# Improved binary particle swarm optimization for feature selection with new initialization and search space reduction strategies

**Article** *in* Applied Soft Computing · July 2021

DOI: 10.1016/j.asoc.2021.107302

CITATIONS
13

READS
324

**3 authors**, including:

An-Da Li
Tianjin University of Commerce
**14** PUBLICATIONS **94** CITATIONS

SEE PROFILE

Bing Xue
Victoria University of Wellington
**326** PUBLICATIONS **7,367** CITATIONS

SEE PROFILE

# Improved Binary Particle Swarm Optimization for Feature Selection with New Initialization and Search Space Reduction Strategies

An-Da Li[a,*], Bing Xue[b], Mengjie Zhang[b]

[a]*School of Management, Tianjin University of Commerce, Tianjin 300134, China*
[b]*Evolutionary Computation Research Group, Victoria University of Wellington, Wellington 6140, New Zealand*

**Abstract**

Feature selection (FS) is an important preprocessing technique for dimensionality reduction in classification problems. Particle swarm optimization (PSO) algorithms have been widely used as the optimizers for FS problems. However, with the increase of data dimensionality, the search space expands dramatically, which proposes significant challenges for optimization methods, including PSO. In this paper, we propose an improved sticky binary PSO (ISBPSO) algorithm for FS. ISBPSO adopts three new mechanisms based on a recently proposed binary PSO variant, sticky binary particle swarm optimization (SBPSO), to improve the evolutionary performance. First, a new initialization strategy using the feature weighting information based on mutual information is proposed. Second, a dynamic bits masking strategy for gradually reducing the search space during the evolutionary process is proposed. Third, based on the framework of memetic algorithms, a refinement procedure conducting genetic operations on the personal best positions of ISBPSO is used to alleviate the premature convergence problem. The results on 12 UCI datasets show that ISBPSO outperforms six benchmark PSO-based FS methods and two conventional FS methods (sequential forward selection and sequential backward selection) — ISBPSO obtains either higher or similar accuracies with fewer features in most cases. Moreover, ISBPSO substantially reduces the computation time compared with benchmark PSO-based FS methods. Further analysis shows that all the three proposed mechanisms are effective for improving the search performance of ISBPSO.

*Keywords:* Classification, feature selection, particle swarm optimization, initialization

## 1. Introduction

Feature selection (FS) has shown to be an essential preprocessing technique for classification and cluster problems in machine learning and data mining scenarios [1, 2, 3]. It aims to select a subset of critical

*https://doi.org/10.1016/j.asoc.2021.107302

*Corresponding author
*Email address:* adli@tjcu.edu.cn (An-Da Li)

features from the original feature set because substantially irrelevant and redundant features are contained in the data collected from the real-world applications. As summarized in [1], FS provides benefits such as improving the classification performance, reducing the time complexity, and enhancing the interpretability of the learned model. In this paper, we mainly focus on FS for classification.

FS can be categorized into filter and wrapper approaches in terms of the used feature importance evaluation strategies [4]. A filter approach evaluates the features based on the intrinsic properties of the data [5]. Feature weighting measures based on the information theory (e.g., symmetrical uncertainty [6], mutual information (MI) [7], and dispersion entropy [8]), distance measures (e.g., Relief [9] and ReliefF [10]), etc., are generally used to build filters. A wrapper approach evaluates the importance of a feature subset based on the classification performance of a learning algorithm [11]. Wrappers generally cost more computation time than filters since evaluating each feature subset requires training the learning algorithm. Meanwhile, wrappers usually obtain better classification results since the classification performance is directly used for feature importance evaluation.

FS is reported to be an NP-hard problem that tries to find the best subset from $2^N - 1$ possible subsets of features given a dataset with $N$ features. This proposes a big challenge for the search strategies applied to FS problems. Sequential forward selection (SFS) and sequential backward selection (SBS) are two typical wrappers based on heuristics [12]. These two methods apply greedy search strategies to find the best feature subset, while they can be easily trapped into local optima. Metaheuristics, inspired by various behaviors or phenomena in nature (e.g. the hunting behaviors), are widely recognized as powerful optimization approaches. In recent years, many kinds of metaheuristics have been proposed for solving FS problems. These metaheuristics include genetic algorithms (GAs) [13], particle swarm optimization (PSO) [14], genetic programming (GP) [15], differential evolution (DE) [16], ant colony optimization (ACO) [17], and the artificial bee colony (ABC) algorithm [18]. In particular, an increasing number of studies have applied PSO to FS [4].

Continuous PSO (CPSO), which simulates the behaviors of birds flocking or fish schooling, is a metaheuristic stochastic algorithm proposed by Kennedy and Eberhart [19] in 1995. A binary PSO (BPSO) algorithm was proposed in 1997 to handle binary variables [20]. Both CPSO [21] and BPSO [22] have been used to build FS methods. BPSO can encode a feature subset more straightforwardly because each bit in a particle's position is either 1 or 0, which denotes whether a feature is selected or not. Recently, Xue et al. [23] proposed a new probability-based BPSO (PBPSO) for FS, which shows to have better performance than standard BPSO. Nevertheless, PBPSO lacks a very important concept of PSO, i.e., momentum. Nguyen et al. [24] proposed a sticky BPSO (SBPSO) algorithm which introduces a stickiness parameter in PBPSO to mimic the momentum of a particle.

PSO-based wrappers have been studied a lot in recent years for their higher classification performance than filters. The FS objective is to eliminate as many irrelevant and redundant features as possible. To

achieve this objective, existing PSO-based wrappers mainly adopt an integrated fitness function combining maximizing the classification performance and minimizing the feature subset size [23, 24, 25, 26, 27]. However, with the increase of data dimensionality, the search space of FS problems expands exponentially. This causes a problem that a satisfactory feature subset with a few key features may not be found because the numerous feasible feature subsets (solutions) propose significant challenges for PSO considering the limited computational resources. In other words, PSO algorithms typically evolve too slowly to achieve a good FS result for high dimensional data. Therefore, improving the evolutionary effectiveness and efficiency of PSO algorithms is always a key task for PSO-based wrappers.

In this paper, we propose a novel PSO-based wrapper method for FS. The proposed FS method adopts an improved SBPSO (ISBPSO) algorithm as the optimizer, where three novel mechanisms are adopted based on SBPSO to enhance the search performance:

a) We propose a feature weighting directed initialization (FWDI) method to improve the quality of the initial swarm. FWDI adopts the feature weighting results based on MI to initialize the particles, where the bits (features) with higher weights are given higher probabilities to be initialized as 1.

b) We propose a dynamic bits masking (DBM) strategy to narrow the search space of ISBPSO. This strategy iteratively puts a mask on the features every certain number of generations to stop these features from further evolving. Such a strategy can substantially narrow the search space during the evolutionary process, which is beneficial for ISBPSO to find better solutions in a smaller search space.

c) Inspired by the idea of memetic algorithms, we propose a refinement procedure for ISBPSO that conducts the genetic operations to update the personal best position (*pbest*) of each particle. The *pbest* of each particle is a key factor of the evolutionary mechanism of PSO algorithms. Thus, updating the *pbest*s with genetic operators, i.e., crossover and mutation, can help ISBPSO to escape from the local optima.

We will test the proposed FS method on 12 datasets from the UCI repository [28] to investigate whether the proposed ISBPSO algorithm is effective and efficient for FS.

The remainder of this paper is organized as follows. Section 2 presents the background and related works. Section 3 introduces the proposed ISBPSO-based FS approach. Section 4 describes the experimental design. Section 5 presents the parameter study results of ISBPSO. Section 6 presents and discusses the FS results of the proposed method and benchmark methods. Further analysis of the proposed method is provided in Section 7. Finally, Section 8 gives the conclusions.

## 2. Background

This section provides the basic concepts of MI and SBPSO, which are the basic components of the proposed approach, followed by an overview of related work on FS.

*2.1. Mutual Information (MI)*

MI, a basic concept of Shannon's information theory, can capture both the linear and non-linear relations between two random variables [29]. Entropy, which measures the uncertainty of a variable, is a basic component of MI. For a discrete variable $X \in \mathbb{X}$, the entropy $H(X)$ is obtained as

$$H(X) = -\sum_{x \in \mathbb{X}} p(x) \log p(x), \tag{1}$$

where $p(x)$ denotes the prior probability of $X = x$. Given another discrete variable $Y \in \mathbb{Y}$, the conditional entropy $H(X|Y)$ is obtained as

$$H(X|Y) = -\sum_{y \in \mathbb{Y}} \sum_{x \in \mathbb{X}} [p(x, y) \log p(x|y)], \tag{2}$$

where $p(x|y)$ is the posterior probability of $X = x$ given $Y = y$, and $p(x, y)$ is the joint probability of $X = x$ and $Y = y$. In information theory, the MI of variables $X$ and $Y$ is defined as

$$I(Y; X) = \sum_{y \in \mathbb{Y}} \sum_{x \in \mathbb{X}} [p(y, x) \log \frac{p(y, x)}{p(y)p(x)}]. \tag{3}$$

Thus, we obtain a different form of MI based on Eqs. (1) and (2) as

$$I(Y; X) = H(X) - H(X|Y), \tag{4}$$

which is also the definition of information gain (IG). If variables $X$ and $Y$ are continuous, MI is defined as

$$I(Y; X) = \int \int p(y, x) \log \frac{p(y, x)}{p(y)p(x)} \mathrm{d}y\mathrm{d}x. \tag{5}$$

In the case of continuous variables, estimating the probability distributions of $X$ or $Y$ is hard, which complicates the calculation of Eq. (5). A proper way is discretizing continuous variables first, and then using Eq. (4) to calculate MI [1]. In this paper, the minimum description length (MDL) method [30] is used to discretize the continuous variables. Based on the above definitions, given a dataset with a set $\mathbb{F} = \{F_1, F_2, ..., F_N\}$ of features and a class variable $C$, we can obtain a feature weight vector $\mathbf{W} = (w_1, w_2, ..., w_N)$ using the MI measure, where each $w_d$, $d = 1, 2, ..., N$, is calculated as

$$w_d = I(C; F_d). \tag{6}$$

*2.2. Sticky BPSO (SBPSO)*

SBPSO [24] is a recently proposed BPSO variant that adopts the flipping probability instead of velocity in the traditional BPSO algorithm to update a particle's position. To retain the momentum characteristic of PSO, a stickiness parameter is used in SBPSO making a particle tends to stick to the position it just moved to.

Suppose that the swarm size is $K$, each particle's position, flipping probability, and stickiness parameter vectors at the $t$th generation are denoted by $\mathbf{X}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{i,N}^t)$, $\mathbf{P}_i^t = (p_{i,1}^t, p_{i,2}^t, ..., p_{i,N}^t)$, and $\mathbf{S}_i^t = (s_{i,1}^t, s_{i,1}^t, ..., s_{i,N}^t)$, where $x_{i,d}^t \in \{0,1\}$, $p_{i,d}^t \in [0,1]$, $s_{i,d}^t \in [0,1]$, $d = 1, 2, ..., N$, $i = 1, 2, ..., K$. At each generation, SBPSO uses $\mathbf{P}_i^{t+1}$ to update particle $i$'s position from $\mathbf{X}_i^t$ to $\mathbf{X}_i^{t+1}$. Specifically, each bit $x_{i,d}^t$ $(d = 1, 2, ..., N)$ in $\mathbf{X}_i^t$ is updated to $x_{i,d}^{t+1}$ as

$$x_{i,d}^{t+1} = \begin{cases} 1 - x_{i,d}^t & ; rand() < p_{i,d}^{t+1} \\ x_{i,d}^t & ; otherwise \end{cases}, \tag{7}$$

where $rand()$ is a random value in $[0,1]$ from the uniform distribution. The probability vector $\mathbf{P}_i^{t+1}$ is updated from $\mathbf{P}_i^t$. Each $p_{i,d}^{t+1}$ in $\mathbf{P}_i^{t+1}$ is obtained as

$$p_{i,d}^{t+1} = i_m \cdot (1 - s_{i,d}^t) + i_p \cdot |pbest_{i,d} - x_{i,d}^t| + i_g \cdot |gbest_d - x_{i,d}^t|, \tag{8}$$

where $s_{i,d}^t$, $pbest_{i,d}$, and $gbest_d$ denote particle $i$'s stickiness parameter, *pbest*, and the global best position (*gbest*) on the $d$th bit, respectively. $i_p$ and $i_g$ determine particles' moving tendency to *gbest* and *gbest*. According to Eq. (8), $s_{i,d}^t > 0$ lowers the flipping probability of $p_{i,d}^{t+1}$, and the intensity of this tendency is decided by $i_m$. SBPSO sets $i_m + i_p + i_g = 1$ to ensure the maximum value of $p_{i,d}^{t+1}$ is 1, and the default settings are $i_m = 0.25$, $i_p = 0.25$, and $i_g = 0.5$ since *gbest* affects the movements of particles more than *pbest* and the momentum. The stickiness parameter $s_{i,d}^t$ decreases over time, which means that a bit is likely to stick to the new position it just move to. And $s_{i,d}^{t+1}$ at each generation is calculated as

$$s_{i,d}^{t+1} = \begin{cases} s_{i,d}^t - 1/L & ; x_{i,d}^{t+1} = x_{i,d}^t,\ s_{i,d}^t > 0 \\ 1 & ; x_{i,d}^{t+1} \neq x_{i,d}^t \end{cases}, \tag{9}$$

where the step parameter $L$ is a constant that determines the number of generations a stickiness parameter decreases from 1 to 0. It can be seen in Eq. (9), the stickiness parameter changes to 1 when a flip happens. In SBPSO, the initial stickiness parameter $s_{i,d}^0 = 1$, $d = 1, 2, ..., N$.

### 2.3. Related work on feature selection

This section briefly reviews the existing metaheuristics for FS. Here, we categorize these approaches into the non-PSO and PSO-based methods and review them in the following two sub-sections.

#### 2.3.1. Metaheuristic-based (non-PSO) feature selection methods

In recent years, various metaheuristics have been proposed for building FS methods. Among these methods, GAs are one of the most popular metaheuristics used for solving FS problems. Some studies have applied single objective GAs to FS. Oh et al. [13] proposed a hybrid GA-based FS method that applies local search operations in addition to the genetic operations to improve the FS performance. However, applying the local search also increases the computation time of the search algorithm. Freeman et al. [31] proposed a tree-based classification method that adopts a GA to simultaneously select features and optimize the tree structure. However, since one additional optimization task, i.e., optimizing the tree structure, is involved in addition to FS, the search space is increased which further impacts the final search performance of the GA. Similarly, Tao et al. [32] proposed a GA-SVM based hospitalization expense modeling method, which uses a GA for simultaneously selecting features and tuning the parameters in support vector machines. In addition to single objective GAs, multi-objective GAs have also been used for FS. Zhu et al. [33] proposed an improved NSGA-III algorithm for FS in intrusion detection. This method adopts a novel niche preservation strategy to improve the global search performance of the proposed improved NSGA-III algorithm. Li et al. [34] proposed a hybrid multi-objective optimization method combining the search mechanism of GAs with that of direct multi-search for selecting key features (quality characteristics) in production processes related to product quality. The multi-objective GA-based FS methods return a set of feature subsets. Hence, strategies for selecting the final feature subset should be further studied from a practical point of view.

Other metaheuristics, such as GP, DE, ACO, and ABC, have also been used for FS. Nag and Pal [35] designed a GP algorithm that minimizes three objectives, i.e., false positives, false negatives, and the number of leaf nodes in the tree, for simultaneously selecting features and designing a GP-based classifier. Hancer et al. [16] proposed two versions (single objective and multi-objective) of DE for building FS methods. In this study, the relevance and redundancy measures based on ReliefF, fisher score, and MI are built. These measures are further optimized by DE for obtaining a feature subset with a good relevance property while containing a few redundant features. Zhang et al. [36] proposed a FS method that optimizes both the accuracy and feature subset size based on a multi-objective binary DE algorithm. To improve the FS performance, the proposed DE algorithm applies a local search step to further update the solutions after applying the DE operators, while additional computation time for the local search step is needed. Tabakhi and Moradi [17] proposed a FS method based on ACO. The proposed method uses ACO to obtain the feature weights and select features according to the weights. However, this method needs a predefined number of features to be selected. Hancer et al. [18] proposed an ABC-based FS method, which integrates

evolutionary-based similarity search mechanisms in a binary ABC variant to improve the FS performance. Moreover, Shunmugapriya and Kanmanib [37] proposed a hybrid algorithm combining the search mechanisms of ACO and ABC for FS. Although various metaheuristics have been used for solving FS in literature, it is worth further examining effective strategies to improve the convergence speed of the metaheuristics for the high-dimensional FS problems which have a very large search space.

### *2.3.2. PSO-based feature selection methods*

The PSO algorithms, including both CPSO and BPSO, have been widely used for FS since PSO has the advantages of promising global search performance and fast convergence.

In the applications of CPSO to FS, each particle position is encoded as a real vector, where each element in the vector corresponds to a feature. Each element is compared with a predefined threshold parameter to decide whether the corresponding feature is selected or not. The CPSO and its variants have been widely applied to FS problems in recent years. Xue et al. [21] studied several initialization and updating mechanisms considering the number of selected features for CPSO-based wrappers. However, the initialization mechanisms studied are still based on randomization which does not consider the intrinsic characteristics of data. Competitive particle optimization (CSO) is a CPSO variant recently proposed in [38] for large scale optimization problems. In CSO, a particle learns from a better particle randomly selected from the swarm, rather than learns from the *pbest* and *gbest* used by standard CPSO. Gu et al. [39] adapted CSO to high dimensional FS problems. However, this approach does not adopt any other strategies designed for FS to improve the feature reduction efficiency. Comprehensive learning PSO (CLPSO) is another CPSO variant, in which a particle learns from its *pbest* or an exemplar (particle) chosen by the binary tournament selection from the swarm [40]. Tran et al. [41] proposed a FS method based on the CLPSO algorithm. In this method, particles are set as vectors of different lengths, and the short-length particles are used to promote the method to select fewer features. However, how to decide a good combination of particle lengths needs to be further investigated. Being a CPSO variant, bare bones PSO (BBPSO) has also been used for building FS methods [42]. Zhang et al. [43] proposed a BBPSO-based FS method, which also adopts a uniform combination strategy to avoid the premature convergence problem. Tran et al. [42] proposed a FS method that adopts BBPSO for simultaneously selecting and discretizing features. In BBPSO, a Gaussian distribution based on the *pbest* and *gbest* is generated to decide the position of a particle without using the velocity. However, the generated Gaussian distribution of BBPSO may not be sufficiently good for FS, which is a combinatorial optimization problem, since BBPSO is proposed by analyzing the particle position distribution of continuous optimization problems [44]. Chen et al. [27] proposed a CPSO variant named HPSO-SSM for FS. HPSO-SSM embeds a spiral-shape updating mechanism as the spare particle updating strategy. However, strategies for improving the search performance for high dimensional FS are not designed in this method.

In addition to the single objective CPSO algorithms, multi-objective CPSO algorithms have also been established for FS. Xue et al. [45] established two multi-objective CPSO-based FS methods, i.e, CMDPSOFS and NSPSOFS. The experimental results have shown that CMDPSOFS that adopts the ideas of crowding, mutation, and dominance obtain better FS performance than NSPSOFS and existing GA-based FS methods. Additionally, local search strategies have also been embedded in multi-objective CPSO algorithms to improve the FS performance. For example Nguyen et al. [46] proposed a multi-objective CPSO-based FS method called ISRPSO which adopts several kinds of local search operations, including inserting, removing, and swapping, to improve the FS performance. Amoozegar and Minaei-Bidgoli [47] proposed a multi-objective CPSO-based FS method called RFPSOFS which adopts a process to refine the archive set during the evolutionary process to improve the FS process. Local search can improve the search performance of PSO whereas it also increases the computation time of the algorithm.

BPSO reduces the search space compared with CPSO for FS since each bit of a particle position is either 1 or 0 instead of a continuous value in CPSO [41]. In recent years, the BPSO-based FS methods have been widely studied. Huang and Dun [48] applied the BPSO to simultaneously find the best feature subset while optimizing the parameters for an SVM. Mafarja et al. [49] studied the effects of time inertia weighting changing strategies used in BPSO for FS. The results have shown that gradually decreasing the inertia weight can improve the performance of BPSO for FS. Banka and Dara [25] introduced the hamming distance to the velocity updating procedure of BPSO, which is further applied to FS in high-dimensional data. Zhang et al. [22] proposed a FS method for spam detection using a modified BPSO algorithm. The proposed method applies the mutation operator to improve the global search performance of BPSO. Moradi and Gholampour [50] proposed a hybrid PSO algorithm that combines the solution updating mechanism of BPSO with a local search technique to select distinct features by considering the correlation information of features. Jain et al. [51] proposed a two-stage FS method for gene selection and cancer classification based on correlation-based feature selection (CFS) and BPSO. In this method, the CFS method is first used to filter some features, and BPSO is used to further obtain the final feature subset. The above mentioned BPSO-based FS methods are based on the standard framework of the BPSO proposed in [20], where a sigmoid function is used to convert each element of the velocity to a value in (0,1), which is the probability to set a bit in the particle position as 1. However, the velocity used in BPSO proposes an opposite effect for evolving new solutions compared with that in CPSO, which lowers the performance of BPSO [52]. In [53], the performance of BPSO and CPSO on FS problems was compared. The results show that BPSO performs worse than CPSO although FS is a combinatorial optimization problem. To address the limitations of traditional BPSO, PBPSO [23] and SBPSO [24] were proposed and applied to FS problems. These two methods adopt a flipping probability vector instead of the velocity vector in traditional BPSO algorithms to update each particle's position. Moreover, SBPSO can be seen as an improved version of PBPSO. In SBPSO, a stickiness parameter is used to depict the momentum of particles to further improve the search

performance.

In summary, both CPSO and BPSO algorithms have been studied for FS. However, the growth of data dimensionality proposes significant challenges for optimizers (including PSO) in FS, as the search space of FS problems expands exponentially with the increase of data dimensionality. Therefore, improving the effectiveness and efficiency of PSO algorithms considering the intrinsic properties of FS problems is worth studying. In this paper, we will explore three mechanisms from the perspectives of swarm initialization, search space reduction, and premature alleviation for SBPSO to improve its FS performance.

## 3. The proposed approach

This section proposes the ISBPSO algorithm for FS. First, basic elements for FS are introduced. Second, the FWDI method which initializes the swarm using feature weighting results of MI is proposed. Third, the DBM strategy that dynamically narrows down the search space during the evolutionary process is proposed. Fourth, the genetic operations for alleviating the premature convergence problem are introduced. Finally, the overall ISBPSO-based FS approach is given.

### *3.1. Basic elements for FS*

#### *3.1.1. Solution encoding*

Given a dataset with $N$ features, a particle's position is encoded as a binary vector $\mathbf{X} = (x_1, x_2, ..., x_N)$. Each bit $x_d \in \{0, 1\}$, $d = 1, 2, ..., N$, where 1 denotes the $d$th feature is selected, and 0 denotes it is not selected.

#### *3.1.2. Fitness function*

The wrapper framework is adopted to build the FS method. Therefore, the classification accuracy of a learning algorithm is used to determine the fitness of a particle position (i.e, a feature subset). As well as the classification performance, selecting a small feature subset is another goal of FS. Thus, in this paper, a fitness function combining the accuracy and the percentage of eliminated features is used as in [24, 23]. This fitness function is defined as

$$Fit(\mathbf{X}) = \theta \cdot acc(\mathbf{X}) + (1 - \theta) \cdot (1 - \#\mathbf{X}/N), \tag{10}$$

where $acc(\mathbf{X})$ denotes the estimated classification accuracy of the feature subset $\mathbf{X}$, $\#\mathbf{X}$ denotes the number of selected features, $\theta$ and $1 - \theta$ denote the weights of the classification accuracy and the percentage of eliminated features, respectively. We set $\theta = 0.9$ as used in [24] because classification performance is a more critical factor in FS. To avoid the FS bias, a wrapper-based FS method usually uses the $K$-fold CV [12] on the dataset fed to the FS method to estimate the classification performance of a feature subset. In this paper, the 5-fold CV is adopted to estimate $acc(\mathbf{X})$ in Eq. (10) as used in [12].

### 3.2. Feature weighting directed initialization (FWDI)

Random initialization (RI) is commonly used for initializing particles in a PSO algorithm. Generally, in a BPSO algorithm, a position bit is set to 1 or 0 with a predefined probability. RI is a critical step for initializing a sufficiently diversified swarm. However, RI does not consider the intrinsic characteristics of data, which can provide useful initialization information for PSO to start with a higher quality swarm. MI introduced in Section 2 is a typical feature weighting method with very low computational complexity, and can catch both the linear and non-linear relations between features and the class label to rank features. Thus, we propose a novel initialization method for ISBPSO using MI.

Without loss of generality, suppose that $\mathbf{W} = (w_1, w_2, ..., w_N)$ is a weight vector from a feature weighting method. Each weight $w_d > 0$, $d = 1, ..., N$, denotes the importance level of feature $d$ measured by the weighting method (higher is better). For a particle position $\mathbf{X}_i^0 = (x_{i,1}^0, x_{i,2}^0, ..., x_{i,N}^0)$, let $\mathbf{P} = (p_1, p_2, ..., p_N)$ be the probability vector, where each element $p_d$, $d = 1, ..., N$, denotes the probability to initialize the $d$th bit (feature) as 1. A high probability value should be assigned to the bit with a high weight for obtaining a high quality initial swarm. To achieve this goal, a transfer function can be used to convert the weights to probability values for the initialization, since a transfer function can convert a real value in [0, 1]. Generally, there are two types of transfer functions, S-shaped and V-shaped transfer functions [54]. An illustration of the two types of transfer functions is shown in Fig. 1. According to Fig. 1, a S-shaped transfer function tends to convert a smaller number to a value closer to 0 and convert a larger number to a value closer to 1. In comparison, a V-shaped transfer function converts a number with a larger absolute value to a value closer to 1. Since our objective is converting a higher weight value to a higher probability value and vice versa, we choose the S-shaped transfer function to construct the initialization method. For more details about the S-shaped and V-shaped transfer functions, please refer to [54].

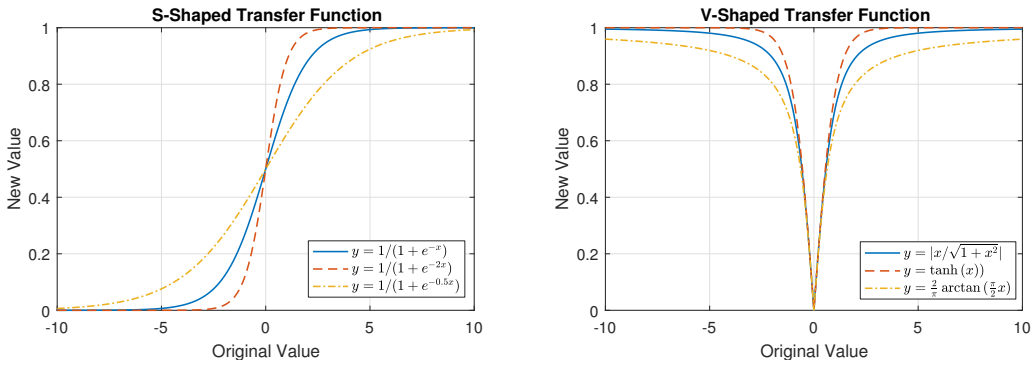


Figure 1: Illustration of S-shaped and V-shaped transfer functions.

In this paper, we construct the initialization method for ISBPSO based on the logistic function, which is one typical S-shaped transfer function. Let $z \in R$ be a variable, the standard logistic function transforms

$z$ to $z'$ as

$$z' = 1/[1 + e^{-z}]. \tag{11}$$

Two key issues should be addressed when transforming a feature weight to a probability value using the logistic function:

a) The ranges of obtained weights can be very different on different datasets. So, a normalization method is required before using the logistic function for a stable transformation. We normalize a feature weight $w$ to a value $z \in [-\sigma, \sigma]$ ($\sigma \in R+$) as

$$z = \frac{2\sigma(w - min(w))}{max(w) - min(w)} - \sigma, \tag{12}$$

where $min(w)$ and $max(w)$ denote the minimum and maximum weights in $\mathbf{W}$.

b) If $z'$ in Eq. (11) is directly used as the probability, the bits (features) with weights close to the maximum weight value would be easily set to 1 for every particle, which can dramatically decrease the swarm diversity. To cope with this problem, we lower the upper bound of $z'$ by transforming it to a value $p \in (0, \lambda)$, $\lambda \in (0.5, 1)$, as

$$p = \lambda z'. \tag{13}$$

According to Eqs. (11), (12), and (13), we obtain a modified logistic function that transforms a feature weight $w$ into a probability $p$ as

$$p = \lambda/[1 + exp(-\frac{2\sigma(w - min(w))}{max(w) - min(w)} + \sigma)]. \tag{14}$$

An illustration of the standard and modified logistic functions is shown in Fig. 2.

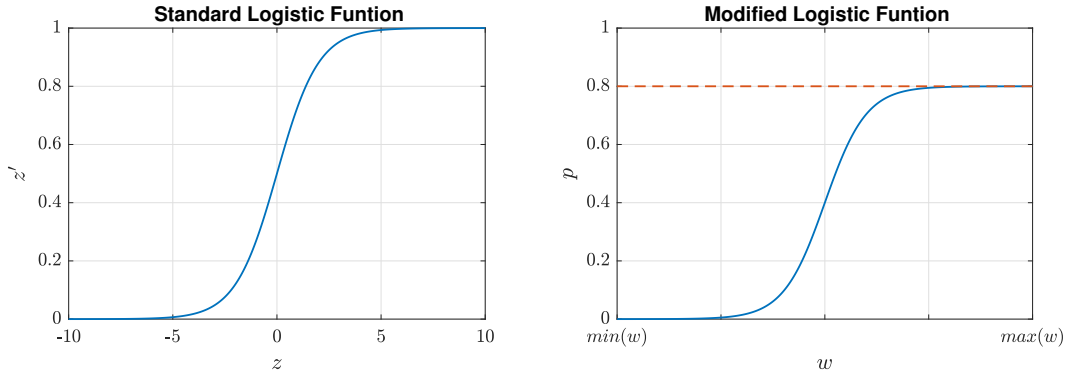


Figure 2: Illustration of the standard logistic function in Eq. (11) and the modified logistic function in Eq. (14), where $\lambda = 0.8$ and $\sigma = 10$.

Based on the modified logistic function in Eq. (14), we propose the FWDI method for swarm initialization. Let $\mathbf{W} = (w_1, w_2, ..., w_N)$ be the feature weight vector from MI. This vector can be transformed into a probability vector $\mathbf{P} = (p_1, p_2, ..., p_N)$ using Eq. (14) on each $w_d$, $d = 1, 2, ..., N$. Each particle $i$'s $(i = 1, 2, ..., K)$ initial position $\mathbf{X}_i^0 = (x_{i,1}^0, x_{i,2}^0, ..., x_{i,N}^0)$ can be initialized by setting each bit $x_{i,d}^0$ to 1 with the probability $p_d$, $d = 1, 2, ..., N$, and we can obtain the initial swarm of ISBPSO.

### *3.3. Dynamic bits masking (DBM) strategy*

Traditionally, a PSO algorithm searches in a fixed $N$-dimensional (where $N$ is the number of original features) space during the evolutionary process. This requires enormous computational resources by setting a large number of particles or generations in PSO algorithms when $N$ is large (e.g., hundreds, thousands, or even more). Thus, it is beneficial to propose a search space reduction strategy that is able to reduce the required computational resources for PSO applied to FS tasks.

In this paper, we propose a search space reduction strategy named dynamic bits masking (DBM), which uses a mask to dynamically reduce the search space of ISBPSO during the evolutionary process. Fig. 3 gives an illustration of the mask. In the mask, the bits to be masked are denoted in gray. The masked bits in the particles are stopped from further evolving, which can decrease the search space. Two key issues related to the DBM strategy are a) how to update the mask and b) when to update the mask.

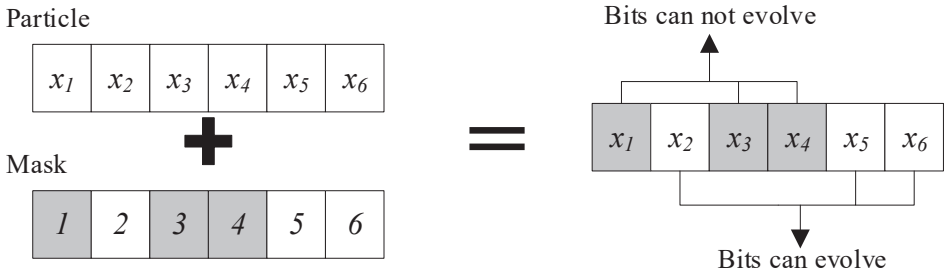


Figure 3: Illustration of the mask vector.

First, we select the bits to be masked by extracting the information from particles' *pbest*s. In most traditional PSO-based FS methods, the selected and eliminated features are finally decided when the PSO algorithms reach a stopping criterion. However, the information of particles during the evolutionary process is not sufficiently used, which can provide us useful information to further improve the optimization performance. In a FS task, the numbers of selected features of all particles decrease during the evolutionary process. The noisy or irrelevant features can be decided even before the stopping criterion is reached. If a feature (bit) is not selected by all the *pbest*s in the swarm after several generations, it is very possible that this feature is an irrelevant feature, because the solutions with this feature are very likely to be eliminated for bad fitnesses. According to the above analysis, we obtain the mask updating strategy. That is, a bit is

masked if it is not selected by all *pbest*s in the swarm.

Algorithm 1 shows the pseudocode of the mask updating strategy. In the algorithm, particles' *pbest*s are denoted by a set *pbests*, a set $\mathbb{M}_u = \{d_1, d_2, ..., d_B\}$ is used to denote the mask, where each element in $\mathbb{M}_u$ corresponds to an unmasked bit. The set $\mathbb{M}_u$ is updated by extracting the information from particles in the swarm. If a bit is not selected by all the *pbest*s in the swarm, the bit is eliminated from $\mathbb{M}_u$. The set $\mathbb{M}_u$ is updated during the evolutionary process of the algorithm. Every time the mask updating process is conducted, some bits in $\mathbb{M}_u$ would be masked (i.e., eliminated from the set). Since only the bits in $\mathbb{M}_u$ can evolve, the mask updating strategy ensures the decrease of the search space. Moreover, we conduct an amending step to set the value of each masked bit in the particles of the swarm as 0 (see line 5), hence the search space indicated by $\mathbb{M}_u$ can be consistent with the particles. Then, we can safely change the position updating mechanism shown in Eq. (7) to

$$x_{i,d}^{t+1} = \begin{cases} (1 - x_{i,d}^t) & ; rand() < p_{i,d}^{t+1}, d \in \mathbb{M}_u \\ x_{i,d}^t & ; rand() \geq p_{i,d}^{t+1}, d \in \mathbb{M}_u \\ 0 & ; d \notin \mathbb{M}_u \end{cases} . \tag{15}$$

**Algorithm 1:** Pseudocode of the mask updating strategy

**Input :** $pbests = \{pbest_1, pbest_2, ..., pbest_K\}$, particle positions $\mathbb{S}^t = \{\mathbf{X}_1^t, \mathbf{X}_2^t, ..., \mathbf{X}_K^t\}$ where $\mathbf{X}_i^t = (x_{i,1}^t, x_{i,2}^t, ..., x_{i,N}^t)$, $i = 1, ..., K$, index set of unmasked bits $\mathbb{M}_u = \{d_1, d_2, ..., d_B\}$ ;

**Output :** Updated $\mathbb{M}_u$, updated $\mathbb{S}^t$ ;

```
1 for d ∈ M_u do
2     if Σ_{i=1}^{K} pbest_{i,d} = 0 then
3         M_u ← M_u \ {d} ;
4         for i ← 1 to K do
5             x_{i,d}^t ← 0;
6         end
7     end
8 end
9 return M_u and S^t ;
```

Second, to decide when the mask $\mathbb{M}_u$ should be updated during the evolutionary process, a parameter $\mu \in (0, 1)$ is used. It means that the mask $\mathbb{M}_u$ is updated every $\mu \cdot T$ generations, where $T$ is the maximum number of generations. As stated above, the mask updating strategy is proposed under the assumption that the effectiveness of a bit to be masked has been tested in several preceding generations. Thus, before updating the mask, the optimizer needs to search for enough generations in the space decided by the previous mask. In Section 5, we will conduct tuning experiments to select a proper value for $\mu$.

### *3.4. Genetic operations*

Being a PSO algorithm, SBPSO updates each particle's position by exchanging information among its current position, its *pbest* and the *gbest*. This strategy makes SBPSO efficiently search for better solutions, however, may also cause the problem of premature convergence due to the quick decrease of the swarm diversity. Memetic algorithms generally use a refinement procedure (e.g., a local search procedure) in addition to the global search process to improve the search performance of a population-bases search algorithm [55]. Therefore, conducting a refinement procedure in ISBPSO can be a useful strategy to handle the premature convergence problem. GAs mainly use the crossover and mutation operators to update solutions. The crossover operator enables exchanging information among solutions and the mutation operator can increase the population diversity to help the algorithm escape from local optima. Inspired by this, the genetic operations are used as a a refinement procedure for ISBPSO to further improve the global search performance. The refinement procedure is conducted on the *pbest* of each particle. The improvement on the *pbest* can further improve the particle positions and *gbest* during the evolutionary process, which is beneficial for improving the evolutionary performance of ISBPSO.

**Algorithm 2:** Pseudocode of the genetic operations

**Input** : $pbests = \{pbest_1, pbest_2, ..., pbest_K\}$, index set of unmasked bits $\mathbb{M}_u$, *gbest*, crossover probability $p_c$, and mutation probability $p_m$ ;
**Output :** Updated *pbests* and *gbest*;

`/* Crossover,` $pbest_i^o = (pbest_{i,1}^o, pbest_{i,2}^o ..., pbest_{i,N}^o)$, $i = 1, ..., K$. `*/`

$\mathbb{O} = \{pbest_1^o, pbest_2^o, ..., pbest_K^o\} \leftarrow Crossover(pbests, p_c)$ ;

`/* Mutation. */`

**for** $i \leftarrow 1$ *to* $K$ **do**
  **for** $d \in \mathbb{M}_u$ **do**
    **if** $rand() < p_m$ **then**
      $pbest_{i,d}^o \leftarrow 1 - pbest_{i,d}^o$ ;
    **end**
  **end**
**end**
$pbests \leftarrow$ A set $\mathbb{Q}$ of $K$ best positions in $\mathbb{O} \cup pbests$ ;
Update *gbest* with *pbests* ;
**return** *pbests* and *gbest* ;

Algorithm 2 shows the pseudocode of the genetic operations, which is proposed to update each particle's *pbest*. First, a set $\mathbb{O}$ of offspring positions is generated by the crossover operator. Specifically, the binary tournament selection is applied to select $K/2$ pairs of parents, and then the single point crossover operator is applied to generate $K$ offspring positions with a crossover probability of $p_c$. Then, the single point mutation operator with a mutation probability of $p_m$ is applied to each position in $\mathbb{O}$. In this paper, $p_c = 0.9$ is adopted in the experiments as suggested in [56]. Note that, only the unmasked bits are permitted to mutate which ensures that the mutation operator only updates *pbests* in the search space indicated by the mask.

We set the mutation rate $p_m = 1/(N - |\mathbb{M}_u|)$ ($|\ |$ denotes the number of elements in a set), which makes the expected number of mutation bits in a particle position be 1. Finally, a set $\mathbb{Q}$ with $K$ best positions in $\mathbb{O} \cup pbests$ are used to update $pbests$ and $gbest$. Note that, it is possible that some positions in $\mathbb{Q}$ are already in current $pbests$, which means that these positions are not updated by the genetic operations. Therefore, the indices of non-updated positions in $\mathbb{Q}$ should be the same as those in $pbests$, since each position's index in $pbests$ determines the particle this position corresponds to. Moreover, it is also required to decide when to conduct the genetic operations in Algorithm 2. In this paper, we mainly use genetic operations as an assistant strategy to alleviate the premature convergence problem. We conduct the genetic operations only when $gbest$ is not updated for $\varphi \cdot T$ generations, where $\varphi \in (0, 1)$ is a user-defined parameter and $T$ is the maximum number of generations.

### *3.5. Overall approach*

The overall procedure of the ISBPSO-based FS method is shown in Algorithm 3. The proposed approach first uses the FWDI method to initialize a set $\mathbb{S}^t$ ($t = 0$) of particles. Then, at each generation, whether the algorithm reaches the condition for updating the mask or conducting the genetic operations is checked. Specifically, the mask is updated every $\mu \cdot T$ generations, and the genetic operations are conducted if $gbest$ is not updated for $\varphi \cdot T$ generations. Note that, if the genetic operations are conducted, such a generation does not conduct the PSO evolutionary process. Thus, the expected number of fitness function evaluations at each generation is equal to the swarm size, which makes a fair comparison with other methods in the experiments.

Computation time is one key performance measure to evaluate a FS method. For a metaheuristic FS method based on the wrapper framework, the evolutionary process and the fitness function evaluation process are two time-consuming parts. The fitness function evaluation process is time-consuming because in a wrapper approach a learning algorithm is involved to evaluate the fitness of each feature subset. However, theoretically evaluating the time cost of the fitness function evaluation process is hard, since the computation time for each function evaluation is highly related to the number of features contained in the feature subset. In other words, given the same original dataset, evaluating a smaller feature subset requires less computation time than evaluating a larger feature subset. This characteristic of the wrapper framework indicates that if a metaheuristic can quickly reduce the number of features, this method can perform more time-efficiently on fitness evaluations.

Based on the above discussion, here we analyze the time complexity of the proposed ISBPSO algorithm without considering the fitness function evaluation process. Suppose that there are $K$ particles in the swarm, the maximum number of generations is $T$, the length of particles is $N$ (the original number of features), and the number of instances in the dataset is $M$. We can reach the following analysis results.

First, we analyze the time complexity of the FWDI method, i.e., line 3 of Algorithm 3. The time

**Algorithm 3:** Pseudocode of ISBPSO-based FS method

**Input :** The training dataset with $N$ features, maximum number of generations $T$, swarm size $K$;
**Output :** A set $\mathbb{F}_s$ of selected features;

1 $t \leftarrow 0$, $\mathbb{M}_u \leftarrow \{1, 2..., N\}$ ;
2 $\mathbf{P}_i^t \leftarrow (0)_{\times N}$, $\mathbf{S}_i^t \leftarrow (1)_{\times N}$, $i = 1, 2, ..., K$ ;
3 $\mathbb{S}^t = \{\mathbf{X}_1^t, \mathbf{X}_2^t, ..., \mathbf{X}_K^t\} \leftarrow$ Initialize each particle's position using the FWDI method proposed in Section 3.2;
4 Evaluate the fitness value of each particle in $\mathbb{S}^t$ using Eq. (10) ;
5 Update *pbests* and *gbest* ;
6 **while** $t < T$ **do**
7  **if** *mask updating condition satisfied* **then**
8  $\mathbb{M}_u \leftarrow$ Update $\mathbb{M}_u$ using **Algorithm** 1 ;
9  $\mathbb{S}^{t+1} \leftarrow \mathbb{S}^t$ ;
10  **end**
11  **if** *genetic operations conducting condition satisfied* **then**
12  Update *gbest* and *pbest*s using **Algorithm** 2 ;
13  **else**
14  **for** *particle* $i \leftarrow 1$ *to* $K$ **do**
15  $\mathbf{P}_i^{t+1} \leftarrow$ Update $i$'s probability vector using Eq. (8) ;
16  $\mathbf{X}_i^{t+1} \leftarrow$ Update $i$'s position using Eq. (15) ;
17  $\mathbf{S}_i^{t+1} \leftarrow$ Update $i$'s stickiness parameter vector using Eq. (9) ;
18  Evaluate the fitness value of $i$ using Eq. (10) ;
19  Update $i$'s *pbest* ;
20  Update *gbest* using $\mathbf{X}_i^{t+1}$ ;
21  **end**
22  $\mathbb{S}^{t+1} \leftarrow \{\mathbf{X}_1^{t+1}, \mathbf{X}_2^{t+1}, ..., \mathbf{X}_K^{t+1}\}$ ;
23  **end**
24  $t \leftarrow t + 1$ ;
25 **end**
26 $\mathbb{F}_s \leftarrow$ Decode *gbest* ;
27 **return** $\mathbb{F}_s$ ;

complexity of FWDI is formed of two parts, feature weighting using MI and particle initialization. The feature weighting step needs to evaluate each feature's weight by calculating the MI between the feature and the class label. The time complexity of calculating the MI value for a feature is $O(M)$ [57], which is related to the number of instances. Thus, the complexity of the feature weighting step is $O(MN)$. The time complexity of the particle initialization step is $O(KN)$. Therefore, the time complexity of the FWDI method is $O(MN) + O(KN)$.

Second, we analyze the time complexity of the main loop of Algorithm 3 from lines 6 to 25. At generation $t$, if the mask vector updating step in line 8 of Algorithm 3 is conducted, the time complexity of this step is $O(KB_t)$ according to the pseudocode shown in Algorithm 1, where $B_t$ is the number of unmasked bits at generation $t$ and $B_t \leq N$. The time complexity of the genetic operations in line 12 of Algorithm 3 is $O(K) + O(K) + O(KB_t) \cong O(KB_t)$ which are formed of the binary tournament selection ($O(K)$), crossover ($O(K)$), and mutation operators ($O(KB_t)$). The time complexity of the PSO evolutionary process shown in lines 14 to 21 of Algorithm 3 is $O(KB_t)$, which is equal to that of the genetic operations step. Therefore,

for a generation $t$, if the mask updating step and the genetic operations step (or the PSO evolutionary process) are both conducted, the time complexity of this generation is $O(KB_t)+O(KB_t) \cong O(KB_t)$. Since $B_t$ denotes the number of unmasked bits at generation $t$ and $B_t$ is smaller than or equal to the length of particles $N$, the time complexity of ISBPSO at generation $t$ is smaller than or equal to $O(KN)$. Thus, we can reach the conclusion that the time complexity of the main loop of ISBPSO during the $T$ generations is not larger than $O(TKN)$, which is the time complexity of the main loop of the standard CPSO and SBPSO algorithms. In the worse case, when no bit is masked during the evolutionary process, the time complexity of the main loop of ISBPSO is $O(TKN)$.

Finally, considering both the initialization step and the main loop, the time complexity of ISBPSO is $O(MN)+O(KN)+O(TKN) \cong O((M+TK)N)$ in the worse case. It can be seen that the time complexity of ISBPSO is not only related to the swarm size and the number of features but also related to the number of instances in the dataset. This is because that the feature weighting strategy is used to guide the initialization of ISBPSO, and the time complexity of this step is related to the number of instances. But of course, as discussed earlier, the actual computation time taken also depends heavily on the time for wrapper-based fitness evaluations.

## 4. Experimental design

This section describes the datasets, benchmark methods, parameter settings, and experimental configuration used for testing the effectiveness of the proposed method.

### *4.1. Datasets*

12 datasets from the UCI machine learning repository [28] (http://archive.ics.uci.edu/ml) are used in the experiments to validate the proposed ISBPSO-based FS method. The numbers of features and instances of these datasets vary from tens to thousands, and both binary and multiclass classification problems are included. The details of these datasets are shown in Table 1, where the datasets are listed in terms of the number of features in ascending order, and "ID" is used to denote each dataset for simplicity.

### *4.2. Benchmark methods*

Eight benchmark methods based on the wrapper framework are used to test the effectiveness and efficiency of ISBPSO. They are SBPSO [24], Up BPSO (UBPSO) [58], Quantum BPSO (QBPSO) [59], CSO(C) [39], CSO(A) [39], CPSO [21], SFS [12], and SBS [12]. The SBPSO algorithm is used to test if the new mechanisms added in ISBPSO are effective. UBPSO is a recently proposed BPSO variant, which adopts a linearly increasing scheme for the inertia weight to improve the search performance. QBPSO is another recently proposed BPSO variant that uses the quantum computing mechanism to update particles. In this paper, we adapt UBPSO and QBPSO to FS problems, where the same fitness function as ISBPSO is adopted

Table 1: Details of datasets.

| Dataset | ID | #Features | #Instances | #Classes |
|---|---|---|---|---|
| Sonar | Sonar | 60 | 208 | 2 |
| Libras Movement | Movement | 90 | 360 | 15 |
| Hill-valley | Hillvalley | 100 | 1212 | 2 |
| Urban Land Cover | Urban | 147 | 675 | 9 |
| Musk (Version 1) | Musk1 | 166 | 476 | 2 |
| Arrhythmia | Arrhythmia | 279 | 452 | 13 |
| LSVT Voice Rehabilitation | LSVT | 310 | 126 | 2 |
| Madelon | Madelon | 500 | 2600 | 2 |
| Isolet5 | Isolet5 | 617 | 1559 | 26 |
| Multiple Features | Mfeat | 649 | 2000 | 10 |
| Internet Advertisements | InterAd | 1558 | 3279 | 2 |
| DrivFace | DrivFace | 6400 | 606 | 3 |

in these two algorithms to make a fair comparison. CSO(C) and CSO(A) are two FS methods based on the CSO algorithm. Specifically, CSO(A) [39] is a wrapper-based FS method that adopts CSO as the optimizer and accuracy as the fitness function. To better compare the performance of ISBPSO and CSO, we develop the CSO(C) method by replacing the fitness function of CSO(A) with the same fitness function used in ISBPSO. CPSO is a variant of the standard CPSO algorithm. It adopts a new initialization mechanism and new updating mechanisms for *pbest* and *gbest* to improve the FS performance. In CPSO, the swarm is initialized by combining particles that select around 10% of features and particles that select over 50% of features. Although accuracy is used as the fitness function, the number of selected features is considered as well in CPSO when updating *pbest* and *gbest*. Specifically, if two particles have the same fitness value, the one that selects fewer features is considered to be better. SFS and SBS are two conventional methods that use greedy search strategies for FS.

### *4.3. Parameter Settings*

In ISBPSO and SBPSO, we set swarm size $K = 30$, maximum number of generations $T = 100$, and step parameter $L = 50$ as used in [24]. Moreover, for ISBPSO, the parameters used by the FWDI method are set as $\lambda = 0.5$ and $\sigma = 10$, and the condition parameters for updating the mask and conducting the genetic operations are set as $\mu = 0.25$ and $\varphi = 0.05$. These parameters are set based on a set of tuning experiments (see Section 5 for details). In UBPSO and QBPSO, the same swarm size $K = 30$ and maximum number of generations $T = 100$ as ISBPSO are used to make a fair comparison. The upper bound of inertia weight $\overline{w}$, lower bound of inertia weight $\underline{w}$, and fraction of generations for changing the inertia weight $\rho$ in UBPSO are set as 1, 0.4 and 0.9 as suggested in [58]. The two parameters $\theta_{max}$ and $\theta_{min}$ that determine the magnitude of the rotation angle of QBPSO are set as $0.05\pi$ and $0.01\pi$ as suggested in [59]. In CSO(A) and CSO(C), each element in a particle is set as a real value in $[0, 1]$, and a threshold parameter $\lambda = 0.5$ is used to determine if a feature is selected or not, and the parameter $\phi$ is set as 0.1 as suggested in [39]. The swarm size is set

as $K = 30$ as used in ISBPSO. CSO(C) and CSO(A) only evolve a half of the swarm at a generation, which means that each generation generate a half of new solutions compared with ISBPSO. Therefore, to make a fair comparison, the maximum number of generations of CSO(A) and CSO(C) is set as $T = 200$, twice that used in ISBPSO. This guarantees the same number of fitness function evaluations are used in ISBPSO, CSO(A) and CSO(A). In CPSO, the parameter settings are the same as those in [21]. Specifically, swarm size $K = 30$, maximum number of generations $T = 100$, $w = 0.7298$, and $c_1 = c_2 = 1.49618$. Each element in a particle of CPSO is a real value in $[0, 1]$, and a threshold parameter $\lambda = 0.6$ is used to determine if a feature is selected or not [21].

### *4.4. Experimental Configuration*

In this paper, K nearest neighbor (KNN) [60] is used as the classifier for its high performance and simplicity to test the FS methods. In KNN, we set $K = 5$ as that used in [21, 24]. SFS, SBS, and the KNN classifier are directly invoked from Waikato Environment for Knowledge Analysis package (Weka 3.8.2) [61], and the other methods are programmed in Java. All the experiments are run on PCs with a 3.4 GHz CPU and 8 GB memory.

We run two sets of experiments. The first set of experiments aims to find a proper setting for the unique parameters proposed in ISBPSO. These parameters are $\lambda$ and $\sigma$ for the initialization method (FWDI), and the condition parameters $\mu$ and $\varphi$ for updating the mask and conducting the genetic operations. Several tuning experiments are conducted on the representative datasets for obtaining a good enough parameter setting for ISBPSO. We adopt fitness value as the measure to evaluate the performance of different parameter settings. A proper parameter setting is selected for ISBPSO based on the representative datasets, and it is then used in the second set of experiments. The second set of experiments aims to compare the FS performance of ISBPSO with that of the benchmark methods on all the datasets. In the experiments, each dataset is randomly divided into a training set (70%) and a test set (30%). The training set is fed to a FS method to obtain a feature subset. The test set, which is the unseen data for the FS method, is then used to validate the classification performance of the features selected by the FS method. The results and discussions on the two sets of experiments are shown in Sections 5 and 6, respectively.

## 5. Parameter study

Compared with SBPSO, $\lambda$, $\sigma$, $\mu$, and $\varphi$ are four unique parameters added in ISBPSO. In this section, we aim to find a proper setting for the four parameters based on the experiments on the three representative datasets, Sonar, Musk1, and DrivFace, which have 60, 166, and 6400 features, respectively. As shown in Eq. (14), parameters $\lambda$ and $\sigma$ define the shape of the modified logistic function, which has potential impacts on the performance of the FWDI method of ISBPSO. Parameters $\mu$ and $\varphi$ decide the frequencies of updating the

mask and adopting the genetic operations. As there are four parameters in total, simultaneously tuning all the parameters would require numerous experiments, which is hard considering the computational resources. Note that our purpose is to obtain a proper parameter setting (not the optimal setting which is problem-dependent) for ISBPSO. Therefore, we use a two-phase strategy to design experiments to obtain a good enough (not necessarily the best) parameter setting. Since $\lambda$ and $\sigma$ are the two parameters related to the initialization process of ISBPSO. In the first phase, we examine different settings of $\lambda$ and $\sigma$ for ISBPSO. Specifically, we examine five values of $\lambda$ (i.e., 0.5, 0.6, 0.7, 0.8, and 0.9) and five values of $\sigma$ (i.e., 5, 10, 15, 20, and 25). Thus, the combination of different values on the two parameters yields 25 parameter settings. We repeat the experiment on each parameter setting 10 times, which leads to 250 runs of experiments on each dataset. Note that, in the experiments, the parameters $\mu$ and $\varphi$ are intuitively set as two constants, i.e., $\mu = 0.20$ and $\varphi = 0.05$. After we decide a proper setting for $\lambda$ and $\sigma$, we conduct the second phase tuning experiments to select a proper setting for $\mu$ and $\varphi$. We examine five values of $\mu$ (i.e., 0.10, 0.15, 0.20, 0.25, and 0.30) and four values of $\varphi$ (i.e., 0.05, 0.10, 0.15, and 0.20). Thus, the combination of different values on the two parameters yields 20 parameter settings. Similarly, the experiment on each parameter setting of $\mu$ and $\varphi$ is repeated 10 times, which leads to 200 runs of experiments on each dataset.

Figs. 4 and 5 show the boxplots of the fitness values obtained by ISBPSO with different values of $\lambda$ and $\sigma$ on the Sonar, Musk1, and DrivFace datasets. On each dataset, the fitness values from the experiments that using the given values of $\lambda$ or $\sigma$ are used to draw the boxplot. For example, for the "Sonar" boxplot in Fig. 4, the experimental results with parameter settings $\{(\lambda, \sigma)|\lambda \in \{0.5\}, \sigma \in \{5, 10, ..., 25\}\}$ are used to draw the distribution of fitness values with $\lambda = 0.5$. According to Fig. 4, the fitness value distributions on all the given values of $\lambda$ are similar on Sonar. On Musk1 and DrivFace, the fitness value distributions decline along with the increase of $\lambda$, and this trend shows more significantly on DrivFace which has a larger number of features. A possible reason for this result is that, for high dimensional data with a large number of features, a large number of redundant features are often included. Given a large $\lambda$ for the swarm initialization process of ISBPSO, the bits (features) of high weights (but may be redundant with each other) would be initialized as 1 in almost all particles. This actually decreases the swarm diversity and results in bad final optimization results. Considering the overall performance of different $\lambda$ values on the three datasets, we suggest $\lambda = 0.5$ for ISBPSO. According to Fig. 5, there is not much difference on the distributions with different values of $\sigma$ on Sonar and Musk1. Specifically, on Sonar, $\sigma = \{5, 10, 15\}$ obtain slightly better fitness values than $\sigma = \{20, 25\}$. On Musk1, the highest median of fitness values is obtained with $\sigma = 10$, and the variation of fitness values is small. This suggests that $\sigma = 10$ is the most desirable setting for $\sigma$ on Musk1. On DrivFace, $\sigma = \{10, 15, 20, 25\}$ obtain significantly better fitness values than $\sigma = 5$. The results with $\sigma = \{10, 15, 20, 25\}$ do not differ a lot, however, we can find that $\sigma = 15$ gives slightly better results than other $\sigma$ values. Considering the results with different $\sigma$ values, $\sigma = 10$ generally gives desirable fitness value results on the three datasets. Therefore, we suggest $\sigma = 10$ for ISBPSO. To sum up, according the results

in Figs. 4 and 5, we obtain a proper setting for $\lambda$ and $\sigma$, i.e., $\lambda = 0.5$ and $\sigma = 10$.

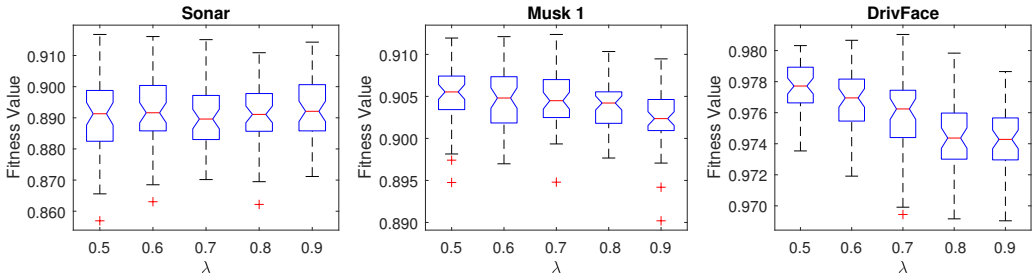


Figure 4: The boxplots of fitness values obtained by ISBPSO with different values of $\lambda$ on the representative datasets.

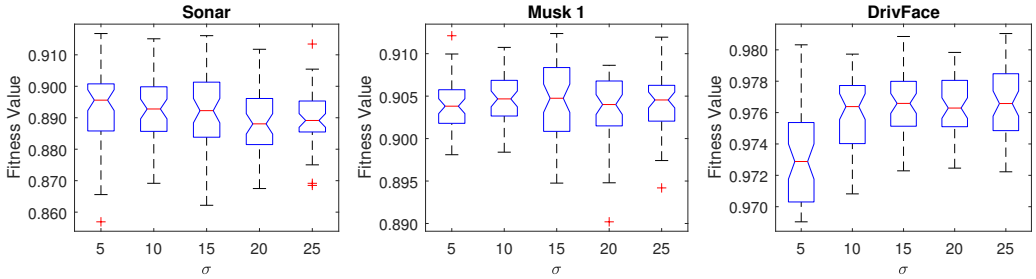


Figure 5: The boxplots of fitness values obtained by ISBPSO with different values of $\sigma$ on the representative datasets.

Figs. 6 and 7 show the boxplots of the fitness values obtained by ISBPSO with different values of $\mu$ and $\varphi$ on the Sonar, Musk1, and DrivFace datasets. According to Fig. 6, there is not much difference on the results of different values of $\mu$ on the three datasets. Specifically, $\mu = 0.25$ gives slightly better fitness values than other settings on Sonar, and $\mu = 0.10$ gives slightly better fitness values than other settings on Musk1. On DrivFace, $\mu = 0.10$ gives slight worse fitness values than other settings, and the results obtained with $\mu = \{0.15, 0.20, 0.25, 0.30\}$ do not vary too much. Generally, $\mu = 0.25$ can obtain high and stable distributions of fitness values on all the three datasets. Thus, we suggest $\mu = 0.25$ for ISBPSO. According to Fig. 7, there is not much difference on the distributions of fitness values obtained by different values of $\varphi$ on Sonar. Specifically, $\varphi = 0.10$ obtains slightly better fitness values than other settings. On Musk1 and DrivFace, we discover a trend that with the increase of $\varphi$ the fitness values decrease. It is obvious that $\varphi = 0.05$ obtains better distributions than other values of $\varphi$. A possible reason for the above results is that, on higher dimensional data, ISBPSO would be more likely to be trapped into local optima, and setting $\varphi$ as a smaller value (adopting the genetic operations more frequently) is a useful strategy to avoid the problem. Considering the results on all the three datasets, we suggest $\varphi = 0.05$ for ISBPSO. To sum up, according to the results in Figs. 6 and 7, we obtain a proper setting for $\mu$ and $\varphi$, i.e., $\mu = 0.25$ and $\varphi = 0.05$.

Based on the above analyses, we obtain a reasonable setting for the four parameters of ISBPSO, i.e., $\lambda = 0.5$, $\sigma = 10$, $\mu = 0.25$, and $\varphi = 0.05$. In the following sections, the experimental results of ISBPSO with

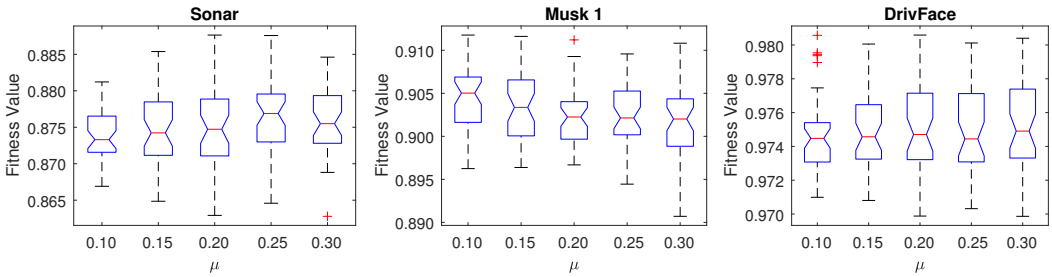


Figure 6: The boxplots of fitness values obtained by ISBPSO with different values of $\mu$ on the representative datasets.

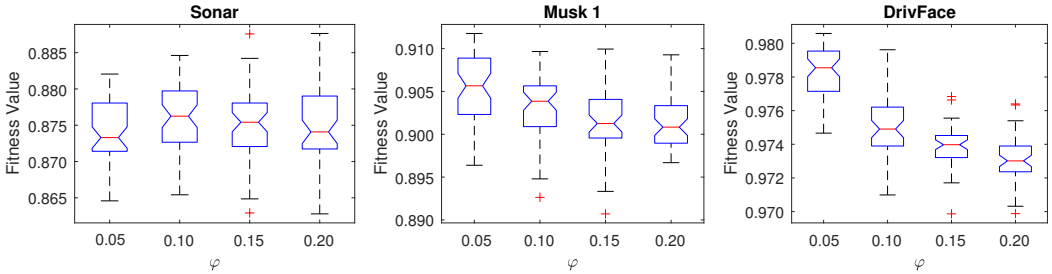


Figure 7: The boxplots of fitness values obtained by ISBPSO with different values of $\varphi$ on the representative datasets.

the suggested parameter setting are presented and analyzed to validate the proposed ISBPSO algorithm.

## 6. Comparisons on feature selection results

This section compares the effectiveness and time efficiency between ISBPSO and the benchmark methods, SBPSO, UBPSO, QBPSO, CSO(C), CSO(A), CPSO, SFS, and SBS. As the methods except for SFS and SBS are based on stochastic search strategies, we run the experiments 40 times with different running seeds for these stochastic methods to better analyze the results. Mann-Whitney U-test [62] with a significance level of 5% is used to test the statistical significance. The remainder of this section is structured as follows. First, the FS results of ISBPSO are compared with those of PSO-based benchmark methods. Second, the FS results of ISBPSO are compared with those of the two conventional methods, i.e., SFS and SBS. Third, the computation time of the methods is compared.

### *6.1. Comparisons with PSO-based methods*

Table 2 shows the comparison results between ISBPSO and PSO-based benchmark methods. In the table, "#Features (Std.)" denotes the average and standard deviation of feature subset sizes (number of selected features) over the 40 runs, "MeanA (Std.)" denotes the average and standard deviation of accuracies over the 40 runs, and "BestA" denotes the best accuracy rate of the 40 runs. Columns $S_f$ and $S_a$ show the statistical significance test results of feature subset size and accuracy, where "+" or "−" denotes ISBPSO

Table 2: Comparisons of feature selection results between ISBPSO and PSO-based methods.

| Dataset | Method | #Features (std.) | $S_f$ | BestA | MeanA (Std.) | $S_a$ | Dataset | Method | #Features (std.) | $S_f$ | BestA | MeanA (Std.) | $S_a$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ISBPSO | 11.0 (2.6) | | 85.71 | 80.32 (2.59) | | | ISBPSO | 17.3 (7.3) | | 94.87 | 87.76 (3.41) | |
| | SBPSO | 21.2 (3.3) | + | 87.30 | 80.64 (2.90) | = | | SBPSO | 136.9 (8.7) | + | 92.31 | 84.87 (3.43) | + |
| | UBPSO | 21.2 (3.0) | + | 88.89 | 81.94 (3.04) | − | | UBPSO | 141.9 (8.7) | + | 89.74 | 83.65 (3.70) | + |
| Sonar | QBPSO | 20.3 (3.5) | + | 87.30 | 82.42 (2.65) | − | LSVT | QBPSO | 133.8 (7.8) | + | 89.74 | 85.19 (3.36) | + |
| | CSO(C) | 21.4 (2.7) | + | 88.89 | 82.54 (2.74) | − | | CSO(C) | 117.3 (9.9) | + | 94.87 | 85.26 (3.80) | + |
| | CSO(A) | 26.9 (3.7) | + | 87.30 | 82.26 (2.43) | − | | CSO(A) | 154.1 (10.0) | + | 94.87 | 85.00 (3.46) | + |
| | CPSO | 25.4 (6.6) | + | 87.30 | 82.06 (2.36) | − | | CPSO | 103.6 (48.9) | + | 92.31 | 85.06 (3.57) | + |
| | ISBPSO | 18.1 (2.7) | | 79.63 | 73.38 (2.57) | | | ISBPSO | 7.2 (1.3) | | 90.51 | 86.61 (2.29) | |
| | SBPSO | 33.4 (4.9) | + | 77.78 | 69.72 (3.25) | + | | SBPSO | 222.4 (12.0) | + | 64.10 | 61.47 (1.36) | + |
| | UBPSO | 31.8 (3.8) | + | 76.85 | 69.47 (2.89) | + | | UBPSO | 224.6 (12.1) | + | 65.39 | 61.91 (1.87) | + |
| Movement | QBPSO | 27.4 (4.1) | + | 77.78 | 70.60 (2.74) | + | Madelon | QBPSO | 213.6 (11.6) | + | 66.41 | 63.03 (1.87) | + |
| | CSO(C) | 29.6 (4.2) | + | 76.85 | 71.13 (2.66) | + | | CSO(C) | 213.9 (10.2) | + | 66.92 | 62.94 (1.79) | + |
| | CSO(A) | 46.1 (4.7) | + | 73.15 | 69.14 (2.11) | + | | CSO(A) | 230.6 (12.0) | + | 66.92 | 62.32 (1.65) | + |
| | CPSO | 49.0 (14.5) | + | 74.07 | 68.94 (2.09) | + | | CPSO | 214.7 (91.2) | + | 67.44 | 60.38 (2.73) | + |
| | ISBPSO | 14.3 (3.0) | | 55.07 | 50.73 (2.06) | | | ISBPSO | 130.6 (21.0) | | 90.17 | 88.43 (1.02) | |
| | SBPSO | 25.6 (3.2) | + | 53.97 | 50.53 (1.36) | = | | SBPSO | 268.1 (11.0) | + | 87.61 | 85.72 (1.03) | + |
| | UBPSO | 27.1 (3.7) | + | 53.42 | 50.93 (1.56) | = | | UBPSO | 275.8 (9.3) | + | 87.61 | 85.81 (0.97) | + |
| Hillvalley | QBPSO | 21.9 (3.8) | + | 54.80 | 50.86 (1.50) | = | Isolet5 | QBPSO | 252.7 (10.6) | + | 88.46 | 87.07 (0.91) | + |
| | CSO(C) | 27.1 (3.5) | + | 54.52 | 50.73 (1.65) | = | | CSO(C) | 261.0 (11.4) | + | 88.89 | 86.60 (0.93) | + |
| | CSO(A) | 43.8 (5.1) | + | 53.97 | 50.55 (1.24) | = | | CSO(A) | 288.8 (11.4) | + | 88.03 | 86.48 (0.80) | + |
| | CPSO | 19.4 (5.8) | + | 54.80 | 51.02 (1.86) | = | | CPSO | 350.2 (51.6) | + | 86.11 | 83.24 (1.27) | + |
| | ISBPSO | 17.7 (3.4) | | 86.77 | 83.74 (1.62) | | | ISBPSO | 30.4 (4.5) | | 99.33 | 98.19 (0.42) | |
| | SBPSO | 48.2 (5.7) | + | 85.78 | 82.08 (1.56) | + | | SBPSO | 250.9 (12.0) | + | 98.67 | 97.79 (0.33) | + |
| | UBPSO | 51.3 (5.7) | + | 84.80 | 81.48 (1.52) | + | | UBPSO | 253.2 (10.9) | + | 98.33 | 97.86 (0.30) | + |
| Urban | QBPSO | 42.9 (5.8) | + | 84.80 | 82.19 (1.44) | + | Mfeat | QBPSO | 223.3 (12.0) | + | 98.50 | 97.87 (0.32) | + |
| | CSO(C) | 46.6 (4.8) | + | 85.29 | 81.92 (1.32) | + | | CSO(C) | 211.3 (11.0) | + | 98.50 | 97.84 (0.32) | + |
| | CSO(A) | 67.2 (6.1) | + | 85.29 | 81.84 (1.31) | + | | CSO(A) | 327.9 (14.4) | + | 98.33 | 97.95 (0.19) | + |
| | CPSO | 61.8 (24.3) | + | 84.31 | 81.43 (1.45) | + | | CPSO | 400.6 (53.7) | + | 98.33 | 97.91 (0.24) | + |
| | ISBPSO | 37.4 (6.2) | | 91.67 | 85.92 (2.96) | | | ISBPSO | 47.4 (18.9) | | 98.27 | 97.35 (0.56) | |
| | SBPSO | 67.2 (5.7) | + | 92.36 | 87.73 (2.19) | − | | SBPSO | 695.5 (15.9) | + | 97.76 | 97.15 (0.39) | = |
| | UBPSO | 64.0 (5.5) | + | 93.06 | 87.76 (2.51) | − | | UBPSO | 704.2 (17.3) | + | 97.76 | 97.28 (0.33) | = |
| Musk1 | QBPSO | 62.4 (6.1) | + | 93.06 | 88.54 (2.56) | − | InterAd | QBPSO | 662.5 (14.8) | + | 97.97 | 97.38 (0.30) | = |
| | CSO(C) | 63.3 (5.1) | + | 91.67 | 87.38 (2.19) | − | | CSO(C) | 621.2 (15.8) | + | 97.87 | 97.19 (0.38) | = |
| | CSO(A) | 82.5 (7.2) | + | 91.67 | 88.23 (2.01) | − | | CSO(A) | 774.5 (17.6) | + | 97.76 | 97.28 (0.33) | = |
| | CPSO | 72.1 (23.9) | + | 89.58 | 84.91 (2.77) | = | | CPSO | 789.5 (120.6) | + | 97.76 | 96.82 (0.40) | + |
| | ISBPSO | 14.8 (4.4) | | 73.72 | 71.13 (1.89) | | | ISBPSO | 309.0 (44.2) | | 98.91 | 97.08 (0.80) | |
| | SBPSO | 107.3 (8.0) | + | 65.69 | 62.70 (1.63) | + | | SBPSO | 2972.2 (38.0) | + | 96.17 | 95.23 (0.37) | + |
| | UBPSO | 112.2 (9.4) | + | 66.42 | 62.90 (1.63) | + | | UBPSO | 3013.4 (36.7) | + | 96.17 | 95.33 (0.39) | + |
| Arrhythmia | QBPSO | 100.2 (8.6) | + | 67.88 | 63.76 (2.04) | + | DrivFace | QBPSO | 2866.7 (40.6) | + | 95.63 | 95.20 (0.36) | + |
| | CSO(C) | 94.8 (7.6) | + | 66.42 | 63.18 (1.86) | + | | CSO(C) | 2803.6 (46.5) | + | 96.72 | 95.30 (0.49) | + |
| | CSO(A) | 127.3 (7.7) | + | 67.15 | 62.68 (1.76) | + | | CSO(A) | 3197.9 (41.6) | + | 95.63 | 95.29 (0.39) | + |
| | CPSO | 53.1 (15.1) | + | 69.34 | 64.82 (2.38) | + | | CPSO | 1305.0 (449.8) | + | 96.72 | 95.37 (0.63) | + |

obtains significantly better or worse results than the compared method, and "=" denotes that there is no (statistically) significant difference.

Compared with SBPSO, ISBPSO obtains better FS results on most datasets. ISBPSO obtains significantly better results on both feature subset size and accuracy on 8 datasets and obtains significantly smaller feature subset sizes with similar accuracies on 3 datasets, Sonar, Hillvalley, and InterAd. ISBPSO only obtains a worse FS result than SBPSO on Musk1, where ISBPSO obtains a significantly lower accuracy rate with fewer features. This denotes that ISBPSO may eliminate some informative features, which lowers

the accuracies in some cases. Additionally, for the higher dimensional datasets (Arrhythmia to DrivFace), ISBPSO selects substantially fewer features than SBPSO. For instance, on Arrhythmia, ISBPSO only selects 14.8 features on average over the 40 runs, whereas SBPSO selects 107.3 features.

Compared with the two BPSO variants (UBPSO and QBPSO) and the two CSO algorithms (CSO(C) and CSO(A)), ISBPSO obtains better FS results on most datasets. Specifically, ISBPSO obtains both better accuracy and feature subset size results on 8 datasets and obtains similar accuracies with fewer features on 2 datasets, Hillvalley and InterAd. ISBPSO obtains worse results than these benchmark methods on 2 datasets, Sonar and Musk1, since it obtains significantly lower accuracies. Additionally, ISBPSO selects substantially fewer features compared with these methods. The above results denote that ISBPSO performs more effectively than UBPSO, QBPSO, CSO(C), and CSO(A) in most cases.

Compared with CPSO, ISBPSO obtains better FS results on most datasets. Specifically, ISBPSO obtains both significantly better feature subset size and accuracy results on 9 datasets and obtains similar accuracies with fewer features on 2 datasets, Hillvalley and Musk1. ISBPSO performs worse than CPSO on Sonar, where ISBPSO obtains a significantly lower accuracy rate than CPSO. Moreover, in terms of the number of selected features, ISBPSO selects substantially fewer features than CPSO on all the datasets.

To sum up, the proposed ISBPSO algorithm obtains higher accuracies while selecting fewer features compared with the PSO-based FS methods in most cases. Among these methods, ISBPSO, SBPSO, UBPSO, QBPSO, and CSO(C) adopt the same fitness function defined in Eq. (10) as ISBPSO. The fact that ISBPSO outperforms these methods with the same fitness function indicates that ISBPSO obtains better optimization results than these PSO algorithms. Compared with SBPSO, the novel mechanisms added in ISBPSO are the FWDI method, the DBM strategy, and mutation operations. Thus, we can conclude that these added mechanisms are effective for improving the search performance of ISBPSO. Moreover, the effects of different fitness functions can be found comparing between CSO(C) and CSO(A), since the only difference between the two methods is that different fitness functions are used. The two CSO algorithms generally obtain similar accuracies on the datasets, while CSO(C) tends to select fewer features than CSO(A) on the datasets. This finding indicates that using the combined fitness function in Eq. (10) (adopted by CSO(C)) can obtain a better feature reduction performance than adopting accuracy as the fitness function (adopted by CSO(A)). Although ISBPSO obtains better FS results in most cases, ISBPSO performs worse than the compared methods in some cases. On Sonar and Musk1, ISBPSO obtains significantly lower accuracies than most of the benchmark PSO-based FS methods. Two possible reasons can explain this result. First, ISBPSO adopts the feature weighting results of MI to initialize particles. However, MI only measures the correlation level between a feature and the class label, which neglects the correlation among the features. Thus, the weighting results of MI for features are not accurate, which leads to the poor performance of the initialized swarm of ISBPSO. Second, in this paper, we use the DBM strategy to reduce the search space of ISBPSO. This strategy may incorrectly mask some key bits (features) during the evolutionary process of ISBPSO,

and then yields worse final FS results.

### *6.2. Comparisons with SFS and SBS*

Table 3 shows the FS results of ISBPSO, SFS, and SBS. The results of SBS on the four highest dimensional datasets, i.e., Isolet5, Mfeat, InterAd, and DrivFace, are not available (NA), since the experiments on these datasets do not finish within four days. The reason is that the high dimensionality of data significantly increases the computation time of SBS. On the one hand, the number of search attempts by SBS is significantly increased with the increase of features. On the other hand, the computation time of each feature subset evaluation of SBS significantly increases since it starts from a full set of features.

Table 3: Comparisons of feature selection results between ISBPSO and SFS/SBS.

| Dataset | Method | #Features (std.) | $S_f$ | BestA | MeanA (Std.) | $S_a$ | Dataset | Method | #Features (std.) | $S_f$ | BestA | MeanA (Std.) | $S_a$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ISBPSO | 11.0 (2.6) | | 85.71 | 80.32 (2.59) | | | ISBPSO | 17.3 (7.3) | | 94.87 | 87.76 (3.41) | |
| Sonar | SFS | 7.0 | − | 79.36 | 79.36 | = | LSVT | SFS | 4.0 | − | 76.92 | 76.92 | + |
| | SBS | 42.0 | + | 80.95 | 80.95 | = | | SBS | 55.0 | + | 84.61 | 84.61 | + |
| | ISBPSO | 18.1 (2.7) | | 79.63 | 73.38 (2.57) | | | ISBPSO | 7.2 (1.3) | | 90.51 | 86.61 (2.29) | |
| Movement | SFS | 15.0 | − | 74.07 | 74.07 | − | Madelon | SFS | 12.0 | + | 90.90 | 90.90 | − |
| | SBS | 77.0 | + | 68.52 | 68.52 | + | | SBS | 487.0 | + | 59.36 | 59.36 | + |
| | ISBPSO | 14.3 (3.0) | | 55.07 | 50.73 (2.06) | | | ISBPSO | 130.6 (21.0) | | 90.17 | 88.43 (1.02) | |
| Hillvalley | SFS | 5.0 | − | 51.23 | 51.23 | − | Isolet5 | SFS | 35.0 | − | 85.68 | 85.68 | + |
| | SBS | 94.0 | + | 49.86 | 49.86 | + | | SBS | NA | NA | NA | NA | NA |
| | ISBPSO | 17.7 (3.4) | | 86.77 | 83.74 (1.62) | | | ISBPSO | 30.4 (4.5) | | 99.33 | 98.19 (0.42) | |
| Urban | SFS | 9.0 | − | 80.88 | 80.88 | + | Mfeat | SFS | 11.0 | − | 97.67 | 97.67 | + |
| | SBS | 113.0 | + | 78.92 | 78.92 | + | | SBS | NA | NA | NA | NA | NA |
| | ISBPSO | 37.4 (6.2) | | 91.67 | 85.92 (2.96) | | | ISBPSO | 47.4 (18.9) | | 98.27 | 97.35 (0.56) | |
| Musk1 | SFS | 10.0 | − | 79.86 | 79.86 | + | InterAd | SFS | 14.0 | − | 96.65 | 96.65 | + |
| | SBS | 122.0 | + | 80.56 | 80.56 | + | | SBS | NA | NA | NA | NA | NA |
| | ISBPSO | 14.8 (4.4) | | 73.72 | 71.13 (1.89) | | | ISBPSO | 309.0 (44.2) | | 98.91 | 97.08 (0.80) | |
| Arrhythmia | SFS | 19.0 | + | 70.80 | 70.80 | + | DrivFace | SFS | 9.0 | − | 93.99 | 93.99 | + |
| | SBS | 110.0 | + | 61.31 | 61.31 | + | | SBS | NA | NA | NA | NA | NA |

Compared with SFS, ISBPSO obtains higher accuracies with more features in most cases. Specifically, ISBPSO obtains significantly higher accuracies on 8 of the 12 datasets. According to the number of selected features, SFS selects fewer features than ISBPSO on all the datasets. These results indicate that SFS does not select some possible key features, which makes SFS obtain lower accuracies than ISBPSO in most cases. A possible reason is that the greedy search strategy of SFS traps it into local optima, which limits the method to further select more informative features. To sum up, as accuracy is often the main consideration for a classification task, ISBPSO performs better than SFS from this perspective.

Compared with SBS, ISBPSO obtains both significantly better accuracy and feature subset size results on 7 of the 8 available datasets. On one dataset (Sonar), ISBPSO obtains a similar accuracy rate with fewer features than SBS. These results indicate that ISBPSO is more effective than SBS for FS. In terms of the number of selected features, it is obvious that ISBPSO generally selects substantially fewer features than SBS. SBS shows poor feature reduction performance because it eliminates uninformative features in a backward way and the greedy search strategy used could easily trap it into local optima.

*6.3. Computation time*

Table 4 shows the computation time (in minutes) of the FS methods per experimental run on each dataset, where the average computation time over the 40 runs is shown for each PSO-based FS method. According to the table, ISBPSO is not as time-efficient as SFS and SBS on datasets with a relatively smaller number of features. Specifically, ISBPSO cost more computation time than SFS on 6 datasets (from Sonar to LSVT except for Arrhythmia), and cost more computation time than SBS on 3 datasets (from Sonar to Hillvalley). The gaps of computation time between ISBPSO and these two methods decrease dramatically with the increase of data dimensionality. On some high dimensional datasets (Madelon to InterAd), ISBPSO performs even more time-efficient than SFS. One possible reason for the above result is that the number of search attempts of SFS (or SBS) increases dramatically with the increase of data dimensionality, whereas the number of generations of ISBPSO does not change.

Table 4: Comparisons of computation time (m).

| Dataset | ISBPSO | SBPSO | UBPSO | QBPSO | CSO(C) | CSO(A) | CPSO | SFS | SBS |
|---|---|---|---|---|---|---|---|---|---|
| Sonar | 0.37 | 0.66 | 0.75 | 0.68 | 0.50 | 0.59 | 0.55 | 0.04 | 0.33 |
| Movement | 1.24 | 1.85 | 2.02 | 1.83 | 1.39 | 1.70 | 1.71 | 0.32 | 0.97 |
| Hillvalley | 3.65 | 4.78 | 4.99 | 4.34 | 4.85 | 5.44 | 4.21 | 0.45 | 2.30 |
| Urban | 3.02 | 7.27 | 9.01 | 7.74 | 6.42 | 7.68 | 6.78 | 0.74 | 20.98 |
| Musk1 | 3.60 | 5.76 | 6.59 | 6.24 | 5.19 | 6.07 | 4.93 | 0.55 | 19.50 |
| Arrhythmia | 1.81 | 6.99 | 8.08 | 7.74 | 6.13 | 6.89 | 3.36 | 1.98 | 138.34 |
| LSVT | 0.34 | 1.10 | 1.20 | 1.18 | 0.93 | 0.97 | 0.71 | 0.07 | 18.40 |
| Madelon | 24.00 | 567.08 | 444.80 | 419.12 | 517.67 | 609.10 | 463.77 | 37.76 | 2803.17 |
| Isolet5 | 67.13 | 161.98 | 170.92 | 151.56 | 175.93 | 187.63 | 236.45 | 105.20 | NA |
| Mfeat | 33.84 | 314.10 | 279.45 | 248.63 | 252.37 | 451.82 | 467.20 | 34.17 | NA |
| InterAd | 92.51 | 1376.00 | 1408.22 | 1278.32 | 1033.45 | 1496.10 | 1189.82 | 134.89 | NA |
| DrivFace | 34.69 | 317.82 | 253.92 | 283.07 | 308.20 | 329.68 | 124.79 | 31.59 | NA |

Compared with the PSO algorithms, ISBPSO takes less computation time in all cases. The gaps of computation time between ISBPSO and the PSO algorithms expand dramatically with the increase of data dimensionality. For the 7 high dimensional datasets with more than 200 features (Arrhythmia to DrivFace), ISBPSO requires substantially less computation time than the benchmark PSO algorithms. Specifically, ISBPSO saves around 46%, 52%, 94%, 56%, 86%, 91%, and 72% computation time than the second efficient PSO algorithm on these datasets.

The above results show that the proposed ISBPSO algorithm is more time-efficient than benchmark PSO algorithms. The following reason can explain why ISBPSO is a more efficient PSO algorithm for FS. All the methods compared in the experiments are based on the wrapper framework. Evaluating the fitness values is time-consuming in a wrapper-based FS method as a learning algorithm is involved in each evaluation, and the evaluation time of a feature subset (solution) is highly related to the size of the feature subset. ISBPSO shows good performance to reduce features during the evolutionary process. So, the feature subsets to be evaluated during the evolutionary process of ISBPSO are generally smaller than those of the benchmark PSO algorithms. This substantially reduces the computation time of ISBPSO for fitness evaluations.

To illustrate the feature reduction performance of ISBPSO, we draw the feature number curves of the PSO algorithms on each dataset in Fig. 8. Specifically, the average number of selected features of all the particles at each generation is used to draw the curve for each PSO algorithm. Since 40 runs of experiments are conducted for each algorithm, the average curve over the 40 runs is shown in Fig. 8. Note that the maximum number of generations of the two CSO algorithms is twice that of other PSO algorithms since they only evolve a half of the particles at a generation. Thus, we convert the curves of the two CSO algorithms from generations $0, 1, ..., 200$ to generations $0, 1(2), ..., 100(200)$ for a fair comparison, i.e., choosing every two generations from generations 0 to 200 to plot the curves to reduce the number of points from 200 to 100. As a result, the actual number of generations for CSO(C) and CSO(A) is twice the number shown in x-axes of Fig. 8. According to the figure, the number of selected features by the swarm of ISBSPO during the evolutionary process is substantially smaller than that of the benchmark PSO algorithms. The main reason for this pattern is that the proposed ISBPSO algorithm uses the FWDI method for initialization instead of the RI strategy used by benchmark PSO algorithms. The FWDI method adopts the feature weighting results of MI to initialize the swarm, where the bits (features) with low weights would have low probabilities to be initialized as 1. So, the average number of selected features by the initial swarm of ISBPSO is substantially reduced. Moreover, we discover a trend that the feature number curves of ISBPSO quickly increase in the early phase of iterations and then converge to a stable level during the evolutionary process. This is because that the MI measure used by FWDI can not accurately weigh all the features, since the correlation among multiple features can not be captured by MI. Some complementary features can be improperly assigned to low weight values, which lowers the probabilities of these bits to be initialized as 1. During the early evolutionary process of ISBPSO, the possible key features (bits) would be gradually selected by particles, which yields an increasing trend in the feature number curve of ISBPSO. After that, the average number of selected features by the swarm gradually convergences to a proper number determined by the ISBPSO during the evolutionary process. To sum up, as shown in Fig. 8, the feature subsets to be evaluated during the evolutionary process of ISBPSO are much smaller than the benchmark PSO algorithms due to a better feature reduction performance. Therefore, less computation time is generally needed by ISBPSO.

## 7. Further analysis

Comparisons in Section 6 have shown that the proposed ISBPSO algorithm is effective and efficient for FS. In Section 7.1, we will further analyze the performance of the three new mechanisms (the FWDI method, DBM strategy, and genetic operations) adopted in ISBPSO in more detail. To test the effectiveness of each mechanism, we propose three ISBPSO variants named ISBPSO-IN, ISBPSO-DBM, and ISBPSO-GO. In each variant, one of the proposed mechanisms is eliminated. More specifically, ISBPSO-IN adopts the RI method instead of the FWDI method, ISBPSO-DBM does not use the DBM strategy, and ISBPSO-GO

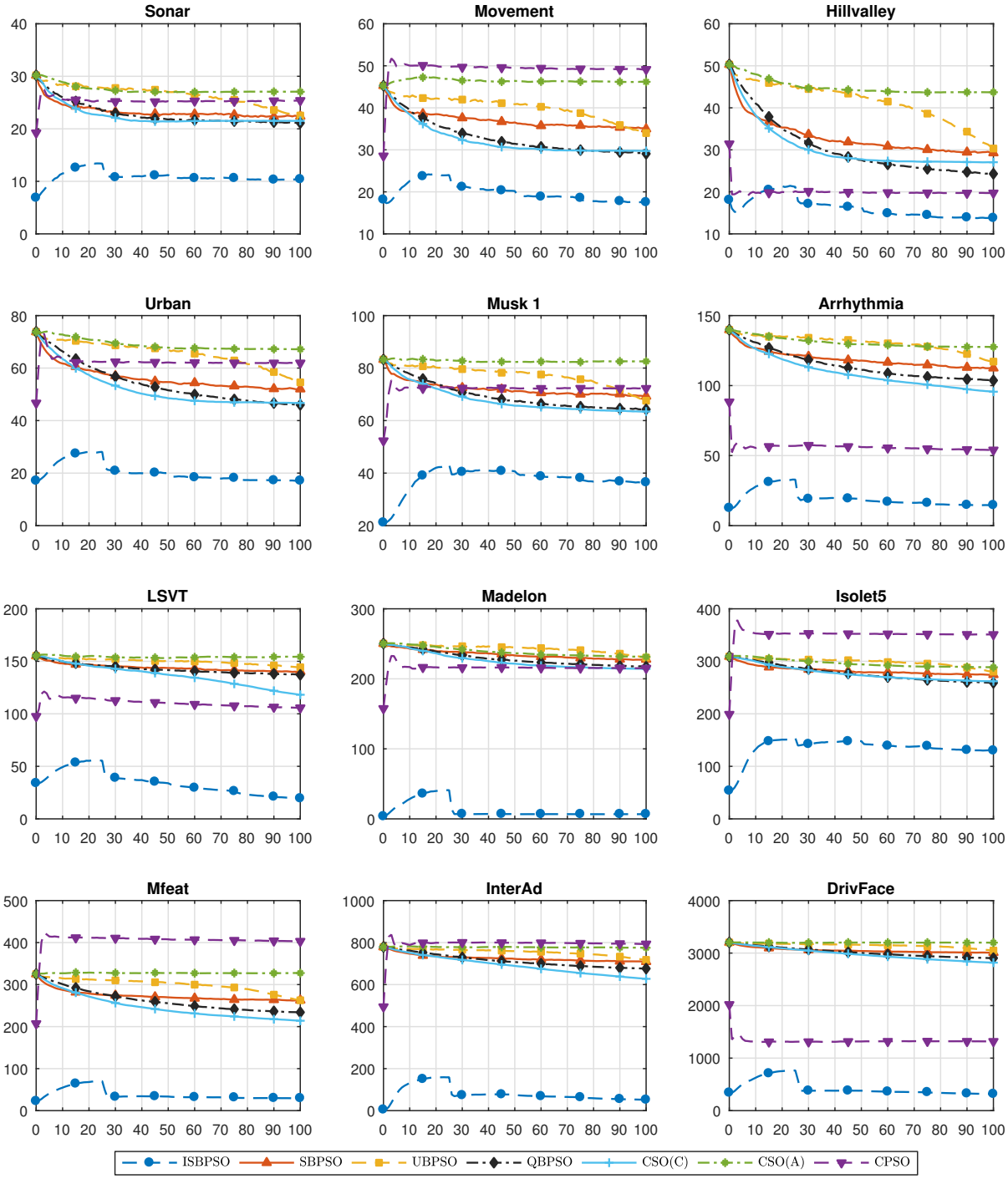


Figure 8: Feature number curves of the PSO algorithms on the datasets. The x-axis denotes the number of generations and y-axis denotes the average number of selected features by the swarm over the 40 runs.

does not conduct the genetic operations. Moreover, the convergence property of ISBPSO will be studied in Section 7.2.

### 7.1. Performance of new mechanisms in ISBPSO

Table 5 shows the comparison results between ISBPSO and its variants. In the table, the averages and standard deviations of fitness values (%) over the 40 runs are listed in column "MeanF (Std.)". The statistical significance test results from Mann-Whitney U-test are listed in column "$S_{fit}$", where "+", "−", or "=" denotes that ISBPSO($\alpha$) obtains significantly higher, lower, or similar fitness values compared with the variants. The computation time of each algorithm is listed in the "Time (m)" column.

Table 5: Comparisons between ISBPSO and its variants.

| Dataset | Method | MeanF (Std.) | $S_{fit}$ | #Features (Std.) | $S_f$ | BestA | MeanA (Std.) | $S_a$ | Time (m) |
|---|---|---|---|---|---|---|---|---|---|
| Sonar | ISBPSO | 88.10 (1.29) | | 11.0 (2.6) | | 85.71 | 80.32 (2.59) | | 0.37 |
| | ISBPSO-IN | 87.80 (1.83) | = | 20.1 (4.5) | + | 88.89 | 82.18 (3.22) | − | 0.58 |
| | ISBPSO-DBM | 88.05 (1.34) | = | 11.2 (2.9) | = | 85.71 | 80.48 (2.34) | = | 0.45 |
| | ISBPSO-GO | 88.62 (1.20) | − | 11.7 (2.5) | = | 88.89 | 80.91 (2.97) | = | 0.44 |
| Movement | ISBPSO | 74.70 (0.77) | | 18.1 (2.7) | | 79.63 | 73.38 (2.57) | | 1.24 |
| | ISBPSO-IN | 72.53 (1.36) | + | 25.0 (5.2) | + | 76.85 | 70.67 (2.82) | + | 1.56 |
| | ISBPSO-DBM | 73.61 (0.75) | + | 19.9 (3.6) | + | 78.70 | 72.64 (2.31) | = | 1.39 |
| | ISBPSO-GO | 74.21 (0.83) | + | 19.3 (2.7) | = | 78.70 | 72.85 (2.57) | = | 1.40 |
| Hillvalley | ISBPSO | 62.43 (0.56) | | 14.3 (3.0) | | 55.07 | 50.73 (2.06) | | 3.65 |
| | ISBPSO-IN | 62.15 (0.73) | = | 19.8 (4.2) | + | 54.52 | 50.80 (1.51) | = | 3.82 |
| | ISBPSO-DBM | 61.32 (0.57) | + | 17.4 (3.0) | + | 53.97 | 50.20 (1.66) | = | 4.37 |
| | ISBPSO-GO | 62.26 (0.64) | = | 15.8 (2.5) | + | 53.97 | 50.72 (1.62) | = | 4.12 |
| Urban | ISBPSO | 89.50 (0.55) | | 17.7 (3.4) | | 86.77 | 83.74 (1.62) | | 3.02 |
| | ISBPSO-IN | 86.69 (1.37) | + | 33.9 (7.5) | + | 85.78 | 82.07 (1.56) | + | 6.18 |
| | ISBPSO-DBM | 88.16 (0.50) | + | 21.2 (3.8) | + | 87.26 | 82.98 (1.94) | + | 3.97 |
| | ISBPSO-GO | 89.07 (0.58) | + | 21.3 (4.2) | + | 86.77 | 83.54 (1.94) | = | 3.75 |
| Musk1 | ISBPSO | 91.38 (0.62) | | 37.4 (6.2) | | 91.67 | 85.92 (2.96) | | 3.60 |
| | ISBPSO-IN | 89.84 (1.05) | + | 53.9 (6.9) | + | 93.06 | 88.00 (2.33) | − | 5.14 |
| | ISBPSO-DBM | 89.75 (0.58) | + | 38.5 (6.7) | = | 91.67 | 86.28 (2.73) | = | 3.66 |
| | ISBPSO-GO | 90.17 (0.58) | + | 44.4 (4.7) | + | 92.36 | 86.48 (3.07) | = | 4.11 |
| Arrhythmia | ISBPSO | 76.86 (0.98) | | 14.8 (4.4) | | 73.72 | 71.13 (1.89) | | 1.81 |
| | ISBPSO-IN | 69.55 (1.56) | + | 78.1 (11.4) | + | 69.34 | 65.22 (1.84) | + | 6.34 |
| | ISBPSO-DBM | 75.76 (0.81) | + | 16.7 (4.2) | + | 72.99 | 70.35 (1.85) | + | 2.27 |
| | ISBPSO-GO | 76.45 (1.03) | = | 31.6 (6.5) | + | 74.45 | 70.46 (1.68) | + | 2.84 |
| LSVT | ISBPSO | 93.55 (0.83) | | 17.3 (7.3) | | 94.87 | 87.76 (3.41) | | 0.34 |
| | ISBPSO-IN | 86.45 (1.19) | + | 108.3 (10.7) | + | 92.31 | 85.51 (3.61) | + | 0.95 |
| | ISBPSO-DBM | 92.00 (1.03) | + | 35.0 (6.5) | + | 94.87 | 88.97 (3.77) | = | 0.46 |
| | ISBPSO-GO | 92.22 (1.14) | + | 46.5 (8.9) | + | 97.44 | 90.00 (3.97) | − | 0.48 |
| Madelon | ISBPSO | 86.97 (2.24) | | 7.2 (1.3) | | 90.51 | 86.61 (2.29) | | 24.00 |
| | ISBPSO-IN | 68.69 (0.97) | + | 176.4 (16.5) | + | 66.80 | 63.96 (1.49) | + | 393.20 |
| | ISBPSO-DBM | 88.06 (1.87) | − | 8.3 (1.1) | + | 91.03 | 88.07 (2.04) | − | 61.46 |
| | ISBPSO-GO | 87.26 (2.48) | = | 7.9 (1.3) | + | 91.41 | 86.82 (2.29) | = | 27.94 |
| Isolet5 | ISBPSO | 88.57 (0.54) | | 130.6 (21.0) | | 90.17 | 88.43 (1.02) | | 67.13 |
| | ISBPSO-IN | 85.91 (0.65) | + | 209.9 (16.2) | + | 88.67 | 86.98 (0.90) | + | 135.09 |
| | ISBPSO-DBM | 87.16 (0.55) | + | 139.1 (22.8) | = | 89.74 | 87.71 (0.91) | + | 74.24 |
| | ISBPSO-GO | 86.21 (0.51) | + | 174.1 (18.2) | + | 89.53 | 87.44 (1.10) | + | 80.86 |
| Mfeat | ISBPSO | 98.70 (0.16) | | 30.4 (4.5) | | 99.33 | 98.19 (0.42) | | 33.84 |
| | ISBPSO-IN | 95.91 (0.27) | + | 175.0 (17.2) | + | 98.33 | 97.77 (0.38) | + | 235.22 |
| | ISBPSO-DBM | 98.28 (0.17) | + | 31.3 (4.3) | = | 99.00 | 98.01 (0.50) | = | 63.53 |
| | ISBPSO-GO | 98.23 (0.18) | + | 49.0 (8.5) | + | 98.83 | 98.22 (0.32) | = | 47.89 |
| InterAd | ISBPSO | 97.22 (0.18) | | 47.4 (18.9) | | 98.27 | 97.35 (0.56) | | 92.51 |
| | ISBPSO-IN | 94.37 (0.37) | + | 509.2 (53.9) | + | 97.87 | 97.24 (0.38) | = | 1158.52 |
| | ISBPSO-DBM | 96.95 (0.13) | + | 54.5 (12.4) | = | 98.17 | 97.19 (0.49) | = | 176.38 |
| | ISBPSO-GO | 96.60 (0.14) | + | 109.8 (29.1) | + | 98.07 | 97.12 (0.45) | = | 176.00 |
| DrivFace | ISBPSO | 97.47 (0.19) | | 309.0 (44.2) | | 98.91 | 97.08 (0.80) | | 34.69 |
| | ISBPSO-IN | 93.15 (0.37) | + | 2337.2 (203.8) | + | 96.72 | 95.40 (0.54) | + | 227.55 |
| | ISBPSO-DBM | 96.97 (0.13) | + | 358.4 (26.6) | + | 98.36 | 97.24 (0.65) | = | 57.21 |
| | ISBPSO-GO | 96.73 (0.13) | + | 488.6 (65.3) | + | 98.36 | 97.02 (0.72) | = | 49.73 |

In terms of the fitness value, the numbers of win/draw/loss times of ISBPSO in comparison with ISBPSO-IN, ISBPSO-DBM, and ISBPSO-GO are 10/2/0, 10/1/1, and 8/3/1, which means that ISBPSO obtains similar or better fitness values than the variants in almost all cases, i.e., 34 out of the 36 comparisons. To sum up, ISBPSO can generally obtain better fitness values than its variants. Thus, we can conclude that

the new mechanisms proposed for ISBPSO are effective for improving the search performance.

In terms of the feature subset size, ISBPSO obtains significantly better results in almost all cases. Specifically, ISBPSO selects significantly fewer features than ISBPSO-IN/ISBPSO-DBM/ISBPSO-GO on 12/7/10 of the 12 datasets and selects a similar number of features in other cases. This denotes that the feature reduction performance of ISBPSO is improved by adopting each of the three new mechanisms.

In terms of accuracy, ISBPSO obtains significantly higher or similar accuracies in most cases compared with the three variants. ISBPSO obtains significantly worse accuracies in 4 cases. Specifically, it obtains lower accuracies than ISBPSO-IN on Sonar and Musk1, obtains a lower accuracy rate than ISBPSO-DBM on Madelon, and obtains a lower accuracy rate than ISBPSO-GO on LSVT. A possible reason is that ISBPSO may incorrectly reduce some critical features seeing that ISBPSO selects fewer features than the variants on Sonar, Musk1, Madelon, and LSVT. Considering accuracy as the priority measure and feature subset size as the second priority measure, the numbers of win/draw/loss times comparing ISBPSO with ISBPSO-IN, ISBPSO-DBM, and ISBPSO-GO are 10/0/2, 7/4/1, and 9/2/1. This shows that ISBPSO obtains better overall FS results than the variants in most cases.

According to the computation time results of ISBPSO and the variants shown in Table 5, ISBPSO generally requires less computation time than the three variants. The computation time of ISBPSO-DBM and ISBPSO-GO is only slightly more than ISBPSO, while the computation time of ISBPSO-IN is much more than ISBPSO. As we have discussed in Section 6.3, the computation time of these PSO-based wrapper methods is highly related to the feature reduction performance. Moreover, as shown in Table 5, the final number of selected features of ISBPSO-IN on each dataset is much larger than that of ISBPSO, ISBPSO-DBM, and ISBPSO-GO. Therefore, we can conclude that by using the FWDI method for initialization, the feature reduction performance of ISBPSO is significantly improved.

The above comparisons between ISBPSO and its variants show that each of the three new mechanisms used in ISBPSO is effective for improving its search performance, which in turn results in better FS results.

### *7.2. Convergence analysis*

Fig. 9 shows the convergence curves of ISBPSO and benchmark PSO algorithms. Each curve is drawn from the average of the 40 best fitness values obtained at each generation over the 40 runs. The convergence curves of SBPSO, UBPSO, QBPSO, and CSO(C) are drawn to compare with those of ISBPSO because these PSO algorithms adopt the same fitness function as ISBPSO. Noted that, similar to Fig. 8, the convergence curves of CSO(C) are transformed from generations $0, 1, ..., 200$ to generations $0, 1, ..., 100$, since CSO(C) evolves a half of particles compared with other PSO algorithms. Moreover, we show the convergence curves of ISBPSO-IN in Fig. 9. Therefore, the effect of the FWDI method can be discovered by comparing ISBPSO with ISBPSO-IN, and effects of the DBM strategy and the genetic operations can be found by comparing ISBPSO-IN with SBPSO. The convergence curves of ISBPSO-DBM and ISBPSO-GO are between the curves

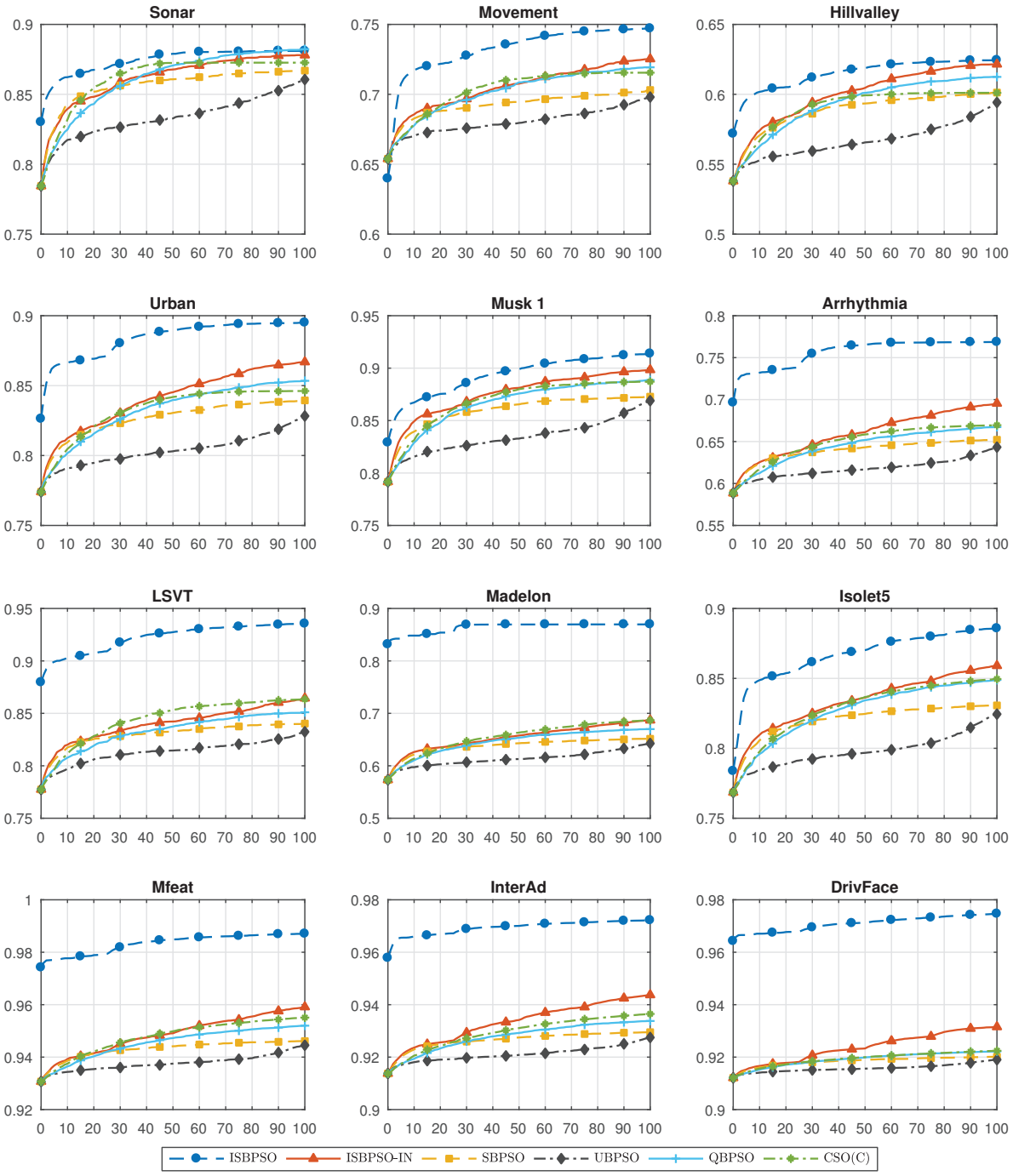


Figure 9: Convergence curves of the PSO algorithms on each dataset. The x-axis denotes the number of generations and y-axis denotes the average fitness value over the 40 runs.

of ISBPSO and ISBPSO-IN. We do not show the curves of these two ISBPSO variants to avoid the figures being too full to be seen clearly.

According to Fig. 9, ISBPSO obtains significantly higher convergence curves than the benchmark algorithms on all the datasets, showing that ISBPSO has a good convergence property. The effect of FWDI can be clearly found in the figure. On most datasets, the fitness values obtained by ISBPSO at generation 0 are higher than those of other PSO algorithms. This means that the initialized swarm of ISBPSO obtains the best *gbest* in most cases. In a few cases (on Movement and Isolet5), the fitness values at generation 0 of ISBPSO are similar or slightly lower than the benchmark algorithms. However, the fitness values of ISBPSO increase quickly during the following iterations after the initialization. This denotes that ISBPSO obtains a higher quality initial swarm than benchmark PSO algorithms, which results in good convergence performance of ISBPSO.

ISBPSO-IN adopts the random strategy instead of the proposed FWDI method for initialization. We further compare ISBPSO-IN with the benchmark PSO algorithms to test if the proposed ISBPSO algorithm still performs effectively using the same initialization method as other PSO algorithms. So, the performance of the DBM strategy and genetic operations can be evaluated. According to Fig. 9, ISBPSO-IN obtains higher convergence curves than the benchmark PSO algorithms in most cases. Only on Sonar, LSVT, and Madelon, ISBPSO-IN obtains similar convergence curves to the best curve of SBPSO, UBPSO, QBPSO, and CSO(C). These results denote that ISBPSO-IN shows a good convergence performance. Except for ISBPSO-IN, CSO(C) generally obtains higher convergence curves than other PSO algorithms in most cases. This is because that most of the tested datasets are high dimensional data (9 of the 12 datasets have more than 100 features), and the optimization mechanism of CSO(C) is suitable for solving large-scale optimization problems. According to the convergence cures, UBPSO performs worst on all the datasets. This is because that UBPSO evolves slowly in the early phase of the evolutionary process by setting a small initial inertia weight, which has an impact on the final search results. Comparing between ISBPSO-IN and SBPSO, these two methods have similar convergence curves in the early phase of the evolutionary process. However, ISBPSO-IN finds better final solutions than SBPSO. This clearly shows that the DBM strategy and the genetic operations are effective for improving the search performance of ISBPSO. The effect of the DBM strategy is also discovered in the convergence cures obtained by ISBPSO-IN. For example, on InterAd, at the generations of 25, 50, and 75 (which are the generations to update the mask to reduce the search space), an obvious increasing trend of the convergence curve is shown.

### *7.3. Discussion*

Comparisons between ISBPSO and its three variants (ISBPSO-IN, ISBPSO-DBM, and ISBPSO-GO) have clearly shown that the FWDI method, DBM strategy, and genetic operations adopted by ISBPSO are all effective for improving the search performance. Moreover, the convergence curves of the PSO algorithms denote that ISBPSO can quickly converge to a high fitness value and thus have a good convergence property. These results show that ISBPSO has very competitive search performance, which yields good FS results as

shown in Section 6. The following reasons can explain the effectiveness of ISBPSO.

First, the convergence curves in Fig. 9 have shown that the FWDI method can lead to a better starting point for the evolutionary process. This is because that the MI measure used by FWDI can catch the correlation level between each feature and the class label. For a feature, the correlation level reflects its potential predictive power for the class label. The features with higher MI values are given more chances to be selected by particles. Thus, particles with potentially better quality are generated, which improves the quality of the initial swarm and the evolutionary speed of ISBPSO.

Second, comparisons between ISBPSO and ISBPSO-DBM in Section 7.1 have shown that the DBM strategy is effective for improving the search performance of ISBPSO. The DBM strategy gradually masks some bits in the particles to stop them from further updating. Thus, ISBPSO only needs to search in a smaller space, and the required computational resources of ISBPSO are reduced. From another perspective, masking the bits means confirming partial FS results during the evolutionary process rather than confirming the final FS results at the end of the iterations. Once a bit is masked, the optimizer does not pay any more effort to evolve it, which simplifies the FS problem to be solved. Another reason for the effectiveness of DBM is that the mask generating strategy is effective. The mask is generated and updated by extracting information from the particles in the swarm. When updating the mask, features that are not selected by all the particles' *pbest*s are masked. This makes sense and has shown to be effective because it is very possible that a feature is uninformative if it is not selected by most of the *pbest*s in the swarm. By masking some features, the optimizer can concentrate on evolving in the space formed of the remaining features that have not been decided to be selected or not.

Third, as shown in Section 7.1, the genetic operations are effective for improving the search performance. In conventional PSO algorithms, a particle mainly learns from its *pbest* and *gbest* to update its position, which can result in a quick decrease of the swarm diversity. In comparison, for ISBPSO, the crossover operations form an effective way for exchanging the information within the particles, and the mutation operation gives more randomness to generate new solutions. Therefore, the swarm diversity is improved, which leads to better search performance.

Although the experimental results in Section 6 have shown that ISBPSO obtains better FS results in most cases compared to the benchmark methods, ISBPSO shows worse FS results in a few cases. ISBPSO obtains significantly worse accuracies than most of the benchmark PSO algorithms on two relatively lower-dimensional datasets, Sonar and Musk1. In terms of the number of selected features, ISBPSO selects fewer features than these benchmark PSO algorithms. This shows that ISBPSO might remove some informative features, which leads to lower accuracies. Two possible reasons can lead to this result. First, MI used by FWDI of ISBPSO only considers the correlation between a feature and the class label, while it neglects the correlations among multiple features. This may yield inaccurate feature weighting information for initialization. Thus, the initialized swarm by FWDI may guide the ISBPSO to local optima and eliminate

some informative features. Second, ISBPSO adopts the DBM strategy that gradually reduces the search space during the evolutionary process. Even though we adopt a conservative scheme that only masks the bits (features) that are not selected by all the particles' *pbest*s in the swarm. There is a possibility that some key features are masked (eliminated) before being selected. As some key features are eliminated by DBM, the final FS results of ISBPSO are affected.

To sum up, the FWDI method and the DBM strategy are designed to accelerate the evolutionary process of ISBPSO, while these two mechanisms may incorrectly eliminate some informative features during the evolutionary process. For high dimensional data (with hundreds or thousands features) that require enormous computational resources, ISBPSO can generally obtain better FS results than existing PSO algorithms since the evolutionary speed of ISBPSO is much improved using the new mechanisms. However, for low dimensional data (several or tens) that require moderate computational resources, existing PSO algorithms can be a better choice since these algorithms have enough computational resources to find a better solution (feature subset) than ISBPSO, while the aggressive feature reduction strategies (i.e., FWDI and DBM) of ISBPSO may incorrectly eliminate some informative features.

## 8. Conclusions

In this paper, an improved binary PSO algorithm named ISBPSO is proposed for FS in classification. To improve the FS performance, ISBPSO adopts three mechanisms, i.e., the FWDI method, the DBM strategy, and the genetic operations, based on standard SBPSO. FWDI is an initialization method that adopts the feature weighting results of MI for improving the quality of the initial swarm. The DBM strategy dynamically reduces the search space of the optimizer during the evolutionary process. The genetic operations are conducted in ISBPSO as a refinement procedure in addition to the main PSO-based solution updating mechanism to update the *gbest* of particles to improve the global search performance. The proposed ISBPSO algorithm is then used to build a wrapper-based FS method.

The experimental results on 12 UCI datasets show the effectiveness and efficiency of ISBPSO. ISBPSO outperforms the benchmark PSO-based FS methods by obtaining better or similar accuracies with fewer features in most cases. ISBPSO outperforms two conventional FS methods, SFS and SBS, by obtaining higher accuracies in most cases. ISBPSO significantly reduces the computation time compared with benchmark PSO-based FS methods, since it has better feature reducing performance than these PSO algorithms. Moreover, the further analysis shows that all the three mechanisms proposed in ISBPSO are effective for improving the search performance, and ISBPSO shows a good converge property.

Although the proposed ISBPSO algorithm significantly reduces the computation time on big data with hundreds or thousands of features and instances compared with the benchmark PSO algorithms, it still takes much computation time. This is because the wrapper framework adopted by ISBPSO requires a large

amount of time for fitness function evaluations. Therefore, our next work will focus on embedding data scale reduction strategies (e.g., instance selection) in PSO-based FS methods to improve time efficiency.

## Acknowledgments

The authors would like to thank the editor and anonymous referees for the constructive comments and suggestions. This work was supported in part by the Humanities and Social Sciences Youth Fund of Ministry of Education of China under Grants 19YJC630071 and 19YJC630221, the Marsden Fund of New Zealand Government under Contracts VUW1509, VUW1615, VUW1913 and VUW1914, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, the University Research Fund at Victoria University of Wellington grant number 223805/3986, MBIE Data Science SSIF Fund under the contract RTVU1914, and National Natural Science Foundation of China (NSFC) under Grant 61876169 and 71802145.

## References

[1] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (Mar.) (2003) 1157–1182.

[2] P. P. Kundu, S. Mitra, Feature selection through message passing, IEEE Transactions on Cybernetics 47 (12) (2017) 4356–4366.

[3] A.-D. Li, Z. He, Q. Wang, Y. Zhang, Key quality characteristics selection for imbalanced production data using a two-phase bi-objective feature selection method, European Journal of Operational Research 274 (3) (2019) 978 – 989.

[4] B. Xue, M. Zhang, W. N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (4) (2016) 606–626.

[5] L. Yu, H. Liu, Efficient feature selection via analysis of relevance and redundancy, Journal of Machine Learning Research 5 (Oct.) (2004) 1205–1224.

[6] G. Karakaya, S. Galelli, S. D. Ahipasaoglu, R. Taormina, Identifying (quasi) equally informative subsets in feature selection problems for classification: A max-relevance min-redundancy approach, IEEE Transactions on Cybernetics 46 (6) (2016) 1424–1437.

[7] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (8) (2005) 1226–1238.

[8] X. Yan, M. Jia, Intelligent fault diagnosis of rotating machinery using improved multiscale dispersion entropy and mRMR feature selection, Knowledge-Based Systems 163 (2019) 450 – 471.

[9] L. Zhang, X. Huang, W. Zhou, Logistic local hyperplane-relief: A feature weighting method for classification, Knowledge-Based Systems 181 (2019) 104741.

[10] M. Robnik-Šikonja, I. Kononenko, Theoretical and empirical analysis of ReliefF and RReliefF, Machine Learning 53 (1) (2003) 23–69.

[11] A.-D. Li, Z. He, Multiobjective feature selection for key quality characteristic identification in production processes using a nondominated-sorting-based whale optimization algorithm, Computers & Industrial Engineering 149 (2020) 106852.

[12] R. Kohavi, G. H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1) (1997) 273 – 324.

[13] I. Oh, J. Lee, B. R. Moon, Hybrid genetic algorithms for feature selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (11) (2004) 1424–1437.

[14] L. Cervante, B. Xue, M. Zhang, L. Shang, Binary particle swarm optimisation for feature selection: A filter based approach, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2012), 2012, pp. 1–8.

[15] K. Neshatian, M. Zhang, P. Andreae, A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming, IEEE Transanctions on Evolutionary Computation 16 (5) (2012) 645–661.

[16] E. Hancer, B. Xue, M. Zhang, Differential evolution for filter feature selection based on information theory and feature ranking, Knowledge-Based Systems 140 (2018) 103 – 119.

[17] S. Tabakhi, P. Moradi, Relevance–redundancy feature selection based on ant colony optimization, Pattern Recognition 48 (9) (2015) 2798 – 2811.

[18] E. Hancer, B. Xue, D. Karaboga, M. Zhang, A binary abc algorithm based on advanced similarity scheme for feature selection, Applied Soft Computing 36 (2015) 334 – 348.

[19] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948 vol.4.

[20] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Vol. 5, 1997, pp. 4104–4108 vol.5.

[21] B. Xue, M. Zhang, W. N. Browne, Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms, Applied Soft Computing 18 (2014) 261 – 276.

[22] Y. Zhang, S. Wang, P. Phillips, G. Ji, Binary PSO with mutation operator for feature selection using decision tree applied to spam detection, Knowledge-Based Systems 64 (2014) 22–31.

[23] B. Xue, S. Nguyen, M. Zhang, A new binary particle swarm optimisation algorithm for feature selection, in: A. I. Esparcia-Alcázar, A. M. Mora (Eds.), Applications of Evolutionary Computation, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 501–513.

[24] B. H. Nguyen, B. Xue, P. Andreae, A novel binary particle swarm optimization algorithm and its applications on knapsack and feature selection problems, in: Intelligent and Evolutionary Systems, Springer International Publishing, Cham, 2017, pp. 319–332.

[25] H. Banka, S. Dara, A hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high dimensional feature selection, classification and validation, Pattern Recognition Letters 52 (2015) 94 – 100.

[26] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, B. Fielding, A micro-GA embedded PSO feature selection approach to intelligent facial emotion recognition, IEEE Transactions on Cybernetics 47 (6) (2017) 1496–1509.

[27] K. Chen, F.-Y. Zhou, X.-F. Yuan, Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection, Expert Systems with Applications 128 (2019) 140 – 156.

[28] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017).
URL `http://archive.ics.uci.edu/ml`

[29] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (3) (1948) 379–423.

[30] U. M. Fayyad, K. B. Irani, Multi-interval discretization of continuous-valued attributes for classification learning, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993, 1993, pp. 1022–1029.

[31] C. Freeman, D. Kulić, O. Basir, Feature-selected tree-based classification, IEEE Transactions on Cybernetics 43 (6) (2013) 1990–2004.

[32] Z. Tao, L. Huiling, W. Wenwen, Y. Xia, GA-SVM based feature selection and parameter optimization in hospitalization expense modeling, Applied Soft Computing 75 (2019) 323 – 332.

[33] Y. Zhu, J. Liang, J. Chen, Z. Ming, An improved NSGA-III algorithm for feature selection used in intrusion detection, Knowledge-Based Systems 116 (2017) 74 – 85.

[34] A.-D. Li, B. Xue, M. Zhang, Multi-objective feature selection using hybridization of a genetic algorithm and direct multisearch for key quality characteristic selection, Information Sciences 523 (2020) 245 – 265.

[35] K. Nag, N. R. Pal, A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification, IEEE Transactions on Cybernetics 46 (2) (2016) 499–510.

[36] Y. Zhang, D. wei Gong, X. zhi Gao, T. Tian, X. yan Sun, Binary differential evolution with self-learning for multi-objective feature selection, Information Sciences 507 (2020) 67 – 85.

[37] P. Shunmugapriya, S. Kanmani, A hybrid algorithm using ant and bee colony optimization for feature selection and classification (ac-abc hybrid), Swarm and Evolutionary Computation 36 (2017) 27 – 36.

[38] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Transactions on Cybernetics 45 (2) (2015) 191–204.

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

[40] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.

[41] B. Tran, B. Xue, M. Zhang, Variable-length particle swarm optimisation for feature selection on high-dimensional classification, IEEE Transactions on Evolutionary Computation 23 (3) (2018) 473–487.

[42] B. Tran, B. Xue, M. Zhang, A new representation in pso for discretization-based feature selection, IEEE Transactions on Cybernetics 48 (6) (2018) 1733–1746.

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

[44] J. Kennedy, Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No.03EX706), 2003, pp. 80–87.

[45] B. Xue, M. Zhang, W. N. Browne, Particle swarm optimization for feature selection in classification: A multi-objective approach, IEEE Transactions on Cybernetics 43 (6) (2013) 1656–1671.

[46] H. B. Nguyen, B. Xue, I. Liu, P. Andreae, M. Zhang, New mechanism for archive maintenance in pso-based multi-objective feature selection, Soft Computing 20 (10) (2016) 3927–3946.

[47] M. Amoozegar, B. Minaei-Bidgoli, Optimizing multi-objective PSO based feature selection method using a feature elitism mechanism, Expert Systems with Applications 113 (2018) 499 – 514.

[48] C.-L. Huang, J.-F. Dun, A distributed PSO–SVM hybrid system with feature selection and parameter optimization, Applied Soft Computing 8 (4) (2008) 1381 – 1391.

[49] M. M. Mafarja, R. Jarrar, S. Ahmad, A. A. Abusnaina, Feature selection using binary particle swarm optimization with time varying inertia weight strategies, in: A. Abuarqoub, B. Adebisi, M. Hammoudeh, S. Murad, M. Arioua (Eds.), Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, Amman, Jordan, June 26-27, 2018, ACM, 2018, pp. 1–9.

[50] P. Moradi, M. Gholampour, A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy, Applied Soft Computing 43 (2016) 117 – 130.

[51] I. Jain, V. K. Jain, R. Jain, Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification, Applied Soft Computing 62 (2018) 203 – 215.

[52] M. A. Khanesar, M. Teshnehlab, M. A. Shoorehdeli, A novel binary particle swarm optimization, in: 2007 Mediterranean Conference on Control Automation, 2007, pp. 1–6.

[53] B. Xue, M. Zhang, W. N. Browne, Multi-objective particle swarm optimisation (PSO) for feature selection, in: Proceedings

of the 14th Annual Conference on Genetic and Evolutionary Computation, GECCO '12, ACM, New York, NY, USA, 2012, pp. 81–88.

[54] S. Mirjalili, A. Lewis, S-shaped versus v-shaped transfer functions for binary particle swarm optimization, Swarm and Evolutionary Computation 9 (2013) 1 – 14.

[55] Y. Ong, M. H. Lim, X. Chen, Memetic computation—past, present future [research frontier], IEEE Computational Intelligence Magazine 5 (2) (2010) 24–31.

[56] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[57] J. K. Martin, D. S. Hirschberg, The time complexity of decision tree induction, Tech. rep., University of California at Irvine (1995).

[58] J. Liu, Y. Mei, X. Li, An analysis of the inertia weight parameter for binary particle swarm optimization, IEEE Transactions on Evolutionary Computation 20 (5) (2016) 666–681.

[59] Y. Jeong, J. Park, S. Jang, K. Y. Lee, A new quantum-inspired binary PSO: Application to unit commitment problems for power systems, IEEE Transactions on Power Systems 25 (3) (2010) 1486–1495.

[60] D. W. Aha, D. Kibler, M. K. Albert, Instance-based learning algorithms, Machine Learning 6 (1) (1991) 37–66.

[61] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The WEKA data mining software: an update, ACM SIGKDD explorations newsletter 11 (1) (2009) 10–18.

[62] J. D. Gibbons, S. Chakraborti, Nonparametric Statistical Inference, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.