A PSO Based Hybrid Feature Selection Algorithm for High-Dimensional Classification

Binh Tran, Mengjie Zhang, and Bing Xue
Evolutionary Computation Research Group
Victoria University of Wellington, PO Box 600, Wellington, New Zealand
Email: binh.tran,mengjie.zhang,bing.xue@ecs.vuw.ac.nz

***Abstract*—Recent research has shown that Particle Swarm Optimisation is a promising approach to feature selection. However, applying it on high-dimensional data with thousands to tens of thousands of features is still challenging because of the large search space. While filter approaches are time efficient and scalable for high-dimensional data, they usually obtain lower classification accuracy than wrapper approaches. On the other hand, wrapper methods require a longer running time than filter methods due to the learning algorithm involved in fitness evaluation. This paper proposes a new strategy of combining filter and wrapper approaches in a single evolutionary process in order to achieve smaller feature subsets with better classification performance in a shorter time. A new local search heuristic using symmetric uncertainty is proposed to refine the solutions found by PSO. The proposed method is examined and compared with three recent PSO based methods on eight high-dimensional problems of varying difficulty. The results show that the hybrid PSO is more effective and efficient than the other methods.**

I. Introduction

Feature selection (FS) is a data preprocessing technique used to reduce the dimensionality of the problem while maintain or increase the discriminating ability of the feature sets. FS has been applied to different machine learning tasks especially in classification where an instance is classified into its corresponding category. FS helps classification algorithms to obtain simpler classifiers with better performance in a shorter time [1]. It is considered as an essential step to deal with high-dimensional data with thousands to tens of thousands of features.These datasets usually include many irrelevant and redundant features which may significantly degrade the performance of learning algorithms.

Researchers have proposed a large number of FS methods which can be generally classified into wrapper and filter approaches [1]. Filter methods evaluate features based on their intrinsic characteristics. On the other hand, wrapper methods use classification accuracy of a learnt classifier to evaluate the feature subset. Although wrapper methods are typically less time efficient than filter methods, they usually achieve better classification performance than filters.

Although FS has been studied for decades, applying it on high-dimensional data is still challenging since exhaustive search for an optimal solution in 2^n possible solutions for a dataset with n features is impractical.

Feature ranking or feature weighting is one of the scalable approaches to FS on high-dimensional data. In these methods, features are individually ranked based on their degrees of relevance using some measure such as distance [2], [3], and feature dispersion [4]. The top ranked features are used to form the final feature subset. Therefore, a certain domain knowledge is required to determine the number of features should be selected. Moreover, since features are evaluated individually, these methods cannot identify redundancy and interaction among features.

To address these limitations, a two-stage approach was proposed where feature ranking is performed following by an heuristic search to remove redundant features in the ranked list [5], [6]. In the second stage, a filter measure can be used to evaluate feature redundancy such as coefficient correlation [7] and symmetric uncertainty [6] or a wrapper measure such as classification accuracy [5]. In general, these methods have shown to be effective and efficient to remove redundant features. However, since the features are ranked individually and the proposed redundancy measures can only detect redundancy between two features, they may fail to identify multiple feature interactions [6], [8]. Therefore, a more powerful search technique is necessary to obtain better performance.

Particle Swarm Optimisation (PSO) [9] is a global search technique that simulates the social behaviors of birds flocking. In PSO, a swarm consists of many particles moving in the solution search space. Each solution is encoded as the position of each particle and evaluated by a fitness function. During the search process, each particle remembers the best solution it has found so far ($pbest$) and shares this information with its neighbours. In this way, each particle knows the best solution that the whole population has explored so far ($gbest$). By moving towards these good locations with its own velocity, particles are able to explore better solutions.

PSO has been applied and shown its high potential to FS [10], [11], [12]. However, due to the large search space, PSO-based methods on high-dimensional data still face the problem of stagnation in local optima. To overcome this problem, different strategies have been proposed such as changing $gbest$ when its fitness does not improve [13], [14] or combining filter and wrapper in two stages to reduce the number of features before using PSO for feature subset selection [15],

[16]. Hybrid approaches are also proposed as a flexible way to combine filter and wrapper in a single stage. For example, in a wrapper based PSO for FS [17], a local search using mutual information is applied on $gbest$ to remove redundant features. In contrast, the filter based PSO proposed in [18] uses classification accuracy to decide if a new $pbest$ should be updated or not. Results showed that the proposed method achieved higher classification accuracy than purely filter based PSO method. However, different combinations of filter and wrapper evaluations also showed that the more filter evaluations the algorithm used, the worse classification performance and the larger feature subsets it obtained. Therefore, how to combine filter and wrapper to synthesize their strengths while limiting their drawbacks is still challenging.

In a recent study [19], we proposed to use the $gbest$ resetting mechanism in [14] and a local search on every new $pbest$ found during the evolutionary process to find better solutions surrounding the new $pbest$ location. The local search was done by randomly flipping a small percentage of features from selected to not selected and vice versa. If a better flipped solution is found, then $pbest$ is updated. The proposed method achieved a significantly better classification accuracy with a smaller number of features than the previous method [14]. However, using a randomly flipping mechanism, the chance of reaching better solutions in this local search may be low. In order to improve its performance, the local search should be guided by some general or common sense knowledge in FS which can identify relevant and redundant features. Therefore, in this paper, we propose a new local search heuristic that uses a filter measure in choosing features for flipping in such a way that better solutions are more likely to be found.

A. Goals

The main goal of this paper is to develop a new local search PSO-based FS in classification on high-dimensional data. The proposed method will be examined and compared with other local search based PSO methods for FS. Specifically, we will investigate the following research objectives:

- Whether the feature subsets selected by the proposed algorithm are smaller and achieve similar or better classification performance than the original feature sets and those selected by the standard PSO.
- Whether the proposed local search heuristic significantly reduces the number of features and increases the classification performance over the randomly flipping local search.
- Whether the proposed method is more effective than the method proposed in [19] which combines both local search and resetting $gbest$.
- Whether the proposed method has significant shorter running time than the above compared methods.

II. Background and Related Work

A. Particle Swarm Optimisation

Particle swarm optimisation (PSO) [9] is a population based algorithm. It works by maintaining a swarm or population of particles. Each particle encodes a candidate solution in its position which is a vector of n real numbers where n is the dimensionality of the problem. Besides position, each particle also has a velocity which is n-dimension vector with each dimension showing the speed and direction that the particle should move in the next iteration. In each iteration, a fitness function is used to evaluate the particles. The best position that each particle has explored so far ($pbest$) is recorded and share among particles to find the best position of the whole population ($gbest$). Then, the velocity of a particle is updated based on its current velocity, and these two best positions. Specifically, the velocity and the position of each particle in each dimension are updated as follows:

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

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where v_{id}^{t} and x_{id}^{t} are the velocity and the position of particle i in dimension d at time t, respectively. p_{id} and p_{gd} are *pbest* and *gbest* positions in dimension d. c_1 and c_2 are acceleration constants, and r_1 and r_2 are random values uniformly distributed in [0, 1].

When applying PSO to FS, a candidate solution which is a feature subset is encoded in the position vector. The value in each dimension ranges from 0 to 1 indicating the probability of selecting the corresponding feature. Therefore, a threshold (e.g. 0.6) is used to convert this value into 1 or 0 indicating the feature is selected or not.

B. PSO for feature selection in classification on high-dimensional data

In the last decades, many PSO-based FS methods have been proposed and shown promising results in general [20], [21] as well as on high-dimensional data [22]. To tackle the problem of early convergence of PSO in the large search space, different approaches have been proposed. Yang et al. [13] replaced $gbest$ by applying a Boolean operator 'AND' on each bit of the $pbest$ of all particles. Similarly, Chuang et al. [14] proposed to reset $gbest$ to zero which is equivalent to an empty subset, which encourages particles to explore smaller feature subsets. Experiments on gene expression datasets showed that this method effectively reduced the number of selected features and got higher classification accuracy than [13] in most cases.

A two-stage approach which employs a filter method to reduce the number of features before applying PSO has become popular in FS on high-dimensional data. In [15], gain ratio was first used to select 500 top-ranked genes. Then, in PSO, speed concept was introduced to update particles' positions instead of velocity to increase the probability of not choosing a feature. In this way, the proposed algorithm was enabled to find much smaller feature subsets than [14] and other compared methods; however, with the deterioration in classification performance. Another strategy to reduce the number of features in the first stage is to build clusters of "similar" features and choosing only one or some features from each cluster to put into the second stage [23], [24]. The results showed that much smaller

feature subsets with higher accuracy were achieved. However, domain knowledge is required to set an appropriate number of clusters for each dataset [24]. Another two-stage method was proposed in [16] where F-statistic was used as a filter to select top ranked features before applying PSO. However, instead of predefining the number of features used in the second stage, cross-validation was used to choose this value from predefined ones. Results showed that the proposed method achieved good performance. However, the computational time was very high. In general, the results of these methods show that two-stage approach is feasible for high-dimensional data. However, the inherent drawback of individually evaluating features in the first stage may hinder PSO from achieving good better performance.

To overcome this limitation, hybrid approaches combining filter and wrapper method in a flexible way have been proposed. In a wrapper based PSO FS method [17], a filter local search was used to remove redundant features in $gbest$ using mutual information to evaluate feature relevance and redundancy. All features were first clustered into N clusters using a statistical clustering technique. In the local search, features in $gbest$ are removed if more than a predefine number of features from its cluster are selected. The results on UCI datasets showed that the proposed method achieve better solutions than previous PSO-based methods and traditional methods. However, it is not easy to define an appropriate number of features that should be chosen from each cluster especially for high-dimensional data where there may exist thousands of irrelevant or redundant features. In [18], mutual information was used to evaluate particles. However, a better solution is only updated to $pbest$ if its classification accuracy is also improved. The proposed methods had a shorter running time than wrapper based PSO; however, with lower classification performance and larger feature subsets.

III. The Proposed Method

A. Local search heuristic

In order to improve the performance of the local search proposed in [19], we used a filter measure to incorporate some general knowledge in FS to identify relevant and redundant features. A feature is relevant to the target concept if it demonstrates a significant degree of correlation to the class label. Similarly, features f_1 and f_2 are redundant features if they highly correlate with each other, which means f_1 can represents f_2 and vice versa; therefore, only one of them should be selected.

To evaluate the correlation of two variables, in this study, we use a normalised version of information gain (IG) [25] called symmetric uncertainty (SU) [26], which is defined as follows:

$$SU(X,Y) = \left[\frac{IG(X|Y)}{H(X)+H(Y)}\right] \quad (3)$$

$$IG(X|Y) = H(X) - H(X|Y) \quad (4)$$

where $H(X)$ is the entropy of X and $H(X|Y)$ is the conditional entropy of X given Y. The value of $SU(X|Y)$ is in the range [0,1]. The higher the value of SU, the higher the dependence between X and Y. Based on Eq. (3), we measure the relevance of a feature f using the SU between f and the class label. We denote this value as SU_C. Similarly, we use SU_F to denote the redundancy between two features f_1 and f_2 which is measured based on the SU between f_1 and f_2.

Algorithm 1: Local search heuristic

```
Input  : pbest, α, β
Output: better pbest
begin
    for t = 1 to β do
        m ← pbest_subset_size × α;
        ones ← Randomly pick m selected features in pbest;
        zeros ← Randomly pick m non-selected features in pbest;
        pbest' ← pbest ;
        Sort ones in descending order SU_C values (Eq. 3);
        for i = 1 to ones.length do
            for j = i + 1 to ones.length do
                if (ones[i] and ones[j] are still in pbest' and
                  SU_F(ones[i], ones[j]) > SU_C(ones[j]) ) then
                    Remove feature ones[j] from pbest';
                end
            end
        end
        AvgSU ← Average SU_C of all features in ones;
        for i = 1 to zeros.length do
            if SU_C(zeros[i]) > AvgSU then
                Add feature zeros[i] to pbest';
            end
        end
        if pbest' is better than pbest then
            pbest ← pbest' ;
        end
    end
    return pbest;
end
```

The objective of this local search is to find better $pbest$ by removing from the current $pbest$ a certain number of redundant features and introduce some more relevant features. The pseudo-code of the local search is presented in Algorithm 1. The inputs of this procedure are the current $pbest$, a percentage of features to flip (α) and a maximum number of flipping tries (β). While α is used to specify how difference the new $pbest$ compared to the current $pbest$, β specifies how long the local search should run. m is the number of features that will be considered to flip, which is proportional to the number of features selected in the current $pbest$. m selected features are chosen randomly from current $pbest$ to form the $ones$ list. $ones$ is sorted based on the SU_C of each feature and the class label. Then it is scanned from the first feature to remove any feature that is more correlated to the previous features than to the class label. In other words, we remove any feature that has its SU_F with a previous selected feature higher than its SU_C value.

In addition to removing redundant features, introducing more relevant features may improve the classification performance of the whole feature subset. Therefore, m random features are also chosen from the not selected features in $pbest$ to form the $zeros$ list. Features in this list will be added only if its SU_C is greater than the average SU_C values of the $ones$

list. After evaluated, if the new $pbest$ is better than the current $pbest$, the current $pbest$ will be updated.

B. Fitness function

As a wrapper method, any learning algorithm can be used to evaluate the classification performance of the candidate solution. In this study, we choose K-Nearest Neighbours (KNN) in our fitness function as it is simple, fast and non-parametric. Since many datasets used in the experiment are unbalanced, a balanced classification accuracy [27] as shown in Eq. 5 is used to guide the search.

$$balanced\_accuracy = \frac{1}{n}\sum_{i=1}^{n}\frac{TP_i}{|S_i|} \quad (5)$$

where n 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 total number of instances in class i. Since there is no bias to any specific class, the weight here is set equally to $1/n$.

In addition to accuracy, a distance measure is used to better evaluate feature subsets. Because in cases where the boundary margin between different classes is quite large, many different classifiers can achieve the same 100% classification accuracy. Therefore, using solely classification accuracy can not distinguish these candidate solutions. An additional measure is added with a weighting coefficient (μ) to maximize the distance between instances of different classes and minimize the distance between instances of the same class. The distance measure [28] is used in the fitness function as follows.

$$fitness = (\mu \cdot balanced\_accuracy + (1-\mu)\cdot distance) \quad (6)$$

where

$$distance = \frac{1}{1+exp^{-5(D_B - D_W)}} \quad (7)$$

$$D_b = \frac{1}{|S|}\sum_{i=1}^{|S|}\min_{\{j|j\neq i, class(V_i)\neq class(V_j)\}} Dis(V_i, V_j) \quad (8)$$

$$D_w = \frac{1}{|S|}\sum_{i=1}^{|S|}\max_{\{j|j\neq i, class(V_i)= class(V_j)\}} Dis(V_i, V_j) \quad (9)$$

$Dis(V_i, V_j)$ is any measure used to approximate the distance between two vectors V_i and V_j. Here we choose the number of matches or overlapping between two nominal vectors divided by the size of the vectors. Since KNN also works based on this overlapping distance calculation, adding this distance component to the fitness function does *not* increase the computation time to the evaluation process.

D_b is the average distance between each instance and the nearest instance of other classes. D_w is the average distance between each instance to the farthest instance of the same class. Although both are in the range [0,1], 1 is considered as the best case for D_b and the worst case for D_w. As a result, Eq. (7) maximises D_b and minimises D_w in order to find feature subsets that keep the same class instances close together and different class instances far away. The logistic function is used to transform the difference between D_b and D_w from the range [-1,1] into the range [0,1] for distance which 0 is the worst and 1 is the best case.

Fitness evaluation in Local Search: During the local search process, the same fitness as shown in Eq. (6) is used to evaluate the new $pbest$. This means that a significant amount of computation will be added to the algorithm. Therefore, to speed up the fitness evaluation process in this local search procedure, we use the same strategy proposed in [19] where distance between instances are calculated at the beginning of each local search. Then during the local search process, these distances are updated based only on a small number of flipping features.

C. The overall algorithm

Algorithm 2: The pseudo code of PSO-LSSU

```
input : Training set
output: The best feature subset
begin
    Discretise the training set using MDL method [?];
    Calculate SU_C and SU_F;
    Randomly initialise particles;
    while Maximum iterations or stopping criterion is not met do
        for i = 1 to Population Size do
            F_i ← Calculate the fitness of particle i using Eq. (6);
            if F_i is better than pbest's fitness then
                Update pbest ;
                if Local search then
                    pbest ← Local_Search(pbest, α, β);
                end
            end
        end
        Update gbest of the swarm;
        for i = 1 to Population Size do
            Update velocity of particle i using Eq. (1);
            Update position of particle i using Eq. (2);
        end
    end
    Return the position of gbest;
end
```

The aim of the proposed method is to apply local search on $pbest$ when a new $pbest$ is found in PSO so that smaller feature subsets with better classification performance can be achieved. The overall algorithm is shown in Algorithm 2. First, data is discretised using the MDL method [**?**]. Then, the symmetric uncertainty between each feature and the class label (the SU_C) and the symmetric uncertainty between pairs of features (the SU_F) are recorded for the local search procedure. The whole swarm will be randomly initialised. During the evolutionary process, local search is applied to the new found $pbest$. Since every local search will try 100 times to find better pbest, the computational time will significantly increase. On the other hand, running local search on every iteration may not increase the chance of finding better solutions. Therefore, we only apply local search in the first ten odd iterations.

IV. Experiment Design

This section shows the datasets used to test the performance of the proposed method and the parameter settings used for all methods in the experiment.

TABLE I
DATASETS

Dataset	#Features	#Ins.	#Class	%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
Prostate	10,509	102	2	49	51
11Tumor	12,533	174	11	4	16

A. Datasets

To test the performance of the proposed method, we use 8 gene expression datasets with thousands of features. These datasets are publicly available on http://www.gems-system.org. Details about these datasets are shown in Table I. It can be seen that these datasets have a small number of instances compared to their number of features. They are also imbalanced with the percentages of instances in the smallest class and the largest class are quite different. These characteristics make the FS problems in these datasets very challenging.

B. Experiment Configuration and Parameter Settings

To test the performance of the proposed method (called PSO-LSSU), we compared the classification accuracy of the selected features versus the original features, the feature subsets selected by the standard PSO, the PSO with random local search (PSO-LS) [19], and the PSO with random local search and resetting $gbest$ (PSO-LSRG) [19]. We also compare PSO-LSSU with two traditional FS methods, which are Linear forward selection (LFS) and greedy stepwise backward selection (GSBS). LFS and GSBS were derived from two typical FS approaches, which are sequential forward selection (SFS) and sequential backward selection (SBS), respectively. By restricting the number of features to be considered in each step, LFS [29] runs faster and found smaller feature subsets with better classification performance than SFS. On the other hand, GSBS (implemented based on the greedy stepwise FS in Weka [30]) starts with the full feature set and gradually remove features until no further improvement in classification accuracy.

In PSO-LS and PSO-LSRG, the fitness function is only based on the classification accuracy. The parameter settings for these two methods are kept the same as in [19]. However, the results reported in this paper will be different from those in [19] because in [19] the whole dataset was used in the FS process and training accuracy was reported, which included a selection bias [31]. Therefore, this experiment uses 10 fold cross-validation (10F-CV) in which one fold is kept as test data and the rest will be used to train PSO for FS. The training and test sets will be transformed based on the feature subset and put into the learning algorithm to evaluate its performance. Another difference is that in this experiment, all the datasets are discretised using MDL method [**?**] while [19] used the original continuous features.

TABLE II
PARAMETER SETTINGS

Parameters	Settings
Population Size	#features/20 (restriction to 300)
Maximum iterations	70
$c1 = c2$	2
w	$0.9 - 0.5 * \frac{current\ iteration}{max\ iteration}$
Threshold for selected feature	0.6
Communication topology	Fully connected
Stopping criterion	*PSO-LSSU:* same $gbest$ for 10 iterations
Local search tries	100
Local search flipping size	– *PSO-LS and PSO-LSRG:* 2% number of original features – *PSO-LSSU:* 25% of current $pbest$'s size
$gbest$ resetting	*PSO-LSRG:* same $gbest$'s fitness for 3 iterations

Since PSO is a stochastic optimisation technique, 30 independent runs with different seeds are executed for each training set. As a result, PSO is run 300 times (30 runs x 10 folds) for each dataset. Table II shows the parameter settings used in the experiment. Since these datasets have quite different numbers of features ranging from 2,000 to 12,000, which means that their search space are also very different, we set the population size equals to one twentieth of the number of features, but limited to 300 due to memory limitation.

V. Results and Discussions

Table III shows the results of the PSO based algorithms. "Full" means KNN using the original full set of continuous features. "#F" shows the average number of features selected by each method over the 30 runs. The best, the average and the standard deviation of the training and the test accuracies are shown in the "Training accuracy" and "Test accuracy" columns. The reported accuracy is the balanced accuracy calculated using Eq. (5). The best test classification accuracy on each dataset is bold. T_1 and T_2 display the Wilcoxon significant test results (with significant level of 0.05) of the corresponding method over the proposed method on training and test, respectively. "+" or "–" means the result is significantly better or worse than the proposed method and "=" means they are similar in the Wilcoxon tests. In other words, the more "–", the better the proposed methods.

A. PSO-LSSU versus Full

It can be seen from Table III that PSO-LSSU always achieve significantly better classification performance than using all features. The highest improvement is more than 16% on 11Tumor. In 6 out of 8 datasets, the feature subsets selected by PSO-LSSU are reduced one to two orders of magnitude with the best ratio of 1/209 in Leuk2. On SRBCT, the proposed method selects about 60 features among 2,308 features to achieve 100% accuracy in almost all 300 runs.

B. PSO-LSSU versus PSO

According to Table III, PSO-LSSU generates feature subsets giving significantly higher accuracy than PSO on 6 of the 8 datasets and selects at least an order of magnitude fewer

TABLE III
AVERAGE RESULTS OF 30 INDEPENDENT RUNS.

Dataset	Method	#F	Training accuracy		T_1	Test accuracy		T_2
			Best	Avg±Std		Best	Avg±Std	
SRBCT	Full	2308.0		83.35	–		87.08	–
	PSO	916.2	100.0	100.0 ± 0.02	=	98.75	96.61 ± 1.49	–
	PSO-LS	545.1	100.0	100.0 ± 0.00	=	99.17	96.08 ± 1.73	–
	PSO-LSRG	27.0	100.0	100.0 ± 0.00	=	92.50	86.39 ± 4.21	–
	PSO-LSSU	59.7	100.0	100.0 ± 0.00		100.0	**99.97 ± 0.15**	
DLBCL	Full	5469.0		81.71	–		83.00	–
	PSO	2286.5	100.0	99.86 ± 0.09	–	96.67	93.70 ± 2.01	+
	PSO-LS	1417.4	100.0	100.0 ± 0.00	=	97.33	**93.72 ± 1.79**	+
	PSO-LSRG	5.9	100.0	100.0 ± 0.00	=	85.83	76.90 ± 5.33	–
	PSO-LSSU	47.4	100.0	100.0 ± 0.00		96.67	90.86 ± 3.19	
9Tumor	Full	5726.0		33.44	–		36.67	–
	PSO	2551.6	96.80	95.67 ± 0.65	–	60.00	**53.28 ± 3.43**	=
	PSO-LS	1352.0	97.78	97.76 ± 0.05	=	58.33	48.39 ± 4.88	–
	PSO-LSRG	1122.5	97.78	97.77 ± 0.03	=	56.67	47.50 ± 4.50	–
	PSO-LSSU	46.7	97.78	97.78 ± 0.00		60.00	51.39 ± 4.22	
Leuk1	Full	5327.0		79.77	–		79.72	–
	PSO	2141.1	100.0	99.89 ± 0.13	–	95.56	93.93 ± 1.33	–
	PSO-LS	1534.9	100.0	100.0 ± 0.00	=	95.56	93.45 ± 1.71	–
	PSO-LSRG	16.5	100.0	100.0 ± 0.00	=	93.19	77.39 ± 6.01	–
	PSO-LSSU	31.9	100.0	100.0 ± 0.00		95.42	**94.84 ± 1.16**	
Brain1	Full	5920.0		65.07	–		72.08	–
	PSO	2478.1	100.0	99.26 ± 0.45	–	77.50	75.28 ± 1.48	–
	PSO-LS	1549.0	100.0	99.75 ± 0.22	–	77.50	75.00 ± 1.80	–
	PSO-LSRG	85.0	100.0	100.0 ± 0.00	=	73.33	63.81 ± 5.01	–
	PSO-LSSU	1081.5	100.0	99.96 ± 0.12		82.50	**76.78 ± 2.09**	
Leuk2	Full	11225.0		88.82	–		89.44	–
	PSO	4579.5	100.0	99.98 ± 0.05	=	95.00	92.05 ± 1.42	–
	PSO-LS	3426.5	100.0	100.0 ± 0.00	=	93.89	91.72 ± 1.46	–
	PSO-LSRG	16.5	100.0	100.0 ± 0.00	=	91.67	86.19 ± 2.94	–
	PSO-LSSU	53.7	100.0	100.0 ± 0.00		98.33	**95.56 ± 1.68**	
Prostate	Full	10509.0		82.08	–		85.33	–
	PSO	4590.3	97.82	97.39 ± 0.20	–	87.17	84.91 ± 1.30	–
	PSO-LS	2690.3	98.48	98.18 ± 0.12	–	89.17	85.79 ± 1.49	–
	PSO-LSRG	237.0	100.0	99.93 ± 0.07	+	89.33	84.33 ± 2.80	–
	PSO-LSSU	2670.3	98.92	98.64 ± 0.16		91.17	**86.98 ± 1.76**	
11Tumor	Full	12533.0		71.01	–		71.42	–
	PSO	5585.7	99.35	99.01 ± 0.20	–	87.63	84.58 ± 1.40	–
	PSO-LS	3163.9	99.92	99.83 ± 0.09	–	87.77	84.19 ± 1.47	–
	PSO-LSRG	1139.9	100.0	99.94 ± 0.05	–	88.58	82.50 ± 2.19	–
	PSO-LSSU	266.8	100.0	100.0 ± 0.00		90.72	**87.51 ± 1.73**	

features than PSO on most datasets. The highest reduction can be seen in Leuk2, PSO-LSSU selects 85 times fewer features than PSO and still improves on the PSO performance more than 3%. On 9Tumor, PSO-LSSU obtains a similar classification performance to PSO with only 46 features which is 2,505 fewer features than PSO. Only on DLBCL does PSO achieve a higher accuracy but it selects a much larger number of features. We also note that the accuracy of PSO-LSSU feature set on the training data is always higher than that obtained by standard PSO except on SRBCT where both obtain 100%. Therefore, the proposed local search heuristic of PSO-LSSSU has achieved the goal of removing redundant features to obtain smaller feature subsets while maintaining or improving the classification performance.

C. PSO-LSSU versus PSO-LS

Compared to the random local search (PSO-LS), the proposed local search heuristic (PSO-LSSU) achieves significantly better classification accuracy on 7 out of 8 datasets. Although PSO-LS further reduces the number of features selected by PSO, it still selects an order of magnitude more features than PSO-LSSU. On 9Tumor and Leuk2, while PSO-LS selects 1,352 and 3,426 features, PSO-LSSU selects only 46 and 53 features to further improve 3% and 4% accuracy, respectively. The results show that by using a filter measure to guide the flipping process, the local search effectively removes redundant features so that much smaller feature subsets with better discriminating power can be found.

D. PSO-LSSU versus PSO-LSRG

The significant test results ($T2$) in Table III show that in terms of classification performance, PSO-LSSU outperforms PSO-LSRG on all datasets. Even though PSO-LSSU selects more features than PSO-LSRG on six datasets, the features selected by PSO-LSSU are important in improving or maintaining the discriminating ability of the feature subset. For example on Leuk1 and DLBCL, PSO-LSSU selects 15 and 42 more features than PSO-LSRG to achieve 23% and 24% higher accuracy, respectively. However, on 9Tumor, PSO-LSSU selects 24 times fewer features than PSO-LSRG while obtaining 4% higher accuracy than PSO-LSRG. A similar pattern can be seen on 11Tumor.

We also observe that PSO-LSRG obtains the smallest feature subsets on 6 datasets. Although its training results are similar to or better than those of PSO and PSO-LS, which is consistent with the results reported in [19], its test results is always the worst among the compared methods. This observation shows that resetting $gbest$ to an empty subset effectively guides PSO to smaller solution space; however, without guarantee to maintain the same performance on the test set, especially when there are a large number of redundant features and the number of instances for training is limited as in these datasets. As a result, using general knowledge in FS to guide the search for smaller feature subsets is better than solely resetting $gbest$ to an empty set.

E. Computation Time

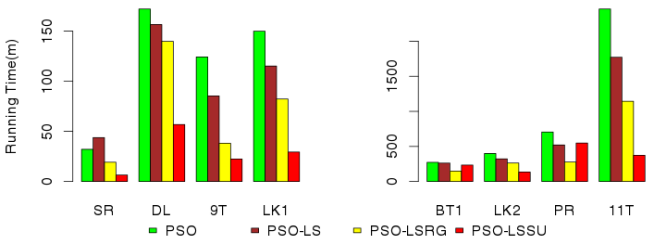

Fig. 1. Average running time.

Figure 1 shows the average running time (in minutes) of PSO, PSO-LS, PSO-LSRG and PSO-LSSU in a single run on each dataset. It can be seen that PSO-LSSU used the shortest time on 6 out of 8 datasets. Although PSO-LSSU has to spend additional time on the local search, it always runs faster than the standard PSO on all datasets. On SRBCT, 9Tumor, Leuk1, and 11Tumor, its running time is five to six times shorter than PSO. By applying local search on $pbest$ to remove redundant features, PSO-LSSU significantly reduces the number of features selected in each candidate solution, which dramatically speeds up the fitness evaluation process.

Although the proposed local search has extra steps for considering the flipping features based on their symmetric uncertainty values, PSO-LSSU has a much shorter running time than PSO-LS on six datasets and similar on the other two. Thanks to the small feature subsets and the strategy of running local search only in the first ten odd iterations, PSO-LSSU has significantly reduced its running time while still achieve much better discriminating subsets than PSO-LS. The results suggest that applying a small number of informed local search is better than applying an extensive number of random local search. Similarly, PSO-LSSU also has a significant shorter running time than PSO-LSRG on six datasets. In general, the results indicate that the proposed local search heuristic is not only an effective but also efficient combination of wrapper and filter approach to achieve a much smaller feature subsets with significantly better classification performance.

F. Comparisons with Traditional Methods

TABLE IV
RESULTS OF LFS

SRBCT			DLBCL			9Tumor			Leuk1		
#F	Accuracy	T	#F	Accuracy	T	#F	Accuracy	T	#F	Accuracy	T
6.1	88.75	–	4.0	74.00	–	12.6	41.67	–	4.8	81.39	–
Brain1			Leuk2			Prostate			11Tumor		
#F	Accuracy	T	#F	Accuracy	T	#F	Accuracy	T	#F	Accuracy	T
9.9	59.17	–	4.3	90.00	–	4.9	73.17	–	14.3	61.71	–

Table IV shows the average number of features and the average accuracy obtained by LFS on 10 fold cross validation of the 8 datasets. Column T shows the significance test result of LFS over PSO-LSSU. The results of GSBS are not displayed because GSBS cannot finish its run in a week for any of the 8 datasets.

Comparing the result of PSO-LSSU in Table III with Table IV, we can see that LFS selected a much smaller number of features than PSO-LSSU. However, in terms of the classification performance, PSO-LSSU outperformed LFS on all datasets with the biggest difference of 25% on 11Tumor. This indicates that PSO-LSSU can better explore the solution space to obtain better solutions than LFS.

G. Further Analysis

In this experiment, each algorithm has run 300 independent times producing 300 different solutions for each dataset. To see if the features selected by the proposed method in all runs are relevant and not selected by chance, we compare the Z-score [32], [8] of the top 100 features selected by each method. Z-score of a feature indicates the significance of the selection frequency of that feature. Z-score of feature i is defined as follows:

$$Z_i = \frac{f_i - \mu}{\sigma} \tag{10}$$

where f_i is the number of times feature i appeared in M solutions, μ and σ are the mean and standard deviation of f_i. Let A be the average feature subset size of M solutions, T is the total number of features, then the probability of feature i being selected is denoted as, $P(f_i) = A/T$. Using this probability, the mean and the standard deviation of f_i are calculated using $\mu = P(f_i) \cdot M$ and $\sigma = \sqrt{P(f_i) \cdot (1 - P(f_i)) \cdot M}$.

It can be seen from Eq. (10) that the higher the value of Z_i, the less likely that feature i is selected by chance. Therefore, an algorithm that selects more features with higher Z-score is said to be more robust. To compare the robustness of the

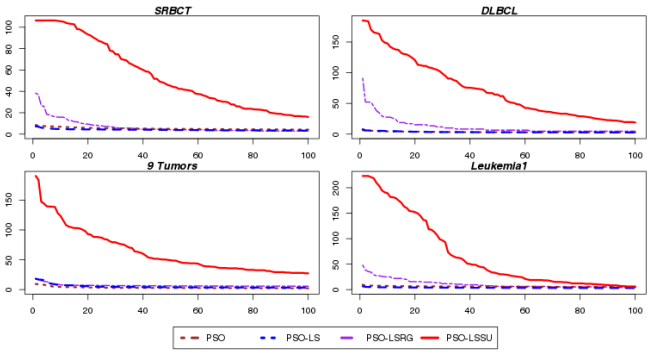

Fig. 2. Z-score of the top 100 selected features on SRBCT, DLBCL, 9Tumor and Leuk1.

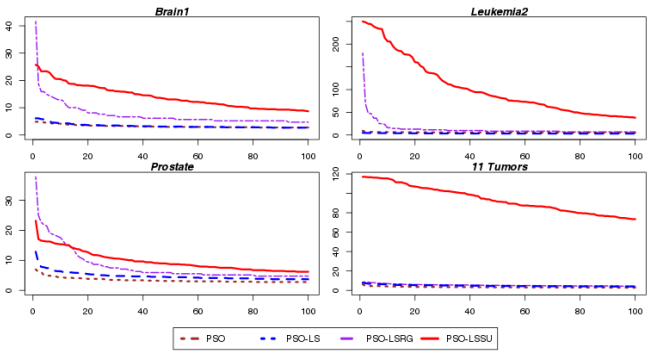

Fig. 3. Z-score of the top 100 selected features on Brain1, Leuk2, Prostate and 11Tumor.

four methods, we sort all features in descending order based on their selection frequency in each method. Z-score of the top 100 selected features by each method on each dataset are plotted in Fig. 2 and Fig. 3. It can be seen from these figures that the features selected by PSO-LSSU have much larger Z-score than other methods. Further more, these Z-scores form a gentle slope in almost all cases which indicates that more relevant features are selected by the proposed method. On Brain1 and Prostate, although the top-ranked feature of PSO-LSRG has higher Z-score than PSO-LSSU, the following features have their Z-score dramatically dropped in PSO-LSRG while PSO-LSSU still maintain these values with a flatter slope. The results show that PSO-LSSU is more robust than the other methods.

VI. CONCLUSIONS

This paper has developed a new PSO-based FS method (PSO-LSSU) that dynamically employed local search during the evolutionary process of PSO. While PSO uses classification accuracy to guide the search, the local search is guided by

a filter measure to remove redundant features and add more relevant features to the current $pbest$ solution.

The experimental results on 8 high-dimensional datasets with varying difficulty have shown that the proposed method achieve much smaller feature subsets with significantly better classification performance than the original feature sets and the standard PSO. Compared to the random local search [19], the use of symmetric uncertainty to choose features for flipping helps local search can effectively discover relevant and redundant features. As a result, PSO-LSSU obtained much smaller feature subsets with higher discriminating ability than the PSO-LS. Compared to PSO-LSRG, although PSO-LSSU selects slightly more features on six datasets, it significantly outperformed PSO-LSRG on all datasets in terms of classification accuracy. PSO-LSSU's performance is confirmed by the analysis on the Z-score of the top 100 selected features. The results show that PSO-LSSU selected more relevant features than the other compared methods.

The results and analysis have suggested that local search using general knowledge in FS can significantly improve the performance of PSO in FS on high-dimensional data. In this study, we have applied local search on $pbest$, it would be interesting to investigate where we should apply local search, whether on particles, $pbest$, or $gbest$ or any combination of these candidate solutions will produce the best results. Other investigations can be on when and how long the local search should be run to have the best compromise between effectiveness and efficiency as well as to balance the exploration and exploitation in PSO search.

REFERENCES

[1] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[2] M. Robnik-Sikonja and I. Kononenko, "Theoretical and Empirical Analysis of ReliefF and RReliefF," *Machine Learning*, vol. 53, pp. 23–69, 2003.

[3] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992, pp. 129–134.

[4] A. J. Ferreira and M. A. Figueiredo, "Efficient feature selection filters for high-dimensional data," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1794–1804, 2012.

[5] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005.

[6] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *J. Mach. Learn. Res.*, vol. 5, pp. 1205–1224, 2004.

[7] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 359–366.

[8] Z. Zhu, Y.-S. Ong, and M. Dash, "Markov blanket-embedded genetic algorithm for gene selection," *Pattern Recognition*, vol. 40, no. 11, pp. 3236–3248, 2007.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

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

[11] B. Chakraborty and G. Chakraborty, "Fuzzy consistency measure with particle swarm optimization for feature selection," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 4311–4315.

[12] 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 Science*, vol. 24, no. 2-3, pp. 91–116, 2012.

[13] C. S. Yang, L. Y. Chuang, C. H. Ke, and C. H. Yang, "Boolean binary particle swarm optimization for feature selection," in *IEEE Congress on Evolutionary Computation*, 2008, pp. 2093–2098.

[14] L.-Y. Chuang, H.-W. Chang, C.-J. Tu, and C.-H. Yang, "Improved binary PSO for feature selection using gene expression data," *Computational Biology and Chemistry*, vol. 32, no. 29, pp. 29–38, 2008.

[15] M. 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," *Information Technology in Biomedicine*, vol. 15, no. 6, pp. 813–822, 2011.

[16] W. Zhou and J. A. Dickerson, "A novel class dependent feature selection method for cancer biomarker discovery," *Computers in biology and medicine*, vol. 47, pp. 66–75, 2014.

[17] H. B. Nguyen, B. Xue, I. Liu, and M. Zhang, "Filter based backward elimination in wrapper based pso for feature selection in classification," in *IEEE Congress on Evolutionary Computation*, 2014, pp. 3111–3118.

[18] T. Butler-Yeoman, B. Xue, and M. Zhang, "Particle swarm optimisation for feature selection: A hybrid filter-wrapper approach," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 2428–2435.

[19] B. Tran, B. Xue, and M. Zhang, "Improved PSO for Feature Selection on High-Dimensional Datasets," in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science, 2014, vol. 8886, pp. 503–515.

[20] A. Unler and A. Murat, "A discrete particle swarm optimization method for feature selection in binary classification problems," *European Journal of Operational Research*, vol. 206, no. 3, pp. 528–539, 2010.

[21] B. Xue, M. Zhang, and W. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[22] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[23] M. Lane, B. Xue, I. Liu, and M. Zhang, "Gaussian based particle swarm optimisation and statistical clustering for feature selection," in *Evolutionary Computation in Combinatorial Optimisation*, ser. Lecture Notes in Computer Science, 2014, vol. 8600, pp. 133–144.

[24] B. Sahu and D. Mishra, "A Novel Feature Selection Algorithm using Particle Swarm Optimization for Cancer Microarray Data," *Procedia Engineering*, vol. 38, no. 0, pp. 27–31, 2012.

[25] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, Inc., 1993.

[26] W. H. Press, S. Teukolsky, W. Vetterling, and B. Flannery, "Numerical recipes in c," *Cambridge University Press*, vol. 1, p. 3, 1988.

[27] G. Patterson and M. Zhang, "Fitness functions in genetic programming for classification with unbalanced data," in *Advances in Artificial Intelligence*. Springer, 2007, pp. 769–775.

[28] H. Al-Sahaf, M. Zhang, M. Johnston, and B. Verma, "Image descriptor: A genetic programming approach to multiclass texture classification," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 2460–2467.

[29] M. Gutlein, E. Frank, M. Hall, and A. Karwath, "Large-scale attribute selection using wrappers," in *IEEE Symposium on Computational Intelligence and Data Mining*, 2009, pp. 332–339.

[30] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[31] C. Ambroise and G. J. McLachlan, "Selection bias in gene extraction on the basis of microarray gene-expression data," *Proceedings of the National Academy of Sciences*, vol. 99, no. 10, pp. 6562–6566, 2002.

[32] T. Jirapech-Umpai and S. Aitken, "Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes," *BMC bioinformatics*, vol. 6, no. 1, p. 148, 2005.