. Zhang, F. Gu, and W. Xu, "A multi-objective discrete particle swarm optimization method for particle routing in distributed particle filters," *Knowl. Based Syst.*, vol. 240, Mar. 2022, Art. no. 108068.

[21] J. Q. Yang, C. H. Chen, J. Y. Li, D. Liu, T. Li, and Z. H. Zhan, "Compressed-encoding particle swarm optimization with fuzzy learning for large-scale feature selection," *Symmetry*, vol. 14, no. 6, Jun. 2022, Art. no. 1142.

[22] C. Qiu, "Bare bones particle swarm optimization with adaptive chaotic jump for feature selection in classification," *Int. J. Comput. Intell. Syst.*, vol. 11, no. 1, pp. 1–14, Jan. 2018.

[23] S. Gu, R. Cheng, and Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Comput.*, vol. 22, no. 3, pp. 811–822, Feb. 2018.

[24] Y. Xue, B. Xue, and M. Zhang, "Self-adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 5, 2019, Art. no. 50.

[25] B. Tran, B. Xue, and M. Zhang, "Variable-length particle swarm optimization for feature selection on high-dimensional classification," *IEEE Trans. Evol. Comput.*, vol. 23, no. 3, pp. 473–487, Jun. 2019.

[26] K. Chen, B. Xue, M. Zhang, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 7172–7186, Jul. 2022.

[27] X. F. Song, Y. Zhang, Y. N. Guo, X. Y. Sun, and Y. L. Wang, "Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 882–895, Oct. 2020.

[28] K. Chen, B. Xue, M. Zhang, and F. Zhou, "Correlation-guided updating strategy for feature selection in classification with surrogate-assisted particle swarm optimisation," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1015–1029, Oct. 2022.

[29] X. F. Song, Y. Zhang, D. W. Gong, and X. Z. Gao, "A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9573–9586, Sep. 2022.

[30] X. Song, Y. Zhang, D. Gong, H. Liu, and W. Zhang, "Surrogate sample-assisted particle swarm optimization for feature selection on high-dimensional data," *IEEE Trans. Evol. Comput.*, to be published, 2022, doi: 10.1109/TEVC.2022.3175226.

[31] J. Li et al., "Feature selection: A data perspective," *ACM Comput. Surv.*, vol. 50, no. 6, 2017, Art. no. 94.

[32] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 874–888, Feb. 2021.

[33] F. Cheng, J. J. Cui, Q. J. Wang, and L. Zhang, "A variable granularity search based multi-objective feature selection algorithm for high-dimensional data classification," *IEEE Trans. Evol. Comput.*, to be published, 2022, doi: 10.1109/TEVC.2022.3160458.

[34] P. Wang, B. Xue, J. Liang, and M. Zhang, "Differential evolution based feature selection: A niching-based multi-objective approach," *IEEE Trans. Evol. Comput.*, to be published, 2022, doi: 10.1109/TEVC.2022.3168052.

[35] Y. Xue, Y. Tang, X. Xu, J. Liang, and F. Neri, "Multi-objective feature selection with missing data in classification," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 2, pp. 355–364, Apr. 2022.

[36] T. Li, Z. H. Zhan, J. C. Xu, Q. Yang, and Y. Y. Ma, "A binary individual search strategy-based bi-objective evolutionary algorithm for high-dimensional feature selection," *Inf. Sci.*, vol. 610, pp. 651–673, Sep. 2022.

[37] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf . Syst., Man, Cybern.*, 1997, pp. 4104–4108.

[38] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Appl. Soft Comput.*, vol. 18, pp. 261–276, May 2014.

[39] M. Shen, Z. H. Zhan, W. N. Chen, Y. J. Gong, J. Zhang, and Y. Li, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7141–7151, Dec. 2014.

[40] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[41] B. Tran, B. Xue, and M. Zhang, "A new representation in pso for discretization-based feature selection," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1733–1746, Jun. 2018.

[42] A. Bommert, X. Sun, B. Bischl, J. Rahnenführer, and M. Lang, "Benchmark for filter methods for feature selection in high-dimensional classification data," *Comput. Statist. Data Anal.*, vol. 143, Mar. 2020, Art. no. 106839.

[43] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," *Mach. Learn.*, vol. 53, no. 1, pp. 23–69, Oct. 2003.

[44] D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proc. Speech Nat. Lang.*, 1992, pp. 212–217.

[45] B. Singh, N. Kushwaha, and O. P. Vyas, "A feature subset selection technique for high dimensional data using symmetric uncertainty," *J. Data Anal. Inf. Process.*, vol. 2, no. 4, pp. 95–105, Nov. 2014.

[46] L. Y. Chuang, C. S. Yang, K. C. Wu, and C. H. Yang, "Gene selection and classification using Taguchi chaotic binary particle swarm optimization," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 13367–13377, Sep. 2011.

[47] X. Song, Y. Zhang, D. Gong, and X. Sun, "Feature selection using bare-bones particle swarm optimization with mutual information," *Pattern Recognit.*, vol. 112, Apr. 2021, Art. no. 107804.

[48] J. González-López, S. Ventura, and A. Cano, "Distributed selection of continuous features in multilabel classification using mutual information," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 7, pp. 2280–2293, Jul. 2020.

[49] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, 2003, pp. 80–87.

[50] Y. Xue, T. Tang, W. Pang, and A. X. Liu, "Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106031.

[51] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometr. Bull.*, vol. 1, no. 6, pp. 80–83, 1945.

[52] P. Somol, P. Vácha, S. Mikeš, J. Hora, P. Pudil, and P. Zid, "Introduction to feature selection toolbox 3–the c++ library for subset search, data modeling and classification," Acad. Sci. Czech Republic Inst. Inf. Theory Automation, Prague, Czech Republic, Tec. Rep. no. 2287, Oct. 2010.

[53] J. Y. Li, Z. H. Zhan, C. Wang, H. Jin, and J. Zhang, "Boosting data-driven evolutionary algorithm with localized data generation," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 923–937, Oct. 2020.

[54] J. Y. Li, Z. H. Zhan, H. Wang, and J. Zhang, "Data-driven evolutionary algorithm with perturbation-based ensemble surrogates," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 3925–3937, Aug. 2021.

[55] S. H. Wu, Z. H. Zhan, and J. Zhang, "SAFE: Scale-adaptive fitness evaluation method for expensive optimization problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 478–491, Jun. 2021.

[56] Y. Q. Wang, J. Y. Li, C. H. Chen, J. Zhang, and Z. H. Zhan, "Scale adaptive fitness evaluation-based particle swarm optimisation for hyperparameter and architecture optimisation in neural networks and deep learning," *CAAI Trans. Intell. Technol.*, to be published, 2022, doi: 10.1049/cit2.12106.

[57] Z. H. Zhan et al., "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[58] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.

[59] J. Y. Li, Z. H. Zhan, R. D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation-level parallelism for evolutionary computation: A pipeline-based parallel particle swarm optimization," *IEEE Trans. Cybern.*, vol. 51, no. 10, pp. 4848–4859, Oct. 2021.

[60] J. Y. Li, K. J. Du, Z. H. Zhan, H. Wang, and J. Zhang, "Distributed differential evolution with adaptive resource allocation," *IEEE Trans. Cybern.*, to be published, 2022, doi: 10.1109/TCYB.2022.3153964.

Jia-Quan Yang (Student Member, IEEE) received the BS degree in network engineering from the South China University of Technology, Guangzhou, China, in 2021, where he is currently working toward the MS degree in computer science and technology with the School of Computer Science and Engineering. His research interests include swarm intelligence, feature selection, large-scale optimization, and their applications in real-world optimization problems.

Qi-Te Yang (Student Member, IEEE) received the BS degree from Dalian University, Dalian, China, in 2017, and the MS degree from Xiangtan University, Xiangtan, China, in 2020. He is currently working toward the PhD degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests mainly include computational intelligence, data-driven optimization, multiobjective optimization, and their applications in real-world problems. He has been invited as a reviewer of the *IEEE Transactions on Evolutionary Computation*, the *Swarm and Evolutionary Computation*, and the *Neurocomputing*.

Ke-Jing Du received the BS degree from Sun Yat-Sen University, Guangzhou, China, in 2012, and the MS degree from the City University of Hong Kong, Hong Kong, in 2014. She is currently working toward the PhD degree with Victoria University, Melbourne, VIC, Australia. Her current research interests include evolutionary computation (EC) and supply chain management, especially the distributed EC and application of EC in supply chain and feature selection.

Chun-Hua Chen (Member, IEEE) received the bachelor's degree in computer science and PhD degree in information and communication engineering from the South China University of Technology, Guangzhou, China, in 2006 and 2012, respectively. He is currently an assistant professor with the School of Software Engineering, South China University of Technology, Guangzhou, China. His current research interests include evolutionary computation, feature selection, knowledge graph, distributed system, component-based software engineering, and their applications in industrial sector.

Hua Wang (Senior Member, IEEE) received the PhD degree from the University of Southern Queensland, Toowoomba, Qld., Australia in 2004. He is now a full time professor with Victoria University. He has expertise in electronic commerce, business process modeling, and enterprise architecture. As a chief Investigator, three Australian Research Council (ARC) Discovery grants have been awarded since 2006, and 200 peer-reviewed scholar papers have been published.

Sang-Woon Jeon (Member, IEEE) received the BS and MS degrees in electrical engineering from Yonsei University, Seoul, South Korea, in 2003 and 2006, respectively, and the PhD degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2011. He has been an associate professor with the Department of Military Information Engineering (undergraduate school) and the Department of Electronics and Communication Engineering (graduate school), Hanyang University, Ansan, South Korea, since 2017. He was a recipient of the Haedong Young Scholar Award in 2017, which was sponsored by the Haedong Foundation and given by the Korea Institute of Communications and Information Science (KICS), the Best Paper Award of the IEEE International Conference on Communications in 2015, the Best Thesis Award from the Department of Electrical Engineering, KAIST, in 2012.

Jun Zhang (Fellow, IEEE) received the PhD degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002. He is currently a Korea Brain Pool fellow professor with Hanyang University, South Korea. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits. Dr. Zhang was a recipient of the Changjiang chair professor with the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists with the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an associate editor of the *IEEE Transactions on Evolutionary Computation* and the *IEEE Transactions on Cybernetics*.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's and PhD degrees in computer science from the Sun Yat-Sen University, Guangzhou China, in 2007 and 2013, respectively. He is currently the Changjiang Scholar Young professor with t