

A New Representation in PSO for Discretization-Based Feature Selection

Binh Tran, *Student Member, IEEE*, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Senior Member, IEEE*

Abstract—In machine learning, discretization and feature selection (FS) are important techniques for preprocessing data to improve the performance of an algorithm on high-dimensional data. Since many FS methods require discrete data, a common practice is to apply discretization before FS. In addition, for the sake of efficiency, features are usually discretized individually (or univariate). This scheme works based on the assumption that each feature independently influences the task, which may not hold in cases where feature interactions exist. Therefore, univariate discretization may degrade the performance of the FS stage since information showing feature interactions may be lost during the discretization process. Initial results of our previous proposed method [evolve particle swarm optimization (EPSO)] showed that combining discretization and FS in a single stage using bare-bones particle swarm optimization (BBPSO) can lead to a better performance than applying them in two separate stages. In this paper, we propose a new method called potential particle swarm optimization (PPSO) which employs a new representation that can reduce the search space of the problem and a new fitness function to better evaluate candidate solutions to guide the search. The results on ten high-dimensional datasets show that PPSO select less than 5% of the number of features for all datasets. Compared with the two-stage approach which uses BBPSO for FS on the discretized data, PPSO achieves significantly higher accuracy on seven datasets. In addition, PPSO obtains better (or similar) classification performance than EPSO on eight datasets with a smaller number of selected features on six datasets. Furthermore, PPSO also outperforms the three compared (traditional) methods and performs similar to one method on most datasets in terms of both generalization ability and learning capacity.

Index Terms—Classification, discretization, feature selection (FS), high-dimensional data, particle swarm optimization (PSO).

I. INTRODUCTION

MORE and more high-dimensional datasets with thousands to tens of thousands of features become common in many machine learning applications in different fields such

as bioinformatics, genomics, image processing, and text classification [1]. These datasets usually have a significant number of redundant features and irrelevant features that can be considered as noise, which negatively affects the performance of the learning algorithm. Therefore, feature selection (FS) is usually a critical preprocessing step to select only relevant features for many machine learning tasks such as clustering and classification [1]. In this paper, we focus on FS for classification. Although many studies have shown the effectiveness of applying FS on high-dimensional data [1], [2], it is still challenging due to the large search space and the existence of feature interactions.

In addition to FS, discretization is also crucial in preprocessing high-dimensional data. First of all, many learning algorithms are applicable to or efficient on discrete data only. Therefore, discretization is used to transform features with continuous values into discrete or nominal values. Furthermore, via discretization, minor fluctuations or possible noise in the data can be ignored. In this way, discretization helps learning algorithms improve their effectiveness and efficiency [4]. Last but not least, since discrete data is more compact than continuous data, it requires less memory, and thus improving efficiency of learning algorithms. Therefore, FS and discretization are commonly used in order to improve not only classification performance, but also computation time and storage requirement [5], [6].

Although many discretization methods have been proposed, the most commonly used discretization methods are univariate. By discretizing one feature at a time, these methods are quite efficient when feature interactions do not exist [7]. When this assumption does not hold, it is necessary to discretize multiple features simultaneously. However, the time complexity of multivariate discretization will be much higher. Therefore, a more powerful search technique is required to enable multivariate discretization especially on high-dimensional data.

Furthermore, in practice, univariate discretization is usually applied before FS as a requirement of many FS methods. However, univariate discretization can destroy information about feature interactions which may be important for FS stage. Therefore, better representation may be obtained by merging the two processes in a single stage.

Particle swarm optimization (PSO) is a meta-heuristic algorithm proposed by Eberhart and Kennedy [8]. PSO imitates social behaviors found in birds flocking. Many FS methods have been proposed using different types of PSO algorithms such as continuous PSO [9], [10] and binary PSO [11]–[13]. Results of these methods have shown high potential of

Manuscript received September 10, 2016; revised January 22, 2017 and March 22, 2017; accepted May 28, 2017. Date of publication June 23, 2017; date of current version May 15, 2018. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1209, Contract VUW1509, and Contract 16-VUW-111, in part by the Huawei Industry Fund under Grant E2880/3663, and in part by the University Research Fund at the Victoria University of Wellington under Grant 209861/3580, Grant 209862/3580, and Grant 213150/3662. This paper was recommended by Associate Editor A. M. Alimi. (Corresponding author: Binh Tran.)

The authors are with the Evolutionary Computation Research Group, Victoria University of Wellington, Wellington, New Zealand (e-mail: binh.tran@ecs.vuw.ac.nz).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCYB.2017.2714145

2168-2267 © 2017 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

PSO in this field. However, PSO has not been applied to discretization.

In a recent method [evolve particle swarm optimization (EPSO)] [14], we proposed to use a derived type of PSO called “bare-bones” PSO (BBPSO) [15] to simultaneously discretize and select features because of the following reasons. In PSO for FS, the PSO representation is usually a N -dimension vector corresponding to N features. Each value is of the range $[0, 1]$. If it is greater than a predefined threshold, the corresponding feature is selected and vice versa, regardless of how much smaller or greater it is when compared to the threshold. Therefore, two different evolved vectors may result in the same feature subset. On the other hand, in discretization, a slightly different evolved cut-point may lead to a different discrete feature. As a result, finding a good cut-point requires a fine-tuned search mechanism which can be found in BBPSO. In this derived version of PSO, new positions are sampled using a Gaussian random generator with the center being the mean position of the individual best position (pbest) and its neighbors’ best position (gbest) and standard deviation being the distance between them.

EPSO used BBPSO in a straightforward way to achieve discretization and FS. One cut-point is evolved for each feature. Since a cut-point can be any value within the feature’s range, the number of possible solutions in discretization is much larger than FS. Therefore, entropy-based cut-points inspired from [16] are used as initial or potential cut-points to narrow the search space. The proposed method has shown promising results. However, with the proposed representation, the search space is still too big for BBPSO to obtain better performance. To narrow the search space, in this paper, we propose a new method called potential particle swarm optimization (PPSO) which uses BBPSO to choose an appropriate cut-point for each feature among the potentially good ones. A new fitness function and a scaling mechanism are also proposed to improve the performance of the method.

A. Goals

This paper aims to develop a new method combining discretization and FS in a single stage using BBPSO for high-dimensional continuous data. A new representation for particles in BBPSO is proposed to achieve this goal. While EPSO [14] used BBPSO to directly evolve a cut-point for each feature, the new approach called PPSO allows BBPSO to automatically choose *potentially* good cut-points for discretization and FS. Performance of PPSO is compared with using full feature set, the two-stage approach of discretization and FS, and EPSO on high-dimensional data. Our specific research objectives include the following.

- 1) How to perform multivariate discretization and selection in a single stage to improve the feature set’s discriminating power.
- 2) Whether the features discretized and selected by PPSO can obtain better classification accuracy than using full feature set.
- 3) Whether PPSO performs better than the corresponding two-stage approach in terms of accuracy, feature subset size and computation time.

- 4) Whether PPSO is more effective and efficient than EPSO.
- 5) Whether PPSO achieves better classification performance than traditional methods in both cases with and without *FS bias*.¹
- 6) Whether PPSO results are generalized well to the learning algorithms other than the wrapped method.

Although both new method (PPSO) proposed in this paper and the existing method (EPSO) in [14] aim to tackle FS via discretization, their major components, which are the representation and the evaluation method, are very different. In evolutionary computation (EC) techniques, representation plays an important role in the effectiveness and efficiency of the method. A good representation can reduce the size of the search space, which is usually very large in EC approaches. The search space of FS problem alone is already very large, so combining it with discretization results in an even larger space. Therefore, in this paper, we propose a new representation so that the search process can be more effective. Another contribution of this paper is a new fitness function that combines wrapper and filter measures in order to synthesize the strength of both approaches. The proposed hybrid approach is designed to better evaluate the candidate solution, however, without paying more computational cost by using wrapper and filter methods that are based on the same distance measure.

II. BACKGROUND AND RELATED WORK

This section provides background on FS and discretization, introduces the entropy-based cut-points and the minimum description length principle (MDLP) which is employed in this paper. It then presents methods that are closely related to our proposed method, FS via discretization.

A. Feature Selection

FS is essentially a combinatorial optimization problem since there are 2^N possible different feature subsets to choose from a dataset with N features. An FS method usually has two main components which are a search technique and a feature evaluation method.

In terms of feature evaluation, FS methods can be generally categorized into filter and wrapper approaches [18]. *Filter approaches* evaluate features based on their intrinsic characteristics. Examples of filter measures are distance [19], information gain [20], consistency [21], and correlation [22]. On the other hand, *wrapper approaches* use a learning algorithm to measure the classification performance of the selected features. Different learning algorithms can be used in this process, such as K -nearest neighbor (KNN) [23], decision tree [24], and support vector machines [25]. In general, filter methods are faster than wrapper methods; however, they usually obtain lower classification accuracy than wrappers [26].

Search techniques in FS methods can be divided into exhaustive search, heuristic search, random or stochastic search [27]. While exhaustive search guarantees to find the

¹*FS bias* is an important issue happened in FS when the whole set of data is used during the FS process; therefore, no unseen data is used to test the performance of the FS method [17].

best solutions, it is infeasible for most real-world applications due to its high computation time. Sequential search such as linear forward selection (LFS) [28] and greedy stepwise backward selection (GSBS) [29] are typical methods of heuristic search. These two methods are derived versions of sequential forward selection (SFS) [30] and sequential backward selection [31], respectively. LFS [28] improved SFS effectiveness and efficiency by limiting the number of features considered in each step. Although backward selection can consider feature interactions better than forward selection [32], it is impractical to apply to datasets with a large number of features. GSBS cannot finish within a week when running on datasets with hundreds of features [33]. In additions, both forward and backward strategies usually face the problem of local optima [18].

Random search may generate subsets in a perfectly random fashion using Las Vegas algorithm such as LVW [34] which is too slow to converge in a large search space. Instead of randomly generation, EC is a stochastic approach that applies evolutionary principle or swarm intelligence to generate better subsets from the current ones. PSO is a swarm intelligence technique that has been applied and shown their effectiveness in FS [20], [35]. Readers are referred to more comprehensive survey on different strategies for FS using EC techniques [36], [37]. Although PSO has been successfully applied to different optimization problems including FS, it has not been applied to discretization.

B. Feature Discretization

Feature discretization is a topic with a long research history. Many discretization methods with different strategies have been proposed in this area. However, all of them share the same purpose which is to determine cut-points to partition features values into discrete values. Cut-points or split-points are real values within the range of the feature's values that are used to partition that range into several intervals. Existing discretization methods can be categorized using different criteria [38]–[40]. In *direct* methods, intervals are generated based on a predefined parameter. On the other hand, *incremental* methods recursively split (or merge) intervals until some criteria are met creating *splitting* (or *merging*) approach. They are also known as *top-down* or *bottom-up* approaches, respectively. A discretization method is *supervised* or *unsupervised* depending on whether class labels are used in the discretization process or not. It is said to be *global* if the entire instance space is used in each discretization step, or *local* if each discretization step just uses a subset of instances. A method is also belong to *univariate* or *multivariate* depending on whether features are discretize individually or multiple features are discretize at the same time taking into account interactions between features [38], [40].

Equal-width and equal-frequency binning are two simple unsupervised methods. They discretize features into a predefined number of m intervals with the same width or the same number of values, respectively. These simple methods are easy to implement but sensitive to value of m which is

usually difficult to determine especially when features are not uniformly distributed or contain outliers [41].

Using the class labels as a guide in searching cut-points, supervised discretization usually performs better than the unsupervised counterpart. 1R [42] defines cut-points are feature values lying at the boundary of different classes. Each bin needs to have at least six instances except for the last bin. In addition to different searching techniques, different evaluation measures are proposed such as classification error rates [46], [47], information gain [16], [43], and statistical measures [44], [45]. More comprehensive reviews can be found in [38]–[40] and [48].

Among supervised methods, minimum description length (MDL) proposed by Fayyad and Irani [16] is one of the most popular used methods. It is an entropy-based incremental splitting discretization method. Information gain is used to evaluate candidate cut-points. MDL recursively chooses the best cut-point to split one interval into two until the MDLP is met. Inspired by this strategy, we propose to use the entropy-based cut-points that are accepted by MDLP as initial or candidate cut-points for BBPSO to evolve from.

C. Entropy-Based Cut Points

Entropy-based discretization aims to find the best splits so that the discretized features are as pure as possible in terms of class labels. This means that the majority of the values in one interval are preferred to have the same class label. If entropy $E(S)$ is used to measure the purity level of a set S , then according to this criterion, the cut-point with the highest information gain is the best one. Equation (1) is used to calculate information gain of a cut-point T for feature A given S as the set of feature A values. S_1 and S_2 are subsets of S after partitions

$$\text{Gain}(T, A; S) = E(S) - \frac{|S_1|}{|S|}E(S_1) - \frac{|S_2|}{|S|}E(S_2). \quad (1)$$

Furthermore, not all cut-points are useful especially when the features are noisy or irrelevant to the class label. Therefore, Fayyad and Irani [16] proposed to use the MDLP as a criterion to accept a cut-point. According to MDLP, as shown in (2), a cut-point T is only accepted if its information gain is greater than the cost of encoding the cut-point T and the classes of the instances in the intervals induced by T . If the cut-point T induces impure intervals, the sum of these two costs may exceed its information gain. As shown in (3), the latter cost $\delta(T, A; S)$ will become larger if the entropy values $E(S_1)$ and $E(S_2)$ are larger. In noisy or irrelevant features, the cut-points usually cannot induce intervals that are pure enough to satisfy the condition in (2). Therefore, using MDLP can deal with noise or irrelevant data

$$\text{Gain}(T, A; S) > \frac{\log_2(|S| - 1)}{|S|} + \frac{\delta(T, A; S)}{|S|} \quad (2)$$

where

$$\delta(T, A; S) = \log_2(3^{k_s} - 2) - [k_s E(S) - k_{S_1} E(S_1) - k_{S_2} E(S_2)] \quad (3)$$

given $|S|$ as size of set S , $E(S)$ as the entropy of S , and k_S as the number of classes appeared in S .

D. Feature Selection via Discretization

Although FS and discretization are emerging fields in recent decades, approaches that combine these tasks have not gained much attention. Chi2 [49] is one of the first methods proposing FS via discretization. It is a bottom up method starting from intervals with only one feature value. Then adjacent intervals with the lowest χ^2 test result will be merged recursively until χ^2 values of all pairs exceeds a threshold. This threshold is determined by attempting to maintain a predefined consistency level of the data. By loosing this consistency level, Chi2 can come up with features that have only one interval, which can be removed for FS. Results on two synthetic datasets showed that Chi2 effectively discretized relevant features and removed all noise features. However, the user-defined inconsistency rate is hard to predefined and may also cause inaccuracy to the discretization process [50]. Modified Chi2 (MChi2) [50] is a completely automatic discretization method that address the drawbacks of Chi2.

Another approach of FS via discretization is first ranking features based on some measures calculated during the discretization process. Then, a number of top ranked features will be selected. An example of this approach is PEAR [51] in which features are ranked from those with the smallest number of cut-points to the largest. The top ranked features are considered as relevant and selected to form the final subset. Results showed that it has similar performance as the original feature set and better than Relief. However, it is difficult to choose appropriate parameters for PEAR and the number of features that should be selected to form the final subset. Similarly, in [52], features are ranked based on the ratio of the variance of the original continuous values and the number of bits used to encode the discrete feature.

In summary, FS via discretization has been proposed in two separate stages. However, a single stage integrating them together has not been investigated yet.

E. Particle Swarm Optimization

1) *Continuous PSO*: PSO proposed by Eberhart and Kennedy [8] is a population-based algorithm which maintains a swarm of particles. Each particle's position represents a complete candidate solution of the problem. It is actually a vector of N real numbers, where N is the dimensionality of the problem. In addition to position, each particle also has a velocity vector indicating its speed and direction that it should move in each dimension. These two elements are updated using (4) and (5). As can be seen from these equations that particles are moving toward pbest and gbest

$$v_{id}^{t+1} = w * v_{id}^t + c_1 * r_{1i}^t * (p_{id}^t - x_{id}^t) + c_2 * r_{2i}^t * (p_{gd}^t - x_{id}^t) \quad (4)$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1} \quad (5)$$

where v_{id}^t and x_{id}^t are velocity and position of particle i in dimension d at time t , respectively. p_{id}^t and p_{gd}^t are pbest and

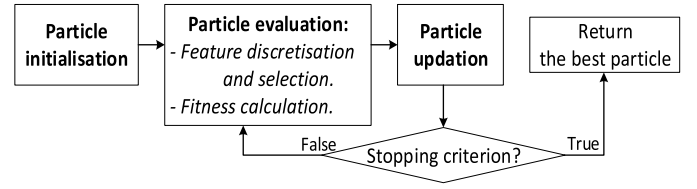


Fig. 1. EPSO and PPSO basic steps.

gbest positions in dimension d at time t . r_{1i}^t and r_{2i}^t are uniformly distributed random values generated at time t . c_1 and c_2 are acceleration constants which determines the type of trajectory the particle travels, thus they are important to control the searching behavior of the particle [53]. The inertia weight w is used to control the impact of the last velocity to the current velocity.

2) *Bare-Bone PSO*: As can be seen in (4) and (5) that PSO operates by sampling points in the search space. It uses pbest and gbest which are the discovered knowledge to guide the sampling. Kennedy [15] investigated the trajectory of a single particle in a standard PSO where pbest and gbest were set as constants. All of its visited positions after a million iterations were plotted. The obtained histogram is a tidy bell curve centered midway between these constant pbest and gbest positions. The result suggests that trajectory of a particle is determined by the difference between its pbest and gbest. Therefore, the step size of a particle's movement should be based on these best positions. This finding resulted in a new PSO method called bare-bones PSO (BBPSO). In this method, particle's position is sampled using a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with the mean μ and standard deviation σ as

$$x_{id}^{t+1} = \begin{cases} \mathcal{N}(\mu, \sigma), & \text{rand}() < 0.5 \\ p_{id}^t, & \text{otherwise} \end{cases} \quad (6)$$

where μ is the center of pbest and gbest and σ is the absolute difference between these best positions. The rand() function is used to speed up convergence by retaining the previous best position pbest.

By using the Gaussian number generator, BBPSO avoids the need to optimize the velocity vector and the delay of position adaptation. Although PSO and BBPSO have been proposed for FS [37], [54], [55], both have not been proposed for discretization or discretization and FS simultaneously.

III. PROPOSED APPROACH

This section describes EPSO and PPSO. Both methods follow the basic steps shown in Fig. 1. After initialization, particles are iteratively evaluated and updated until the stopping criteria are met. In order to evaluate a particle, the training data is first discretized and features are selected based on the evolved cut-points. The transformed data is then put into a learning algorithm to calculate the fitness. Based on this fitness, pbest and gbest are updated and used to update a particle's position as described in (6).

Discretization and FS steps in both methods work in the same principle. To achieve *discretization*, a feature value is converted/discretized into 0 if it is smaller than the evolved cut-point; otherwise, it will be 1. If all values of a feature

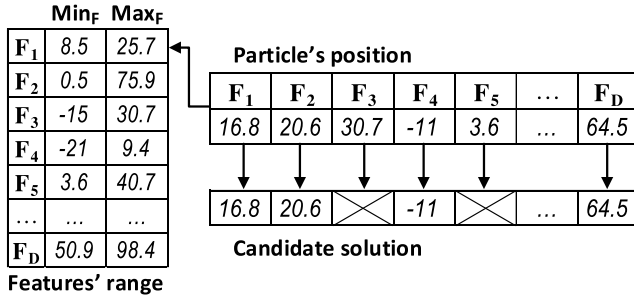


Fig. 2. Particle representation of EPSO.

are converted into the same discrete value, it is considered as an irrelevant feature because it cannot distinguish instances of different classes. FS is done by eliminating these useless features. The remaining discrete features are evaluated based on the improvement in classification performance of the whole discretized data.

A. EPSO Method

The main idea of EPSO is using BBPSO to directly *evolve* a cut-point which can be any real value falling in the range of the corresponding feature values $[\text{Min}_F \dots \text{Max}_F]$. Position of each particle represents a candidate solution which is a real vector with N -dimension corresponding to the dimensionality of the problem. Fig. 2 shows an example of a particle's position and its corresponding candidate solution. In this example, the first dimension of the particle, which represents the cut-point for the first feature (F_1), needs to have a value within the range $[8.5, 25.7]$. If an updated cut-point of a feature F is out of this range, it will be set to the nearest boundary.

1) *Particle Initialization*: Because the search space of multivariate discretization on high-dimensional data is huge, each initial particle is restricted to a random feature subset of size 50 for binary-class problems and size 150 for multiclass problems as suggested in [56]. This means that for those features that are not selected in the initial candidate solutions, their cut-points will be set to the maximum values of the corresponding features. For the other selected features, their cut-points are initialized using the best entropy-based cut-point that satisfies MDLP [16] (see Section II-C). In principle, they can be initialized based on any value within the range of the corresponding feature. However, completely random initial cut-points may lead to slow convergence. Furthermore, information gain of the best cut-point of a feature is an indicator of its relevancy. Therefore, features with larger information gain will have higher probability to be selected in the initialization procedure.

2) *Particle Evaluation*: Based on the cut-points evolved by the particle, the training data is transformed into a new training set with discrete values and a smaller number of features thanks to eliminating features whose cut-points are equal to the minimum or maximum values. For example, in Fig. 2, F_3 cut-point is equal to its maximum value and F_5 cut-point is equal to its minimum value, both features will be discarded.

The discretization and FS solution in each particle is then evaluated based on the classification accuracy of the transformed training set. By evaluating the whole discrete data,

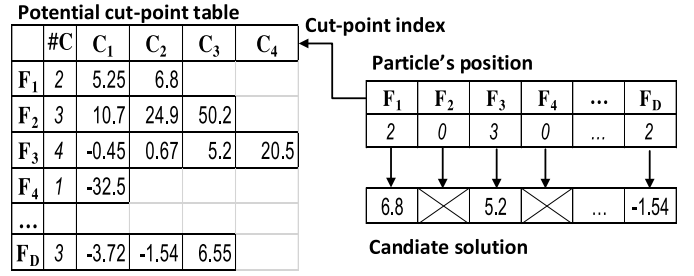


Fig. 3. Particle representation of PPSO.

the proposed method is able to evaluate cut-points of all the selected features as well as consider feature interactions at the same time. Fitness function uses the balanced classification accuracy [57] as follows:

$$\text{balanced_accuracy} = \frac{1}{c} \sum_{i=1}^c \frac{\text{TP}_i}{|S_i|} \quad (7)$$

where c is the number of classes of the problem, TP_i is the number of correctly identified instances in class i , and $|S_i|$ is the sample size of class i . All classes are treated equally with the weight of $1/c$.

B. Proposed Method: PPSO

In EPSO, BBPSO is free to evolve a cut-point within the range of the corresponding feature. This may result in a huge search space especially in a multivariate approach on high-dimensional data. Therefore, to narrow the search space into highly potential areas, in PPSO, we use BBPSO to choose a cut-point from *potential* cut-points of each feature. Potential cut-points are entropy-based cut-points that have their information gain satisfying the MDLP criterion shown in (2). Each feature may have a different number of potential cut-points which are calculated and stored in a potential cut-point table. Fig. 3 shows an example of this table and a particle's position as well as the corresponding candidate solution. Each particle position is an integer vector denoting the chosen cut-point indexes. The size of vector, therefore, is equal to the number of the original features and the evolved value needs to be between 1 and the number of potential cut-points of the corresponding feature. For example, in Fig. 3, the first feature (F_1) has two potential cut-points with indexes 1 and 2. Therefore, the first dimension of the particle needs to fall in the range $[1, 2]$. If it is 2 as in the example, then the cut-point 6.8 is chosen to discretize F_1 .

During the updating process, if the updated value of a dimension is outside the cut-point index range, then it is set to 0 which indicates that the corresponding feature does not have a good cut-point and therefore should be ignored.

1) *Particle Initialization*: Each particle position is initialized as a random feature subset with a restricted number of selected features which have their cut-point indexes different from 0. The selected features will have their cut-point index set to the index of the best MDLP cut-point (see Section II-C) [16]. Similar to EPSO, features with higher information gain will have higher chance to be selected.

In order to make PPSSO general to all problems, PPSSO uses one restricted size for all datasets. Then during the evolutionary process, when BBPSO seems to get stuck in local optima, BBPSO will be reset with a larger size if the current gbest fitness is better than the last gbest fitness at least 10%. The aim of this scaling mechanism is to start searching from small feature subsets while opening chances for larger and better feature subsets.

2) *Particle Evaluation*: Based on the chosen cut-point index for each feature, the cut-point value is retrieved from the potential cut-point table. It is then used to discretize the corresponding feature. However, if the evolved cut-point index of a feature is 0, that feature is considered as not selected. Therefore, in Fig. 3, features F_2 and F_4 are not selected in this example.

In EPSO, classification accuracy is used as the fitness measure to evaluate each particle. This may be difficult to distinguish cases where the boundary between classes is quite large, enabling many different models to obtain the same accuracy. Furthermore, while the wrapper methods can produce solutions with high accuracy, filter methods are usually faster and general. Combining the strength of these two approaches in the evaluation function is expected to produce better solutions. In addition, combining the two measures can also better distinguish the subtle difference between feature subsets, providing a smoother fitness landscape to facilitate the search process. However, simply combining these measures may be impractical in terms of computation. Therefore, we need to find a smart way to combine them without requiring more running time. Among the commonly used filter measures, distance is a multivariate measure that can evaluate the discriminating ability of a feature set and it is used as the base measure of KNN. Therefore, incorporating this measure with the KNN wrapper method will not increase the computation time because the distance measure is calculated only once but used twice.

In this paper, the proposed fitness function uses both balanced classification accuracy (7) and a distance measure with a weighting coefficient (μ) as shown in (8). The distance measure [58], as shown in (9), is used to maximize the distance between instances of different classes (D_B) and minimize the distance between instances of the same class (D_W). D_B and D_W are calculated by using the following equations:

$$\text{fitness} = (\mu * \text{balanced_accuracy} + (1 - \mu) * \text{distance}) \quad (8)$$

$$\text{distance} = \frac{1}{1 + \exp^{-5(D_B - D_W)}} \quad (9)$$

$$D_B = \frac{1}{|S|} \sum_{i=1}^{|S|} \min_{\{j|j \neq i, \text{class}(V_i) \neq \text{class}(V_j)\}} \text{Dis}(V_i, V_j) \quad (10)$$

$$D_W = \frac{1}{|S|} \sum_{i=1}^{|S|} \max_{\{j|j \neq i, \text{class}(V_i) = \text{class}(V_j)\}} \text{Dis}(V_i, V_j) \quad (11)$$

where $\text{Dis}(V_i, V_j)$ is the distance between two vectors V_i and V_j . In this paper, we use the proportion of matches or overlaps between two binary vectors as the distance between them.

As can be seen in (10) and (11), D_B and D_W are the average distance between each instance to its nearest miss and to its

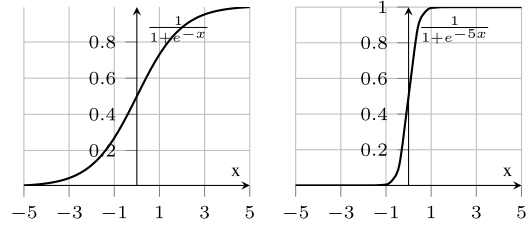


Fig. 4. Logistic function $f(x) = 1/(1 + e^{-x})$, $f(x) = 1/(1 + e^{-5x})$.

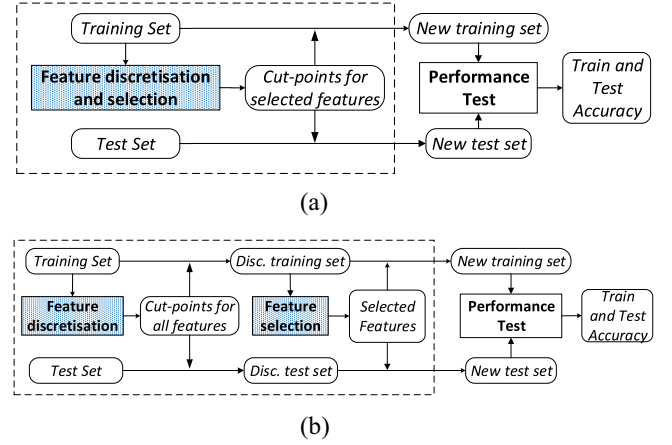


Fig. 5. System overview. (a) EPSO and PPSSO (discretization and FS in a single stage). (b) PSO-FS (discretization and FS in two stages).

farthest hit, respectively. While both of them are in the range $[0, 1]$, 1 is the best case of D_B and the worst case for D_W . Therefore, (9) maximizes D_B and minimizes D_W to find feature subsets showing a close distance between instances of the same class and a far distance between instances of different classes.

The logistic function with coefficient -5 as shown in Fig. 4(right) is used to transform the difference between D_B and D_W in the range $[-1, 1]$ into a value in the full range $[0, 1]$ in which 0 is the worst *distance* and 1 is the best distance. Note that the logistic function with coefficient -1 shown in Fig. 4(left) does not return a value in the full range $[0, 1]$ for an input between -1 and 1 .

C. Overall Approach

An overview of the proposed approach to discretization and FS in one stage is shown in Fig. 5. The two-stage approach (named PSO-FS) is shown in Fig. 5(b) in which features are first discretized and then selected. Except for the details inside the dotted box, both systems have the same structure. First, the dataset is split into a training and a test set. Based on the training set, the evolved cut-points for the selected features are returned and used to transform the training and the test sets. These transformed datasets are input to the classification algorithms for performance evaluation of both methods. Algorithms 1 and 2 present the pseudo-code of EPSO and PPSSO, respectively.

Both EPSO and PPSSO follow the overall structure described in Fig. 5, but are very different in the key aspects. The

Algorithm 1: EPSO Pseudo Code

Input : Original training set T
Output: Selected features and their cut-points

```

begin
   $Best\_cut\_points \leftarrow$  the best entropy cut-points that satisfy Eq. (2) of each feature;
  Initialise particles using  $Best\_cut\_points$ ;
  while Stopping criteria are not met do
    for each particle  $i$  do
       $T'_i \leftarrow$  Transform training set  $T$  based on position of particle  $i$ ;
       $\phi_i \leftarrow$  Evaluate the accuracy of  $T'_i$  using Eq. (7);
       $\xi_i \leftarrow pbest$ 's fitness;
      if  $\phi_i$  is better than  $\xi_i$  then
        Update  $pbest$ ;
      end
    end
    Update  $gbest$ ;
    for each particle  $i$  do
      for each dimension  $j$  do
        Update position of particle  $i$  at dimension  $j$  using Eq. (6);
      end
    end
  end
  Return the selected features and their cut-points from  $gbest$ 's position;
end

```

Algorithm 2: PPSO Pseudo Code

Input : Original training set T
Output: Selected features and their cut-points

```

begin
   $Potential\_cut\_points \leftarrow$  All the entropy cut-points that satisfy Eq. (2) of each feature;
  repeat
    Initialise particles using the index of the best cut-point in  $Potential\_cut\_points$ ;
    while Stopping criteria are not met do
      for each particle  $i$  do
         $T'_i \leftarrow$  Transform training set  $T$  based on position of particle  $i$ ;
         $\phi_i \leftarrow$  Calculate fitness of particle  $i$  based on  $T'_i$  using Eq. (8);
         $\xi_i \leftarrow pbest$ 's fitness;
        if  $\phi_i$  is better than  $\xi_i$  then
          Update  $pbest$ ;
        end
      end
      Update  $gbest$ ;
      if The scale criterion is met then
        Increase the initial size;
        break;
      end
      for each particle  $i$  do
        for each dimension  $j$  do
          Update position of particle  $i$  at dimension  $j$  using Eq. (6);
        end
      end
    end
  until The scale criterion is not met;
  Return the selected features and their cut-points from  $gbest$ 's position;
end

```

major differences are as follows. First of all, the representation scheme of PPSO significantly reduces the search space of the problem when considering only the potentially good cut-points. It improves not only the effectiveness but also the efficiency. Second, the fitness function of PPSO is enforced with a distance measure to provide better evaluation. Third, the initialization procedure in PPSO is more general than EPSO when using one restricted size for all problems to initialize particles. Finally, PPSO uses scaling mechanism to move the search focus to larger feature subsets when needed. These new ideas in PPSO are expected to improve its performance in

TABLE I
DATASETS

Dataset	#Features	#Instances	#Classes	%Smallest class	%Largest class
SRBCT	2,308	83	4	13	35
DLBCL	5,469	77	2	25	75
9Tumor	5,726	60	9	3	15
Leukemia 1	5,327	72	3	13	53
Brain Tumor 1	5,920	90	5	4	67
Leukemia 2	11,225	72	3	28	39
Brain Tumor 2	10,367	50	4	14	30
Prostate	10,509	102	2	49	51
Lung Cancer	12,600	203	5	3	68
11Tumor	12,533	174	11	4	16

terms of the classification accuracy, the number of selected features and the computation cost.

IV. EXPERIMENTAL DESIGN

This section describes the details related to the experiments including the datasets, the baseline methods used to compare with our methods, the parameter settings, the termination criteria as well as the configuration of the experiments.

1) *Datasets*: To test PPSO performance on high-dimensional data, we used ten gene expression datasets which are available on <http://www.gems-system.org>. Table I describes details about these datasets.

2) *Baseline Methods*: To test the effectiveness of PPSO in discretization and FS, we compared the classification performance of KNN on the transformed dataset by PPSO, the original dataset, and the transformed dataset by EPSO. We also compared PPSO with the two-stage method (PSO-FS) to see whether the single stage approach has better performance than the two-stage one. In PSO-FS, MDL [16] is used to discretize data before applying PSO for feature subset selection. PPSO is also compared with several traditional two-stage methods which combines MDL [16] for discretization with LFS [28], with consistency method [21] and correlation-based FS (CFS) method [22] for FS. We also compared PPSO with the MChi2 [50] which is a representative method for FS via discretization. We used the Weka package [29] to run LFS, CFS and consistency, and the KEEL package [59] to run MChi2.

3) *Parameter Settings and Termination Criteria*: Parameter settings for the three compared methods, PSO-FS, EPSO, and PPSO, are described in Table II. Since the size of the search space is proportional to the dimensionality of the problem which varies from one dataset to another. The number of features in the ten datasets ranges from about 2000 to 12 000, leading to very different sizes of search space. Therefore, we set the population size as the number of features divided by 20 with a restriction to 300 due to the limited computer memory. The maximum number of iterations was set to 70; however, an early stop was also applied when $gbest$ is not improved after ten iterations. The scaling criterion in PPSO is when $gbest$ is not improved after ten iterations and the current $gbest$ fitness is better than the previous $gbest$ fitness for at least 1%. PPSO starts with the initial size of 150 which is suggested in [56] for multiclass datasets. However, based on our experiments, this value is also a good initial size for binary class problems

TABLE II
PSO PARAMETER SETTINGS

Parameters	Settings
Population Size	#features/20 (restriction to 300)
Maximum iterations	70
Stopping criterion	<i>gbest</i> not improved for 10 iterations
Communication topology	Fully connected
Scaling criterion (PPSO only)	<i>gbest</i> not improved for 10 iterations and the current <i>gbest</i> fitness is better than the previous <i>gbest</i> fitness at least 1%

since PSO is able to choose an appropriate feature subset size during the evolutionary process. This initial size increases 50 every time the scaling criterion is met.

4) *Experimental Configuration*: As a wrapper approach, PPSO can use the classification performance of any learning algorithm to evaluate particles. In this paper, KNN with $k = 1$ was used because it is simple, fast, and nonparametric.

Since these datasets have small sample sizes, we used ten-fold cross validation (CV) to generate training and test set. In each CV, onefold is used to form test set and the remaining ninefolds are used to form training set. Test set is used for performance evaluation of the discretization and FS solution produced by each method based on the training set. During the evolutionary process, an inner loop of tenfold CV on the training set is used for fitness evaluation. Therefore, each method comprises of two loops of CV as recommended in [18] to avoid *FS bias*.

To eliminate statistical variations, each method was run 30 times with different random seeds for each dataset. Because each dataset was split into training and test set using ten-fold CV, a total of 300 runs were executed for each method. Experiments were run on PC with Intel Core i7-4770 CPU @ 3.4 GHz and 8-GB memory. The results of 30 runs from each method were compared using the statistical Wilcoxon significance test with 5% significance level.

V. RESULTS AND DISCUSSION

Table III shows the results of PSO-FS, EPSO, and PPSO. The average feature subset size returned by each method over the 30 runs is shown in column “size.” The best, mean and standard deviation of KNN accuracies using “full” (i.e., all continuous features), or using the transformed data by each of the compared methods are shown in columns four and five. The reported results are test accuracies calculated based on (7). For each dataset, the best accuracy and the smallest size are highlighted in bold.

Column *S* displays the statistical Wilcoxon significance test results of the method in the corresponding row over PPSO. “+” or “−” means the result is significantly better or worse than PPSO. “=” means they have similar performance. In other words, the more −, the better the proposed approach. The same meaning of + and − will be used throughout this paper.

A. PPSO Versus Full

As can be seen from Table III that the average numbers of features selected by PPSO are significantly smaller than the total number of features. PPSO selects less than 1% of the total number of features on four datasets, less than 2% on five

TABLE III
AVERAGE TEST RESULTS

Dataset	Method	Size	Best	Mean (Std)	<i>S</i>
SRBCT	Full	2308.0		87.08	−
	PSO-FS	150.0	97.50	91.31 (2.71)	−
	EPSO	137.3	100.00	96.89 (1.64)	+
	PPSO	108.5	100.00	95.78 (1.96)	−
DLBCL	Full	5469.0		83.00	−
	PSO-FS	101.8	96.67	80.03 (6.13)	−
	EPSO	42.8	94.17	85.18 (5.46)	=
	PPSO	44.0	94.17	86.22 (3.58)	−
9Tumor	Full	5726.0		36.67	−
	PSO-FS	955.0	55.00	45.95 (4.93)	−
	EPSO	138.5	65.00	58.22 (3.12)	=
	PPSO	118.1	65.00	59.28 (2.08)	−
Leuk1	Full	5327.0		79.72	−
	PSO-FS	150.0	92.22	81.60 (4.72)	−
	EPSO	135.9	95.56	93.37 (1.83)	−
	PPSO	80.4	95.42	94.37 (1.36)	−
Brain1	Full	5920.0		72.08	−
	PSO-FS	317.3	78.75	71.00 (3.06)	−
	EPSO	150.7	79.17	72.79 (3.48)	=
	PPSO	73.4	82.08	74.40 (3.67)	−
Leuk2	Full	11225.0		89.44	−
	PSO-FS	150.0	93.89	86.11 (3.97)	−
	EPSO	139.9	94.44	89.93 (2.79)	−
	PPSO	86.7	100.00	96.74 (1.64)	−
Brain2	Full	10367.0		62.50	−
	PSO-FS	417.9	82.08	69.11 (5.89)	=
	EPSO	152.8	83.75	70.76 (5.30)	=
	PPSO	66.7	74.58	68.75 (4.24)	−
Prostate	Full	10509.0		85.33	−
	PSO-FS	777.4	90.33	85.20 (2.35)	−
	EPSO	54.9	90.33	83.74 (3.55)	−
	PPSO	65.6	95.17	91.82 (1.77)	−
Lung	Full	12600.0		78.05	−
	PSO-FS	686.2	85.73	81.72 (2.08)	+
	EPSO	150.8	85.58	80.60 (2.42)	=
	PPSO	203.0	84.11	79.38 (3.26)	−
11Tumor	Full	12533.0		71.42	−
	PSO-FS	1638.8	86.07	82.62 (1.70)	+
	EPSO	149.9	83.68	79.29 (2.11)	+
	PPSO	167.0	83.20	76.83 (2.91)	−

datasets and 4.6% on SRBCT. In general, PPSO achieves the smallest subsets on six datasets.

With the discretized and selected features, PPSO achieves significantly better classification performance than using all continuous features on all ten datasets. It increases more than 5% accuracy on seven out of ten datasets with the highest improvement of 23% on 9Tumor.

The results indicate that PPSO can produce a much more powerful and compact representation for high-dimensional datasets by simultaneously discretize and select relevant features in a single stage.

B. PPSO Versus PSO-FS

In terms of dimensionality reduction, feature subsets returned by PPSO are much smaller than those selected by PSO-FS which is the two-stage approach. In terms of classification accuracy, the PPSO transformed data outperforms PSO-FS on seven datasets. While PSO-FS degrades the classification performance on Brain1 and Prostate, PPSO still obtains better accuracies than using *full* on these dataset with about 11% and 8% the number of features selected by PSO-FS on these datasets, respectively. Similar pattern can be seen in DLBCL and Leuk2 datasets.

Results of 9Tumor dataset reveal the largest difference between two approaches. While the transformed dataset by

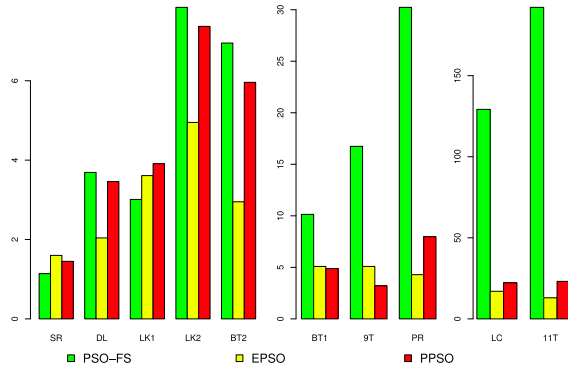


Fig. 6. Running time of PPSO.

PSO-FS improves 9% classification accuracy with 955 features, the transformed dataset by PPSO achieves 23% improvement using eight times less features than PSO-FS (118 features). A similar pattern is witnessed in the first six datasets.

In Brain2, PPSO obtains a similar accuracy to PSO-FS with a quarter of features selected by PSO-FS. In Lung and 11Tumor, PPSO obtains worse results than PSO-FS with about one third and one tenth number of features selected by PSO-FS, respectively.

In general, solutions evolved by the proposed one-stage approach obtain either a significantly better or similar classification accuracy to the two-stage approach on eight out of the ten datasets.

C. PPSO Versus EPSo

As can be seen from Table III that PPSO selects a smaller number of features than EPSo on six datasets while obtaining better or similar classification accuracy to EPSo on eight datasets. In Leuk2, while EPSo obtains a similar classification accuracy to *full* with 140 features, PPSO improves 7% accuracy with only 87 features.

The improvement of PPSO over EPSo may be resulted from several proposed strategies in PPSO. First of all, PPSO only finds an appropriate cut-point for each feature from the potentially good cut-points while EPSo chooses any possible cut-point within the range of the feature values. Therefore, PPSO has a smaller search space than EPSo. This enables PPSO to better cover the search space when using the same parameter setting as EPSo. Furthermore, the evolutionary process of PPSO is not only guided by accuracy of the transformed dataset but also the distance measure. This measure helps PPSO to differentiate feature subsets with the same training accuracy. Finally, the scaling mechanism enables PPSO to move the evolutionary process to larger feature subsets which may provide better solutions.

D. Computation Time

Fig. 6 compares the running time to complete a single run of PSO-FS, EPSo, and PPSO. The reported time (in minutes) is the average of 30 runs. Note that the three compared methods use the same wrapper approach with KNN for fitness evaluation. We also note that to evaluate each particle, PSO-FS

only needs to transform data based on the selected features, while PPSO needs to discretize and select features at the same time. This means that PPSO evaluation time is slightly longer than PSO-FS. Therefore, PPSO is intended to run longer than PSO-FS. However, an opposite trend is reported in Fig. 6.

Compared with PSO-FS, the running time of PPSO is shorter on eight datasets with about eight times shorter in 11Tumor dataset. As can be seen from Fig. 6 that the larger the dataset, the greater the ratio of PSO-FS's running time to PPSO's running time. An investigation of the evolutionary process reveals the main reason of this phenomenon. It is that the evolved feature subsets by PPSO are much smaller than those by PSO-FS, which in turn leads to a much smaller running time required by KNN in fitness evaluation process.

PPSO takes slightly longer time than EPSo on most datasets due to the scaling mechanism which introduces large feature subsets. It is noticed that although the fitness function of PPSO combines both wrapper and filter measures, the time spending for fitness evaluation is not longer than EPSo since both KNN and the filter measure are calculated based on the distance measure which is calculated once and used for both measures. Furthermore, since the number of features selected by PPSO is smaller, the test process will be faster as well.

E. PPSO Versus Traditional Methods

Table IV shows the compared results of PPSO with *full*, MDL+LFS, MDL+CON, MDL+CFS, and MChi2. Besides the size and time shown in the third and the last columns, the best and the mean of the training and test accuracies are shown under each corresponding column. The " S_{Tr} " and " S_{Te} " display the Wilcoxon significance test results of the corresponding method over PPSO in terms of training and test accuracy, respectively. The result of MChi2 on the Lung dataset is unavailable due to the out of memory error. Since CFS method is quite expensive, it fails to finish its run in 12 h for the two largest datasets which are Lung and 11Tumor with more than 12 000 features.

As can be seen from Table IV that PPSO outperforms MDL+LFS and MDL+CON on all the ten datasets in terms of training accuracy and nine datasets in terms of test accuracy. Compared with MDL+LFS, PPSO achieves 6%–19% higher test accuracy on all datasets except for the Lung dataset where both obtain similar accuracy. However, the best test accuracy achieved by PPSO is 4% higher than MDL+LFS on this dataset. Similarly, PPSO achieves 5%–31% higher test accuracy than MDL+CON on all datasets except for DLBCL where MDL+CON achieves the highest average test result. Although these two methods obtain the smallest feature subsets, their test accuracies are usually the lowest among the compared methods. Compared with the Full accuracy, MDL+LFS even has 13% lower on Brain1 and MDL+CON has 18% lower on 11Tumor. In contrast, PPSO achieves better accuracy than the full feature sets on all datasets.

Compared with MDL+CFS, PPSO obtains better or similar training accuracies on the eight datasets. For testing, PPSO achieves better results than MDL+CFS on three datasets, namely 9Tumor, Leuk1, and Prostate. For the other five

TABLE IV
COMPARISON WITH TRADITIONAL METHODS

Dataset	Method	Size	Training			Test			Time(s)
			Best	Mean	S_{Tr}	Best	Mean	S_{Te}	
SRBCT	Full	2,308.0		83.35	—		87.08	—	
	MDL+LFS	6.1		98.19	—		88.75	—	28.78
	MDL+CON	4.3		97.71	—		85.83	—	3.24
	MDL+CFS	80.9		100.00	=		100.00	+	318.04
	MChi2	85.9		100.00	=		100.00	+	83.07
	PPSO	108.5	100.00	100.00		100.00	95.78		87.14
DLBCL	Full	5,469.0		81.71	—		83.00	—	
	MDL+LFS	4.0		98.24	—		74.00	—	73.78
	MDL+CON	3.4		98.14	—		92.50	+	6.29
	MDL+CFS	58.0		99.22	—		91.67	+	1310.49
	MChi2	10.2		87.96	—		75.50	—	369.45
	PPSO	44.0	100.00	100.00		94.17	86.22		207.40
9Tumor	Full	5,726.0		33.44	—		36.67	—	
	MDL+LFS	12.6		82.39	—		41.67	—	64.20
	MDL+CON	7.6		71.61	—		28.33	—	12.65
	MDL+CFS	38.0		90.71	—		53.33	—	656.21
	MChi2	58.5		77.28	—		48.33	—	260.70
	PPSO	118.1	92.87	92.31		65.00	59.28		192.68
Leuk1	Full	5,327.0		79.77	—		79.72	—	
	MDL+LFS	4.8		99.17	—		81.39	—	70.48
	MDL+CON	3.0		98.67	—		89.17	—	5.69
	MDL+CFS	56.0		100.00	=		93.19	—	1358.35
	MChi2	46.4		98.80	—		92.08	—	276.35
	PPSO	80.4	100.00	100.00		95.42	94.37		236.56
Brain1	Full	5,920.0		65.07	—		72.08	—	
	MDL+LFS	9.9		89.13	—		59.17	—	104.41
	MDL+CON	6.2		79.99	—		55.42	—	14.51
	MDL+CFS	115.4		99.93	—		79.58	+	2859.76
	MChi2	290.0		80.63	—		74.58	=	438.82
	PPSO	73.4	100.00	99.99		82.08	74.40		292.84
Leuk2	Full	11,225.0		88.82	—		89.44	—	
	MDL+LFS	4.3		99.08	—		90.00	—	231.15
	MDL+CON	3.0		99.44	—		85.56	—	17.54
	MDL+CFS	79.0		100.00	=		98.89	+	5407.43
	MChi2	166.1		99.56	—		93.33	—	904.96
	PPSO	86.7	100.00	100.00		100.00	96.74		441.95
Brain2	Full	10,367.0		63.52	—		62.50	—	
	MDL+LFS	5.6		98.80	—		53.33	—	179.83
	MDL+CON	4.7		89.92	—		61.67	—	19.33
	MDL+CFS	63.4		100.00	=		71.25	+	3304.74
	MChi2	160.7		84.00	—		70.00	=	562.42
	PPSO	66.7	100.00	99.99		74.58	68.75		357.34
Prostate	Full	10,509.0		82.08	—		85.33	—	
	MDL+LFS	4.9		82.44	—		73.17	—	227.68
	MDL+CON	4.7		98.46	—		70.50	—	23.61
	MDL+CFS	51.6		98.12	—		90.17	—	2694.27
	MChi2	33.6		95.44	—		86.17	—	1246.46
	PPSO	65.6	100.00	99.84		95.17	91.82		478.17
Lung	Full	12,600.0		71.59	—		78.05	—	
	MDL+LFS	12.2		95.12	—		80.55	=	685.78
	MDL+CON	6.6		89.54	—		74.64	—	63.37
	MDL+CFS	NA		NA	—		NA	—	NA
	MChi2	NA		NA	—		NA	—	NA
	PPSO	203.0	97.67	97.09		84.11	79.38		1335.21
11Tumor	Full	12,533.0		71.01	—		71.42	—	
	MDL+LFS	14.3		79.96	—		61.71	—	555.99
	MDL+CON	9.4		72.58	—		53.83	—	85.41
	MDL+CFS	NA		NA	—		NA	—	NA
	MChi2	2098.0		93.32	—		84.54	+	3125.14
	PPSO	167.0	99.47	99.14		83.20	76.83		1387.01

datasets, MDL+CFS has better results than PPSO on average but the best accuracy of PPSO is almost always better than MDL+CFS. It is noticed that the computation time of MDL+CFS quickly increases with the number of features. Therefore, scalability is a major concern when applying this method on high-dimensional data, such as Lung and 11Tumor.

Similarly, in terms of training accuracy, PPSO also outperforms MChi2 on all datasets. In terms of test accuracy, PPSO performs either similar to or significantly better than MChi2 on seven datasets with smaller feature subsets on four datasets. It is noticed that although MChi2 is a filter method, its running time is longer than that of PPSO and MDL+LFS on almost all datasets. While the difference in running time

TABLE V
AVERAGE TRAINING ACCURACY

Dataset	Method	Train-Acc(Std)	T	Dataset	Method	Train-Acc(Std)	T
SRBCT	PSO-FS	100.00 (0.00)	=	Leuk2	PSO-FS	100.00 (0.00)	=
	EPSO	100.00 (0.00)	=		EPSO	100.00 (0.00)	=
	PPSO	100.00 (0.00)	=		PPSO	100.00 (0.00)	=
DLBCL	PSO-FS	100.00 (0.00)	=	Brain2	PSO-FS	99.73 (0.12)	—
	EPSO	100.00 (0.00)	=		EPSO	98.38 (0.27)	—
	PPSO	100.00 (0.00)	=		PPSO	99.99 (0.05)	—
9Tumor	PSO-FS	97.49 (0.23)	+	Prostate	PSO-FS	98.89 (0.10)	—
	EPSO	95.03 (0.22)	+		EPSO	98.56 (0.14)	—
	PPSO	92.31 (0.33)	+		PPSO	99.84 (0.5)	—
Leuk1	PSO-FS	100.00 (0.00)	=	Lung	PSO-FS	97.77 (0.05)	+
	EPSO	100.00 (0.00)	=		EPSO	97.10 (0.14)	=
	PPSO	100.00 (0.00)	=		PPSO	97.09 (0.30)	=
Brain1	PSO-FS	100.00 (0.00)	=	11Tumor	PSO-FS	99.80 (0.08)	+
	EPSO	99.33 (0.29)	—		EPSO	96.21 (0.19)	—
	PPSO	99.99 (0.06)	—		PPSO	99.14 (0.20)	—

between MChi2 and PPSO is not too big on small datasets, it becomes much larger on datasets with more than 10 000 features. This indicates that the proposed algorithm is scalable to high-dimensional datasets.

VI. FURTHER ANALYSIS

In addition to classification performance and feature subset size, generalization and robustness are also important criteria in evaluating the performance of an FS method. While generalizability relates to the ability of the learned model to have a similar performance on both training and test data, robustness relates to the ability to reproduce the results regardless of the small variance of the experiments. In this section, we compared the performance of the one-stage and two-stage approaches in terms of these two aspects. Furthermore, we also compared the performance of PPSO and the traditional ones in the *FS bias* scenario. By using the whole dataset for FS training, the performance difference between them reveals their learning capability.

A. Generalization and Robustness

To compare the generalizability of the one-stage and two-stage approaches, we examined the training accuracy of the returned solutions. Table V shows the mean and standard deviation of the training accuracy obtained by the three methods in 30 runs of each dataset.

As can be seen in Table V that PPSO achieves better training accuracy than PSO-FS on two datasets and similar on five. However, as seen in Table III, PPSO achieves significantly better test accuracy on seven datasets than PSO-FS and similar on one. This indicates that PPSO has better generalizability than PSO-FS.

Compared with EPSO, PPSO obtains the same training accuracy (100%) on SRBCT with a smaller number of features and 3% higher accuracy on 11Tumor with a bigger feature subset. This indicates that the scaling mechanism helps PPSO effectively select appropriate feature subsets that can maintain or increase the training accuracy. However, these solutions seem to overfit the training data when they get 1% and 3% lower test accuracy than those of EPSO, respectively, as shown in Table III. Another possible reason is that PPSO uses the fix potential cut-points calculated from the training data which

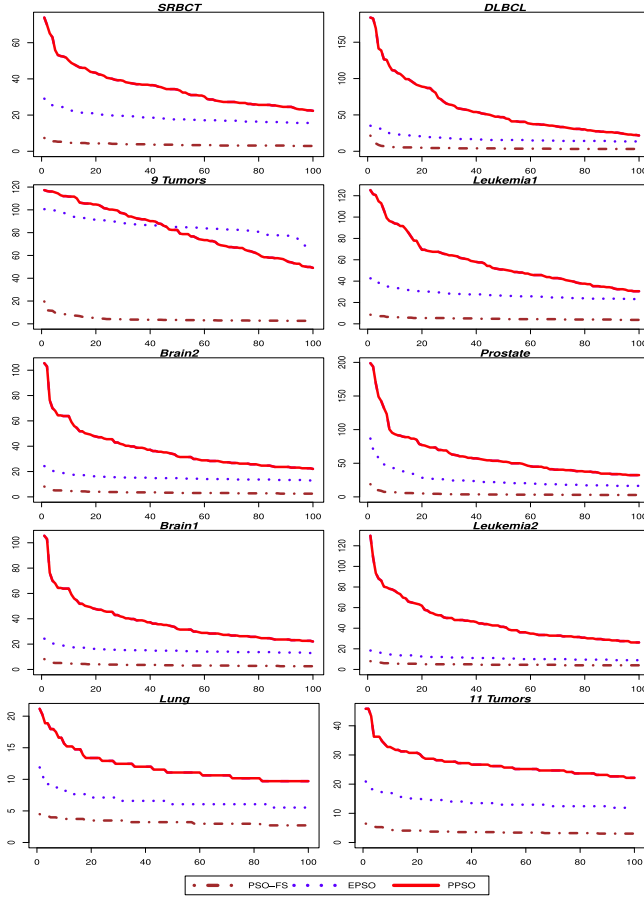


Fig. 7. Z-score of the top 100 selected features on each dataset.

may not be representative to the test data. A better mechanism to calculate the potential cut-points may help to improve the performance of PPSO.

We also note a big difference between the test and training accuracy (shown in Tables III and V), especially in 9Tumor. This indicates the existence of overfitting which may be because the skew distribution of features in these datasets creates different distributions in training and test sets. Therefore, the learned model may not be generalized well to the test data. This problem is worse in datasets with smaller numbers of samples, such as Brain2 and 9Tumor with 50 and 60 instances. On top of this, the large number of classes as well as the class imbalance nature are additional challenges of these datasets. With nine classes, 9Tumor has worse results than Brain2 which has four classes. As can be seen from Table IV that MDL+LFS and MChi2 performances are also affected by this phenomenon. The gaps between training and test accuracy obtained by MDL+LFS on 9Tumor and Brain2 are about 40% which is even larger than PPSO with about 30%.

Since each method was run 30 times on each tenfold CV, it produced 300 solutions for a dataset. If the dataset has redundant features, different solutions may include very different features if the method is sensitive to the experimental conditions, leading to a low selection frequency of each feature. Therefore, we used the Z-score [56], [60] of 100 most selected features to examine the robustness of the compared methods.

Z-score is a measure indicating the significance of the selection frequency of a feature. It is defined as follows:

$$Z_x = \frac{f_x - \mu}{\sigma} \quad (12)$$

where f_x is the number of times feature x being selected in the L solutions (L is 300 in our experiment). μ and σ are the mean and standard deviation of f_x . They are estimated based on the probability of feature x being selected, $P(f_x)$, as follows:

$$\mu = P(f_x) \cdot L$$

and

$$\sigma = \sqrt{P(f_x) \cdot (1 - P(f_x)) \cdot L}$$

where $P(f_x)$ is equal to the average number of selected features in L solutions divided by the size of the original feature set. Therefore, the higher the value of Z_x , the less likely that feature x is selected by chance. Therefore, an algorithm is said to be more robust if it selects more features with higher Z-scores.

Fig. 7 compares the Z-scores of the top 100 selected features by the three methods on each dataset. The chart shows that the Z-scores of the features selected by PPSO are greater than those of EPSO which are in turn greater than PSO-FS. This indicates that PPSO is more robust than the other methods.

In general, the results show that PPSO has better generalizability and robustness than PSO-FS. Solutions produced by PPSO are more compact and have higher discriminating power than PSO-FS. This demonstrates our hypothesis that important information including feature interaction may be lost during the discretization process in the first stage of PSO-FS. Since the two-stage approach evaluates the selected features and their cut-points simultaneously, it takes these important information into account.

B. Learning Capability

To confirm the performance of the proposed approach, we conducted another experiment with *FS bias*. In this experiment, the whole dataset is used to train the algorithm. Based on the evolved cut-points and the selected features, the whole dataset is transformed and used for performance evaluation. Besides KNN, naive Bayes (NB) is also used to see if the evolved solutions by PPSO can also improve the performance of classifiers other than the one used in the fitness function.

Table VI shows the average of tenfold CV classification accuracies obtained by KNN ($k = 1$) and NB on the original datasets (*full*) and the transformed datasets by five methods. Wilcoxon significance test with significance level of 0.05 was used to compare the classification accuracy obtained by KNN and NB between the corresponding method and PPSO. “ S_K ” and “ S_{NB} ” present the statistical test results of the corresponding method over PPSO using KNN and NB, respectively.

According to Table VI, PPSO obtains 100% accuracy in all or almost all 30 runs on seven datasets for KNN and six datasets for NB. For KNN, its results outperform MDL+LFS, MDL+CON, and MChi2 on 8, 9, and 7 datasets, respectively. It is noticed that MDL+LFS performs quite well on datasets with a small number of classes; however, its accuracy is 20%

TABLE VI
RESULTS WITH FS BIAS

Dataset	Method	Size	Time (s)	KNN			Naive Bayes		
				Best	Mean	S_K	Best	Mean	S_{NB}
SRBCT	Full	2,308.0			83.35	–		99.17	–
	MDL+LFS	8.0	5.52		99.14	–		99.14	–
	MDL+CON	4.0	0.53		95.18	–		95.18	–
	MDL+CFS	80.0	36.53		100.00	=		100.00	=
	MChi2	120.0	9.80		100.00	=		100.00	=
	PPSO	114.4	9.51	100.00	99.97		100.00	100.00	
DLBCL	Full	5,469.0			81.71	–		77.17	–
	MDL+LFS	9.0	11.09		97.37	–		97.41	–
	MDL+CON	4.0	0.64		96.10	–		96.10	–
	MDL+CFS	55.0	132.31		100.00	=		100.00	=
	MChi2	15.0	37.80		87.79	–		90.43	–
	PPSO	49.7	21.86	100.00	100.00		100.00	100.00	
9Tumor	Full	5,726.0			33.44	–		40.00	–
	MDL+LFS	13.0	6.09		74.49	–		65.70	–
	MDL+CON	6.0	0.88		68.33	–		68.33	–
	MDL+CFS	44.0	77.68		89.97	–		94.60	+
	MChi2	78.0	26.72		77.60	–		66.14	–
	PPSO	113.5	23.61	98.77	95.83		81.44	78.09	
Leuk1	Full	5,327.0			79.77	–		90.00	–
	MDL+LFS	7.0	7.40		96.30	–		100.00	=
	MDL+CON	4.0	0.63		98.61	–		98.61	–
	MDL+CFS	74.0	221.60		100.00	=		100.00	=
	MChi2	66.0	372.90		99.12	–		99.12	–
	PPSO	90.3	25.10	100.00	100.00		100.00	100.00	
Brain1	Full	5,920.0			65.07	–		77.50	–
	MDL+LFS	10.0	12.10		84.33	–		76.33	–
	MDL+CON	5.0	0.94		93.33	–		93.33	–
	MDL+CFS	103.0	385.85		100.00	=		100.00	=
	MChi2	325.0	130.96		81.00	–		83.00	–
	PPSO	83.9	31.87	100.00	100.00		100.00	99.88	
Leuk2	Full	11,225.0			88.82	–		95.00	–
	MDL+LFS	6.0	32.10		98.61	–		100.00	=
	MDL+CON	3.0	1.39		98.61	–		98.61	–
	MDL+CFS	87.0	1079.57		100.00	=		100.00	=
	MChi2	300.0	46.86		100.00	=		100.00	=
	PPSO	95.6	44.72	100.00	99.92		100.00	99.84	
Brain2	Full	10,367.0			63.52	–		66.67	–
	MDL+LFS	6.0	26.36		100.00	=		100.00	=
	MDL+CON	3.0	1.24		94.00	–		94.00	–
	MDL+CFS	76.0	756.38		100.00	=		100.00	=
	MChi2	110.0	64.52		94.64	–		92.86	–
	PPSO	82.1	36.63	100.00	100.00		100.00	99.94	
Prostate	Full	10,509.0			82.08	–		62.17	–
	MDL+LFS	12.0	20.80		96.04	–		98.04	+
	MDL+CON	4.0	1.78		100.00	+		100.00	=
	MDL+CFS	55.0	532.78		98.00	–		99.04	+
	MChi2	50.0	30.90		94.08	–		95.15	–
	PPSO	70.8	51.58	100.00	99.30		99.04	96.63	
Lung	Full	12,600.0			71.59	–		79.98	–
	MDL+LFS	12.0	87.10		96.77	=		95.10	+
	MDL+CON	5.0	5.40		93.60	–		93.60	+
	MDL+CFS	286.0	17568.40		98.76	+		99.86	+
	MChi2	NA	NA		NA			NA	
	PPSO	218.4	175.81	97.90	96.65		93.13	84.88	
11Tumor	Full	12,533.0			71.01	–		77.47	–
	MDL+LFS	12.0	41.95		78.58	–		78.18	–
	MDL+CON	7.0	6.57		72.99	–		72.99	–
	MDL+CFS	207.0	7911.05		98.89	+		100.00	+
	MChi2	2643.0	100.07		94.16	–		96.29	=
	PPSO	173.5	162.15	99.66	98.44		98.23	96.10	

lower than that of PPSO on 9Tumor and 11Tumor datasets. MDL+CON also suffers from this problem with an even a bigger gap. Compared with MDL+CFS, PPSO obtains a significantly better results on two datasets, similar in 6 and worse in 2. It is also noticed that even though MDL+CFS achieves comparable performance with PPSO, its running time is 4–100 times longer than that of PPSO. In general, with totally 49 comparisons (excluding the unavailable result of MChi2 on Lung dataset), PPSO wins 36 cases, draws 10 and loses 3.

Similarly, the transformed datasets by PPSO also help NB improve its classification accuracy on all datasets, six of which reach 100% accuracy in all or almost all 30 runs. Compared with feature subsets obtained by MDL+LFS, MDL+CON, and MChi2, those of PPSO achieve either similar or better

classification performance on eight to nine datasets. Compared with MDL+CFS, PPSO obtains similar performance on six datasets and worse in the remaining four. On Brain1, while MDL+LFS obtains 76.33% with 10 features and MChi2 obtains 83% with 325 features, PPSO only selects 84 features to improve the NB accuracy to 99.88%. Using over ten times longer time than PPSO, MDL+CFS obtains 100% accuracy on this dataset, however, with 105 features. In general, for NB, PPSO wins 29 cases, draws 13 and loses 7 in the total 49 comparisons.

It is also noticed that the result subsets of PPSO and MDL+CFS on 9Tumor dataset have opposite effects on KNN and NB. While the transformed dataset of PPSO performs better than that of MDL+CFS in KNN, it obtains worse result than MDL+CFS in NB. Given that PPSO selects many more features than MDL+CFS in this dataset, this phenomenon might have happened because the feature subset evolved by PPSO still contains redundant or correlated features which may not strongly affect the performance of KNN but may negatively affect the performance of NB. Since PPSO uses KNN as a learning algorithm to evaluate the feature subset, the redundant features are not well identified. On the other hand, CFS is good at recognizing correlated or redundant features. This possible reason also explains the loses of PPSO on the remaining three datasets, namely Prostate, Lung, and 11Tumor.

VII. CONCLUSION

The goal of this paper was to propose an integrating approach to discretization and FS in a single stage using BBPSO. The goal was achieved by proposing a new PSO-based methods, PPSO, with a new PSO representation to choose cut-points for discretizing multiple features and selecting features simultaneously. PPSO was compared with using the full set of original features, EPSO, and the two-stage approach (PSO-FS).

Experimental results on ten datasets having thousands to tens of thousands of features with variant numbers of classes show that PPSO can discretize multiple features simultaneously and select a much smaller number of relevant features with better discriminating ability. Comparison between PPSO and PSO-FS suggests that it is more effective to combine discretization and FS in a single stage. Compared with EPSO, PPSO obtains either equivalent or better results with smaller numbers of features. Further analysis also shows that PPSO is more general and more robust than the compared PSO methods.

PPSO was also compared with four traditional methods representing for two-stage and one-stage approaches, MDL+LFS, MDL+CON, MDL+CFS, and MChi2. The results of two experiments with and without *FS bias* show that PPSO has significantly better performance than MDL+LFS, MDL+CON, and MChi2 and similar performance to MDL+CFS in most cases. The results also show that PPSO is more scalable than MDL+CFS and MChi2 in dealing with high-dimensional problems. Compared results on both KNN and NB indicate that solutions obtained by PPSO can be generalized to other classifier than the one used during the training process.

As a binary discretization method, PPSO may not work well on data that needs to be discretized into multiple intervals.

Furthermore, the potential cut-points are calculated based on MDLP. Therefore, investigation of using other schemes for cut-point calculation, multiple interval discretization, and different approaches to optimizing the search process will be included in our future work.

Empirical results show that the running time of PPSO is scalable to high-dimensional data with 10000+ features. However, since the PPSO representation is static and proportional to the feature set size, applying PPSO to datasets with 100000+ features may be limited by the memory capacity. Furthermore, the search space will become much larger due to the exponential increase of the possible solutions, so it may require a different way to solve the problem. In the future, we will investigate using a dynamic representation in PPSO to overcome this limitation.

REFERENCES

- [1] A. J. Ferreira and M. A. T. Figueiredo, "Efficient feature selection filters for high-dimensional data," *Pattern Recognit. Lett.*, vol. 33, no. 13, pp. 1794–1804, 2012.
- [2] B. Tran, B. Xue, and M. Zhang, "Improved PSO for feature selection on high-dimensional datasets," in *Simulated Evolution and Learning (LNCS 8886)*. Dordrecht, New Zealand: Springer Int., 2014, pp. 503–515.
- [3] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *J. Bioinform. Comput. Biol.*, vol. 3, no. 2, pp. 185–205, 2005.
- [4] J. Dougherty, R. Kohavi, and M. Sahami, "Supervised and unsupervised discretization of continuous features," in *Proc. 12th Int. Conf. Mach. Learn.*, vol. 12. San Francisco, CA, USA, 1995, pp. 194–202.
- [5] V. Bolón-Canedo, N. Sánchez-Marroño, and A. Alonso-Betanzos, "On the effectiveness of discretization on gene selection of microarray data," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, 2010, pp. 1–8.
- [6] A. J. Ferreira and M. A. T. Figueiredo, "An unsupervised approach to feature discretization and selection," *Pattern Recognit.*, vol. 45, no. 9, pp. 3048–3060, 2012.
- [7] S. Chao and Y. Li, "Multivariate interdependent discretization for continuous attribute," in *Proc. 3rd Int. Conf. Inf. Technol. Appl.*, vol. 1. Sydney, NSW, Australia, 2005, pp. 167–172.
- [8] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach Human Sci.*, Nagoya, Japan, 1995, pp. 39–43.
- [9] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.
- [10] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Appl. Soft Comput.*, vol. 18, pp. 261–276, May 2014.
- [11] L. Cervante, B. Xue, M. Zhang, and L. Shang, "Binary particle swarm optimisation for feature selection: A filter based approach," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Brisbane, QLD, Australia, 2012, pp. 881–888.
- [12] M. S. Mohamad, S. Omatu, S. Deris, and M. Yoshioka, "A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 6, pp. 813–822, Nov. 2011.
- [13] W. Zhou and J. A. Dickerson, "A novel class dependent feature selection method for cancer biomarker discovery," *Comput. Biol. Med.*, vol. 47, pp. 66–75, Apr. 2014.
- [14] B. Tran, B. Xue, and M. Zhang, "Bare-bone particle swarm optimisation for simultaneously discretising and selecting features for high-dimensional classification," in *Proc. Eur. Conf. Appl. Evol. Comput. (EvoApplications)*, Porto, Portugal, 2016, pp. 701–718.
- [15] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp. (SIS)*, Indianapolis, IN, USA, 2003, pp. 80–87.
- [16] U. M. Fayyad and K. B. Irani, "Multi-interval discretization of continuous-valued attributes for classification learning," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, vol. 2. Chambéry, France, 1993, pp. 1022–1027.
- [17] S. K. Singhi and H. Liu, "Feature subset selection bias for classification learning," in *Proc. 23rd Int. Conf. Mach. Learn.*, Pittsburgh, PA, USA, 2006, pp. 849–856.
- [18] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, nos. 1–2, pp. 273–324, 1997.
- [19] J. Liang, S. Yang, and A. Winstanley, "Invariant optimal feature selection: A distance discriminant and feature ranking based solution," *Pattern Recognit.*, vol. 41, no. 5, pp. 1429–1439, 2008.
- [20] B. Xue, L. Cervante, L. Shang, W. Browne, and M. Zhang, "A multi-objective particle swarm optimisation for filter-based feature selection in classification problems," *Connection Sci.*, vol. 24, nos. 2–3, pp. 91–116, 2012.
- [21] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artif. Intell.*, vol. 151, nos. 1–2, pp. 155–176, 2003.
- [22] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proc. 17th Int. Conf. Mach. Learn.*, Stanford, CA, USA, 2000, pp. 359–366.
- [23] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.
- [24] T. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, 1997.
- [25] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
- [26] T. Butler-Yeoman, B. Xue, and M. Zhang, "Particle swarm optimisation for feature selection: A hybrid filter-wrapper approach," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 2428–2435.
- [27] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, nos. 1–4, pp. 131–156, 1997.
- [28] M. Gutlein, E. Frank, M. Hall, and A. Karwath, "Large-scale attribute selection using wrappers," in *Proc. IEEE Symp. Comput. Intell. Data Min.*, Nashville, TN, USA, 2009, pp. 332–339.
- [29] M. Hall *et al.*, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
- [30] A. W. Whitney, "A direct method of nonparametric measurement selection," *IEEE Trans. Comput.*, vol. C-20, no. 9, pp. 1100–1103, Sep. 1971.
- [31] T. Marill and D. M. Green, "On the effectiveness of receptors in recognition systems," *IEEE Trans. Inf. Theory*, vol. 9, no. 1, pp. 11–17, Jan. 1963.
- [32] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, Mar. 2003.
- [33] H. B. Nguyen, B. Xue, I. Liu, and M. Zhang, "Filter based backward elimination in wrapper based PSO for feature selection in classification," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 3111–3118.
- [34] H. Liu and R. Setiono, "Feature selection and classification—A probabilistic wrapper approach," in *Proc. 9th Int. Conf. Ind. Eng. Appl. AI ES*, 1997, pp. 419–424.
- [35] A. Unler and R. B. C. A. Murat, "mr2PSO: A maximum relevance minimum redundancy feature selection method based on swarm intelligence for support vector machine classification," *Inf. Sci.*, vol. 181, no. 20, pp. 4625–4641, 2011.
- [36] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, Aug. 2016.
- [37] B. Tran, B. Xue, and M. Zhang, "Overview of particle swarm optimisation for feature selection in classification," in *Simulated Evolution and Learning (LNCS 8886)*. Dordrecht, New Zealand: Springer Int., 2014, pp. 605–617.
- [38] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 4, pp. 734–750, Apr. 2013.
- [39] H. Liu, F. Hussain, C. L. Tan, and M. Dash, "Discretization: An enabling technique," *Data Min. Knowl. Disc.*, vol. 6, no. 4, pp. 393–423, 2002.
- [40] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 32, no. 1, pp. 47–58, 2006.
- [41] J. Catlett, "On changing continuous attributes into ordered discrete attributes," in *Machine Learning: Proceedings of European Working Session on Learning (EWSL)*. Porto, Portugal: Springer Berlin Heidelberg, 1991, pp. 164–178.
- [42] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," *Mach. Learn.*, vol. 11, no. 1, pp. 63–90, 1993.
- [43] J. W. Grzymala-Busse, "Discretization based on entropy and multiple scanning," *Entropy*, vol. 15, no. 5, pp. 1486–1502, 2013.

- [44] A. Cano, D. T. Nguyen, S. Ventura, and K. J. Cios, "ur-CAIM: Improved CAIM discretization for unbalanced and balanced data," *Soft Comput.*, vol. 20, no. 1, pp. 173–188, 2016.
- [45] P. Yang, J.-S. Li, and Y.-X. Huang, "HDD: A hypercube division-based algorithm for discretisation," *Int. J. Syst. Sci.*, vol. 42, no. 4, pp. 557–566, 2011.
- [46] J. L. Flores, I. Inza, and P. Larrañaga, "Wrapper discretization by means of estimation of distribution algorithms," *Intell. Data Anal.*, vol. 11, no. 5, pp. 525–545, 2007.
- [47] S. Ramirez-Gallego, S. García, J. M. Benitez, and F. Herrera, "Multivariate discretization based on evolutionary cut points selection for classification," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 595–608, Mar. 2016.
- [48] P. Mahanta, H. A. Ahmed, J. K. Kalita, and D. K. Bhattacharyya, "Discretization in gene expression data analysis: A selected survey," in *Proc. 2nd Int. Conf. Comput. Sci. Eng. Inf. Technol.*, Coimbatore, India, 2012, pp. 69–75.
- [49] H. Liu and R. Setiono, "Chi2: Feature selection and discretization of numeric attributes," in *Proc. TAI*, Herndon, VA, USA, 1995, pp. 388–391.
- [50] F. E. H. Tay and L. Shen, "A modified Chi2 algorithm for discretization," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 3, pp. 666–670, May/Jun. 2002.
- [51] L. J. Sheela and D. V. Shanthi, "An approach for discretization and feature selection of continuous-valued attributes in medical images for classification learning," *Int. J. Comput. Theory Eng.*, vol. 1, no. 2, pp. 154–158, 2009.
- [52] A. Ferreira and M. Figueiredo, "Unsupervised joint feature discretization and selection," in *Pattern Recognition and Image Analysis*. Las Palmas de Gran Canaria, Spain: Springer Berlin Heidelberg, 2011, pp. 200–207.
- [53] F. Van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, 2006.
- [54] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Comput. Elect. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.
- [55] Y. Zhang, D. Gong, Y. Hu, and W. Zhang, "Feature selection algorithm based on bare bones particle swarm optimization," *Neurocomputing*, vol. 148, pp. 150–157, Jan. 2015.
- [56] Z. Zhu, Y.-S. Ong, and M. Dash, "Markov blanket-embedded genetic algorithm for gene selection," *Pattern Recognit.*, vol. 40, no. 11, pp. 3236–3248, 2007.
- [57] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data," in *Advances in Artificial Intelligence*. Gold Coast, Australia: Springer Berlin Heidelberg, 2007, pp. 769–775.
- [58] H. Al-Sahaf, M. Zhang, M. Johnston, and B. Verma, "Image descriptor: A genetic programming approach to multiclass texture classification," in *Proc. IEEE Congr. Evol. Comput.*, Sendai, Japan, 2015, pp. 2460–2467.
- [59] J. Alcalá et al., "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple Valued Logic Soft Comput.*, vol. 17, no. 2–3, pp. 255–287, 2011.
- [60] T. Jirapech-Umpai and S. Aitken, "Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes," *BMC Bioinform.*, vol. 6, no. 1, p. 148, 2005.



Binh Tran (S'14) received the B.E. degree in computer science from Can Tho University, Can Tho, Vietnam, in 1998, and the master's degree in applied computer science from the Free University of Brussels, Brussels, Belgium, in 2002. She is currently pursuing the Ph.D. degree with the School of Engineering and Computer Science, Victoria University of Wellington (VUW), Wellington, New Zealand.

Since 2014, she has been with the Evolutionary Computation Research Group, VUW. His current research interests include evolutionary computation, feature manipulation, such as feature selection and construction, high-dimensional data, and machine learning.

Ms. Tran has been a Reviewer for international journals and conferences in this research fields. She is a member of the IEEE Computational Intelligence Society.



Bing Xue (M'10) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington (VUW), Wellington, New Zealand, in 2014.

She is currently a Senior Lecturer with the School of Engineering and Computer Science, VUW. She has over 100 papers published in fully referred international journals and conferences. Her current

research interests include evolutionary computation, feature selection, feature construction, multiobjective optimization, data mining, and machine learning.

Dr. Xue is an Associate Editor/member of Editorial Board for five international journals, including the *IEEE Computational Intelligence Magazine*, *Applied Soft Computing*, the *International Journal of Swarm Intelligence*, the *International Journal of Computer Information Systems*, and *Industrial Management Applications*. She is a Guest Editor of the Special Issue on Evolutionary Feature Reduction and Machine Learning for the *Journal of Soft Computing* (Springer). She is also a Guest Editor of *Evolutionary Image Analysis*, *Pattern Recognition*, and the *Journal of Applied Soft Computing*. She is serving as a Reviewer of over 20 international journals, including the IEEE TRANSACTIONS ON CYBERNETICS and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. She has been the Chair for a number of international conferences, including the Leading Chair of the IEEE Symposium on Computational Intelligence in Feature Analysis, Selection, and Learning in Image and Pattern Recognition at Symposium Series on Computational Intelligence 2016 and 2017, the Program Co-Chair of the 31st Australasian AI 2018, Australasian Conference on Artificial Life and Computational Intelligence 2018, and the 7th Soft Computing and Pattern Recognition 2015, the Special Session Chair for Intelligent and Evolutionary Systems 2016, the Tutorial Chair for the 30th Australasian AI, and the Publicity Chair for Simulated Evolution and Learning (SEAL) 2017 and International Conference on Data Intelligence and Security 2018. She is an Organizer of the Special Session on Evolutionary Feature Selection and Construction in the 2015–2017 IEEE Congress on Evolutionary Computation and SEAL 2014 and 2017. She is currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction and the IEEE CIS Graduate Student Research Grants Committee.



Mengjie Zhang (M'04–SM'10) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand, where he is currently a Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research

and Innovation) of the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 350 research papers in refereed international journals and conferences.

Prof. Zhang has been an Associate Editor or an Editorial Board Member for ten international journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, the *Evolutionary Computation Journal*, and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, and a Reviewer of over 20 international journals. He has been a Steering Committee Member and a Program Committee Member for over 100 international conferences. He has supervised over 50 postgraduate research students. He is the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, a Vice Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing and the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction, the Founding Chair for IEEE Computational Intelligence Chapter in New Zealand, and a member of the IEEE CIS Evolutionary Computation Technical Committee and the IEEE CIS Task Force of Hyper-Heuristics.