A New Binary Particle Swarm Optimisation Algorithm for Feature Selection

Bing Xue, Su Nguyen, and Mengjie Zhang

School of Engineering and Computer Science,
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
{Bing.Xue,Su.Nguyen,Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract. Feature selection aims to select a small number of features from a large feature set to achieve similar or better classification performance than using all features. This paper develops a new binary particle swarm optimisation (PSO) algorithm (named PBPSO) based on which a new feature selection approach (PBPSOfs) is developed to reduce the number of features and increase the classification accuracy. The performance of PBPSOfs is compared with a standard binary PSO based feature selection algorithm (BPSOfs) and two traditional feature selection algorithms on 14 benchmark problems of varying difficulty. The results show that PBPSOfs can be successfully used for feature selection to select a small number of features and improve the classification performance over using all features. PBPSOfs further reduces the number of features selected by BPSOfs and simultaneously increases the classification accuracy, especially on datasets with a large number of features. Meanwhile, PBPSOfs achieves better performance than the two traditional feature selection algorithms. In addition, the results also show that PBPSO as a general binary optimisation technique can achieve better performance than standard binary PSO and uses less computational time.

Keywords: Binary particle swarm optimisation, Feature selection, Classification

1 Introduction

Feature selection is an important task in classification, which aims to select a subset of features and achieve similar or even better classification performance than using all features [1]. By removing irrelevant or redundant features and selecting only relevant features for classification, feature selection can reduce the dimensionality, simplify the learned classifiers, and/or increases the classification accuracy [1]. Feature selection is a difficult combinatorial problem with a large search space. The size of the search space grows exponentially along with the total number of features in the dataset. Therefore, it is usually impractical to perform an exhaustive search and most of the existing methods suffer from the problem of being computationally expensive or becoming stuck in local optima. Therefore, feature selection tasks need an efficient global search method.

Evolutionary computation (EC) techniques are a group of powerful global search algorithms. Particle swarm optimisation (PSO) [2, 3] is a relatively recent EC technique, which is easy to implement, computationally less expensive, and has fewer parameters than other EC algorithms, such as genetic programming (GP) and genetic algorithms (GAs) [4]. Therefore, PSO has gained much attention since it was first proposed [2]. In PSO, each candidate solution is encoded as an individual or a particle in the search space. Each particle i has a position shown by $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and a velocity shown by $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$, where D is the dimensionality of the search space. During the evolutionary process, the best previous position of a particle is recorded as the personal best *pbest* and the best position obtained by the population thus far is called *gbest*. PSO searches for the optimal solutions by updating the velocity and the position of each particle according to *pbest* and *gbest*. There are two main categories of PSO, which are continuous PSO [3] and binary PSO (BPSO) [5]. Both continuous PSO and BPSO have been used for feature selection [6–8].

Feature selection is a binary problem, where BPSO is a more appropriate method than continuous PSO [7]. In BPSO, all elements in the position are either 1 or 0. The velocity in BPSO indicates the probability of the corresponding element in the position vector taking value 1. A sigmoid function is introduced to transform v_{id} to the range of (0, 1). BPSO updates the position and velocity of each particle according to the following formulae:

$$x_{id}^{t+1} = \begin{cases} 1, \text{if } rand() < \frac{1}{1+e^{-v_{id}^{t+1}}} \\ 0, otherwise \end{cases} \tag{1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (y_{id} - x_{id}^{t}) + c_2 * r_{2i} * (\hat{y}_d - x_{id}^{t}) \tag{2}$$

where t denotes the t^{th} iteration. $d \in D$ denotes the d^{th} dimension in the search space. $rand()$ is a random number uniformly distributed in [0,1]. w is inertia weight. c_1 and c_2 are acceleration constants. r_{1i} and r_{2i} are random values uniformly distributed in [0, 1]. y_{id} and $\hat{y}_d$ represent the elements of *pbest* and *gbest* in the d^{th} dimension.

There have been a large number of works on PSO [4], but most of them focus on continuous PSO and there is not much work on BPSO [9] perhaps because there are some limitations on the current standard BPSO. For example, the velocity shows the momentum of a particle's movement in a direction in a particular dimension of the search space. However, this was originally designed for continuous spaces. For binary problems, x_{id} can only be 1 or 0, which means particles cannot keep moving in one direction of a particular dimension. Meanwhile, the parameters in velocity were also designed for continuous PSO. When applying the velocity to BPSO, the parameters cannot produce the effects they were designed for and in fact, they produce an opposite effect compared with in the original continuous PSO (detailed discussions can be seen in [10]). Therefore, the velocity in PSO for continuous space is not meaningful any more in binary space. Since the influence of personal best and global best is reflected through the velocity updating equation, the personal experience and global experience cannot effectively utilised in BPSO. Therefore, in order to address feature selection problems, a new BPSO algorithm is needed.

1.1 Goals

The overall goal of this paper is to develop a new BPSO algorithm for feature selection to select a small feature subset and achieve better classification performance than using all features. To achieve this goal, we propose a new updating mechanism to develop a new BPSO algorithm based on which a new feature selection approach is proposed to reduce the number of features and increases the classification accuracy. Specifically, we will investigate:

– whether the new feature selection approach can be used to address feature selection problems to reduce the number of features and increase the classification accuracy,
– whether the new feature selection approach can achieve better performance than two traditional feature selection algorithms, and
– whether the new BPSO algorithm as a general binary optimisation technique can achieve better performance than the standard BPSO in a shorter computational time.

2 Proposed Approach

To overcome the limitations of standard BPSO [5], we develop a new binary PSO algorithm, where two important issues are considered. The first is to follow the key ideas of the standard (continuous) PSO algorithm, which is that particles are updated according to the best experience of its own (i.e. personal best, *pbest*) and the best experience of its neighbours (i.e. global best, *gbest*). The second is to keep advantages of PSO compared to other EC techniques, i.e. PSO is simple, has fewer parameters and computationally cheaper. Therefore, we aim to develop a new BPSO algorithm, which is simpler than standard BPSO, but has a more powerful search ability.

Since the velocity component in BPSO is not as meaningful as in continuous PSO, we propose a new probability based BPSO named PBPSO, where a "flipping" probability is introduced to replace the velocity to update each particle during the evolutionary process. p shows the "flipping" probability, which is a $D-$dimensional vector. $p_i = (p_{i1}, p_{i2}, ..., p_{iD})$ shows the "flipping" probability for particle i. p_{id} shows the probability of x_{id} being "flipped", i.e. update $x_{id}^{t+1} = 1$ if $x_{id}^t = 0$ or update $x_{id}^{t+1} = 0$ if $x_{id}^t = 1$, where t means the t^{th} iterations in the evolutionary process. The new position updating equation is shown by Equation 3. p is calculated based on the current position of a particle, *pbest* and *gbest*, where the updating formula is shown by Equation 4.

$$x_{id}^{t+1} = \begin{cases} 1 - x_{id}^t, & \text{if } random() < p_{id} \\ x_{id}^t, & otherwise \end{cases} \tag{3}$$

$$p_{id} = p_0 + p_{pd} + p_{gd} \tag{4}$$

where

$$p_{pd} = \begin{cases} p_1, \text{ if } x_{id}^t \neq y_{id}^t \\ 0, \ otherwise \end{cases}$$

and

$$p_{gd} = \begin{cases} p_2, \text{ if } x_{id}^t \neq \hat{y}_d^t \\ 0, \ otherwise \end{cases}$$

In Equation 3, $(1 - x_{id}^t)$ is used to update x_{id}^{t+1} from 1 to 0 or from 0 to 1. p_{pd} and p_{gd} reflect the influence of personal best *pbest* and global best *gbest*. p_0, p_1 and p_2 are real numbers in (0,1). $0 < p_0$ is used to ensure that there is always a probability to change the value of x_{id}^t. The values of p_{pd} and p_{gd} are calculated for each dimension in every iteration, but they are not stored through the evolutionary process, which is cheaper than the standard BPSO in terms of memory. $p_0 + p_1 + p_2 = 1$ ensures that when x_{id}^t is different from both y_{id}^t and $\hat{y}_{id}^t$, the probability for x_{id}^t to change equals to 1.

2.1 PBPSOfs for Feature Selection

Based on the proposed PBPSO, a new feature selection approach named PBPSOfs is developed and the Pseudo-code of PBPSOfs is shown by Algorithm 1. Two key components that need to be shown are the encoding scheme and the fitness function.

In PBPSOfs, the dimensionality (D) of the search space equals to the total number of features in the dataset. So each particle is a D-dimensional binary string, where "1" means the corresponding feature is selected and "0" means the corresponding feature is not selected.

Feature selection has two main objectives, which are maximising the classification accuracy (minimising the error rate) and minimising the number of features. Therefore, a fitness function that combines the two objectives is used in PBPSOfs, which is shown by Equation 5.

$$Fitness = \alpha * ErrorRate + (1 - \alpha) * \frac{\#Features}{\#All\ Features} \tag{5}$$

where *ErrorRate* represents the classification error rate of the selected features. $\#Features$ shows the number of selected features and $\#All\ Features$ shows the total number of features in the datasets. α and $(1 - \alpha)$ reflect the relative importance of the classification performance and the number of selected features. $\alpha \in (0.5, 1]$ because the classification performance is regarded as more important than the number of features.

3 Experiment Design

3.1 Benchmark Techniques

To examine the performance of the proposed algorithm (PBPSOfs), it is compared with a standard BPSO based feature selection algorithm (BPSOfs) [11]. BPSOfs shares the same encoding scheme, fitness function and random seeds with PBPSOfs for fair comparisons.

Algorithm 1: Pseudo-code of PBPSOfs

```
Input  : $p_0$, $p_1$, $p_2$; $P$: the population size; $T$: maximum iterations;
         $D$: dimensionality of search space (i.e. total number of features)
Output: $gbest$ (i.e. selected features), training and testing accuracies of the
         selected features
begin
    randomly initialise the position each particle;
    for $t$=1 to $T$ do
        evaluate fitness of each particle;
        for $i$=1 to $P$ do
            update the $pbest$ of particle $i$;
            update the $gbest$ of particle $i$;
        for $i$=1 to $P$ do
            for $d$=1 to $D$ do
                if $x_{id}^{t-1} \neq y_{id}^{t-1}$ then
                    $p_{pd} = p_1$ ;              // personal experience, $pbest$
                else
                    $p_{pd} = 0$
                if $x_{id}^{t-1} \neq \hat{y}_{d}^{t-1}$ then
                    $p_{gd} = p_2$ ;      // neighbourhood's experience, $gbest$
                else
                    $p_{gd} = 0$
                $p_{id} = p_0 + p_{pd} + p_{gd}$;
                if $rand < prob$ then
                    $x_{id} = 1 - x_{id}$ ;     // $x_{id}$ from 1 to 0 or from 0 to 1
                else
                    $x_{id} = x_{id}$
    return $gbest$, training and testing accuracies of the selected features;
```

Two traditional methods are also used to test the performance of PBPSOfs, which are linear forward selection (LFS) [12] and greedy stepwise backward selection (GSBS) [13]. LFS and GSBS were driven from two typical traditional feature selection algorithms, i.e. sequential forward selection (SFS) [14] and sequential backward selection (SBS) [15]. LFS [12] performs a forward selection, but restricts the number of features that are considered in each step. LFS is computationally less expensive than SFS because it reduces the number of evaluations. More details can be seen in [12]. The greedy stepwise based selection method is implemented in Weka [16], which can perform both forward and backward selection [13]. Given that LFS is based on forward selection, the greedy stepwise search is set to be backward to conduct a greedy stepwise backward selection (GSBS). GSBS starts with all available features and stops when the deletion of any remaining feature results in a decrease in classification accuracy.

3.2 Datasets and Parameter Settings

14 datasets were chosen from the UCI machine learning repository [17] to test the performance of PBPSOfs, BPSOfs, LFS and GSBS. The datasets are shown

Table 1. Datasets

Dataset	No. of Features	No. of Classes	No. of Instances	Dataset	No. of Features	No. of Classes	No. of Instances
Australian	14	2	690	Zoo	17	7	101
Wisconsin Breast Cancer (Diagnostic) (WBCD)	30	2	569	Vehicle	18	4	846
Ionosphere	34	2	351	German	24	2	1000
Hillvalley	100	2	606	Lung	56	3	32
Musk Version1(Musk1)	166	2	476	Sonar	60	2	208
Arrhythmia	279	16	452	Madelon	500	2	4400
Multiple Features	649	10	2000	Isolet5	617	2	1559

in Table 1. The 14 datasets were chosen to have different numbers of features, classes and instances. For each dataset, the instances are randomly divided into two sets: 70% as the training set and 30% as the test set.

As wrapper approaches, all the algorithms need a learning/classification algorithm. A simple and commonly used learning algorithm [7], K-nearest neighbour (KNN), was used in the experiment and K=5 (5NN). During the evolutionary training process, the classification error rate used in the fitness function is calculated using 10-fold cross-validation on the training set. Note that 10-fold cross-validation is performed as an inner loop in the training process to evaluate the classification performance of a single feature subset on the training set and it does not generate 10 feature subsets. After the training process, the selected features are evaluated on the test set to obtain the testing classification performance of the selected features. A detailed discussion of why and how 10-fold cross-validation is applied in this way is given by [18].

PBPSOfs only involves three parameters, which are p_0, p_1 and p_2. $p_0 = 0.05$ is to make sure that there is always at least a very small probability to update the particle. $p_1 = 0.35$ and $p_2 = 0.65$ are to ensure that the global best has slightly more influence than the personal best. The parameters of BPSOfs are set as follows [3]: $w = 0.7298$, $c_1 = c_2 = 1.49618$. For both BPSOfs and PBPSOfs, the population size is 50, and the maximum number of iterations is 100. The fully connected topology is used. $\alpha = 0.9$ is used in the fitness function to make sure that the classification performance is much more important than the number of features. Both PBPSOfs and BPSOfs have been conducted for 40 independent runs on each dataset.

To test their classification performance, the non-parametric statistical significance test, Wilcoxon test, is performed compare the classification performance of BPSOfs (or PBPSOfs) and that of all features. The significance test is used to compare the classification performance between BPSOfs and PBPSOfs. The significance level is selected as 0.05 (or confidence interval is 95%).

4 Results and Discussions

4.1 Results of PBPSOfs and BPSOfs in Testing Process

Table 2 shows the experimental results of PBPSOfs and BPSOfs on the unseen test sets, where "All" means that all of the available features are used for classification, "AveSize" shows the average number of features selected in the 40

Table 2. Experimental Results in Testing Process

Dataset	Method	AveSize	BestAcc	AveAcc ± StdAcc	Test 1	Test 2
Australian	All	14	70.05			
	BPSOfs	3.18	87.44	82.46 ± 7.3131	+	
	PBPSOfs	2.82	85.51	81.96 ± 7.7095	+	=
Zoo	All	17	80.95			
	BPSOfs	4.15	97.14	95.12 ± 0.6455	+	
	PBPSOfs	3.35	95.24	95.17 ± 0.2508	+	=
Vehicle	All	18	83.86			
	BPSOfs	5.38	86.22	84.21 ± 0.8119	+	
	PBPSOfs	5.02	84.84	84.09 ± 0.434	+	=
German	All	24	68.0			
	BPSOfs	7.9	73.33	68.89 ± 2.0768	+	
	PBPSOfs	5.95	72.67	69.09 ± 1.7842	+	=
WBCD	All	30	92.98			
	BPSOfs	7.92	94.74	93.57 ± 1.2474	+	
	PBPSOfs	2.02	94.74	94.4 ± 0.8049	+	+
Ionosphere	All	34	83.81			
	BPSOfs	8.92	93.33	88.14 ± 2.3128	+	
	PBPSOfs	4.58	91.43	88.45 ± 2.0447	+	=
Lung	All	56	70.0			
	BPSOfs	23.28	90	74.75 ± 7.0666	+	
	PBPSOfs	6.78	80	77.25 ± 5.9108	+	=
Sonar	All	60	76.19			
	BPSOfs	23.02	85.71	78.57 ± 3.4594	+	
	PBPSOfs	14.28	87.3	78.21 ± 3.0323	+	=
Hillvalley	All	100	56.59			
	BPSOfs	39.35	60.16	56.88 ± 1.6322	=	
	PBPSOfs	31.08	61.26	58.25 ± 1.6952	+	+
Musk1	All	166	83.92			
	BPSOfs	75.52	90.91	84.21 ± 2.8401	=	
	PBPSOfs	69.3	88.81	85.38 ± 1.8087	+	+
Arrhythmia	All	279	94.46			
	BPSOfs	99.7	95.14	94.21 ± 0.3937	-	
	PBPSOfs	63.42	95.48	94.71 ± 0.3405	+	+
Madelon	All	500	70.9			
	BPSOfs	243.85	78.59	75.81 ± 1.4905	+	
	PBPSOfs	212.42	81.15	78.91 ± 1.2565	+	+
Isolet5	All	617	98.45			
	BPSOfs	225.15	98.59	98.25 ± 0.1354	-	
	PBPSOfs	169.35	98.87	98.61 ± 0.1248	+	+
Multiple Features	All	649	98.63			
	BPSOfs	237.05	99.1	98.89 ± 0.0923	+	
	PBPSOfs	176.15	99.27	99.01 ± 0.1043	+	+

independent runs, "BestAcc", "AveAcc" and "StdAcc" show the best, the average and the standard deviation of the 40 testing accuracies. "Test 1" shows the results of the Wilcoxon significance tests between PBPSOfs (or BPSOfs) and "All", where "+" (-) means PBPSOfs or BPSOfs is significantly better (or worse) than "All", and "=" means they are similar (no significant difference). "Test 2" shows the Wilcoxon significance tests between PBPSOfs and BPSOfs.

Results of BPSOfs. According to Table 2, it can be seen that in most cases (i.e. 12 out of the 14 datasets), the classification performance of the feature subsets selected by BPSOfs is significantly better or similar to that of using all features. In all cases, the average number of features selected by BPSOfs is less than half of the total number of original features. However, on the six datasets with a large number of features (100 or more), the classification performance of BPSOfs is better than using all features on only two datasets.

The results show that BPSOfs with the standard BPSO algorithm can be used to address feature selection problems to reduce the number of features and

Table 3. Comparisons with LFS and GSBS

Dataset	Australian			Zoo			Vehicle			German			WBCD			Ionosphere			Lung		
	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T
LFS	4	70.05	+	8	79.05	+	9	83.07	+	3	68.67	=	10	88.89	+	4	86.67	+	6	90.0	-
GSBS	12	69.57	+	7	80.0	+	16	75.79	+	18	64.33	+	25	83.63	+	30	78.1	+	33	90.0	-
Dataset	Sonar			Hillvalley			Musk1			Arrhythmia			Madelon			Isolet5			MultipleF.		
	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T	Size	Acc	T
LFS	3	77.78	=	8	57.69	+	10	85.31	=	11	94.46	+	7	64.62	+	24	98.34	+	18	99.0	=
GSBS	48	68.25	+	90	49.45	+	122	76.22	+	130	93.55	+	489	51.28	+	560	97.16	+			+

maintain or even increase the classification performance. However, the results also show the BPSOfs cannot scale well for high-dimensional problems, where feature selection is important and necessary on such problems.

Results of PBPSOfs. From Table 2, it can be seen that the classification performance of the feature subsets selected by PBPSOfs is significantly better than using all features on **all** datasets. In most cases, the average number of features selected by PBPSOfs is less than one third of the total number of features. For example, on the WBCD dataset, PBPSOfs selected only 6.73% (i.e. on average 2.02 of 30) of the original features and achieved significantly better classification performance than using all features.

The results show that PBPSOfs with the new updating mechanism can successfully evolve a smaller feature subset to increase the classification accuracy.

Comparisons Between PBPSOfs and BPSOfs. From Table 2, it can be seen that on **all** the 14 datasets, PBPSOfs selected a smaller number of features and achieved similar or significantly better classification performance than BPSOfs. The results of the significance tests (Test 2) show that the classification performance of PBPSOfs is similar to BPSOfs in seven cases. Particularly, PBPSOfs is significantly better than BPSOfs on **all** the six datasets with a large number of features, and the number of features is much smaller in PBPSOfs than in BPSOfs. For example, on the Isolet5 dataset, PBPSOfs further reduced around 24.78% of the number of features selected by BPSOfs to reduce the average number of selected features from 225.15 to 169.35, but PBPSOfs significantly increased the classification accuracy. Meanwhile, in almost all cases, the standard deviation values of PBPSOfs is smaller than that of BPSOfs, which shows that PBPSOfs is more stable than BPSOfs.

The comparisons show that PBPSOfs using the newly developed updating mechanisms can better explore the search space of a feature selection task to further reduce the number of features and simultaneously maintain or increase the classification performance.

Comparisons with LFS and GSBS. Table 3 shows the results of LFS and GSBS. Both LFS and GSBS are deterministic methods, which produce a unique solution/feature subset on each dataset. In the table, "T" shows the results of the significance tests between the classification accuracy of LFS (or GSBS) and PBPSOfs. "+" (or "-") means the PBPSOfs is significantly better than LFS

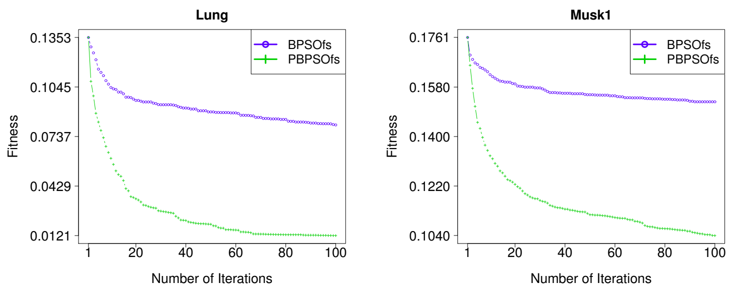

Fig. 1. Evolutionary Process of BPSOfs and PBPSOfs (colour).

or GSBS. Note that the results of GSBS on the Multiple Features (MultipleF.) dataset are not available because the experiment cannot finish within a week.

According to Table 3, it can be seen that PBPSOfs achieved significantly better classification performance than LFS on nine of the 14 datasets and similar classification performance on four datasets. The best accuracy of PBPSOfs is better than LFS on 13 out of the 14 datasets, although the number of features is larger. PBPSOfs selected a smaller or much number of features than GSBS in all cases and achieved significantly better classification performance than GSBS on 13 out of the 14 datasets. Only on the Lung dataset, the classification performance of LFS and GSBS is better than PBPSOfs. The main reason is that the Lung dataset has a small number of examples, where it is easy to have overfitting problems. PBPSOfs clearly has such a problem because it achieved the training accuracy of 100% in 39 of the 40 independent runs.

4.2 Results of PBPSOfs and BPSOfs in Training Process

Analysing the performance of PBPSOfs and BPSOfs in the training process can further show the search abilities of PBPSO and BPSO as general optimisation techniques rather than specific feature selection algorithms.

Evolutionary Process. We take the Lung and Musk1 datasets as two examples to analyse the evolutionary process. Other datasets show a similar pattern. Figure 1 shows the change of the *gbest* during the evolutionary process, where the horizontal axis shows the number of iterations and the vertical axis shows the average fitness value of the *gbest* in the 40 independent runs.

According to Figure 1, it can be observed that the *gbest* in PBPSOfs and BPSOfs have the same average fitness value at the first iteration because they were set to start from the same points for fair comparisons. However, even starting from the same points, PBPSOfs using the newly developed updating mechanisms can better explore the solution space to optimise (i.e. minimise) the fitness value to obtain much better results than BPSOfs. Considering the training process only, feature selection is a binary/combinatorial problem with a large and complex search space. The superior performance of PBPSOfs during the training process shows that the proposed PBPSO algorithm can successfully address difficult binary/combinatorial problems, as a general binary optimisation technique (not only specifically designed for feature selection tasks).

Table 4. Experimental Results in Training Process

Dataset	Method	AveSize	BestAcc	AveAcc ± StdAcc	Test 1	Test 2	Time
Australian	All	14	75.78				
	BPSOfs	3.18	88.2	83.25 ± 7.0772	+		4.91
	PBPSOfs	2.82	86.96	83.41 ± 7.7095	=	=	4.59
Zoo	All	17	86.72				
	BPSOfs	4.15	98.39	97.37 ± 0.3484	+		0.12
	PBPSOfs	3.35	97.99	97.33 ± 0.2327	=	=	0.11
Vehicle	All	18	88.18				
	BPSOfs	5.38	90.54	89.45 ± 0.6187	+		8.48
	PBPSOfs	5.02	90.37	89.44 ± 0.4998	=	=	8.28
German	All	24	80.14				
	BPSOfs	7.9	82.71	80.11 ± 2.3069	=		13.2
	PBPSOfs	5.95	82.71	79.7 ± 1.5259	-	-	12.3
WBCD	All	30	94.97				
	BPSOfs	7.92	96.73	95.6 ± 0.5919	+		4.62
	PBPSOfs	2.02	96.48	95.13 ± 0.396	=	-	3.32
Ionosphere	All	34	85.77				
	BPSOfs	8.92	93.5	91.09 ± 1.1403	+		1.92
	PBPSOfs	4.58	95.53	94.08 ± 0.7378	+	+	1.76
Lung	All	56	81.82				
	BPSOfs	23.28	100	95.68 ± 2.02	+		0.04
	PBPSOfs	6.78	100	99.89 ± 0.7097	+	+	0.02
Sonar	All	60	83.45				
	BPSOfs	23.02	93.1	89.12 ± 1.7137	+		0.97
	PBPSOfs	14.28	94.48	91.03 ± 1.895	+	+	0.77
Hillvalley	All	100	71.46				
	BPSOfs	39.35	73.94	72.28 ± 1.0503	+		50.28
	PBPSOfs	31.08	75.94	73.75 ± 1.0739	+	+	46.1
Musk1	All	166	92.19				
	BPSOfs	75.52	95.5	93.15 ± 1.1531	+		13.96
	PBPSOfs	69.3	97.3	94.83 ± 1.0976	+	+	12.7
Arrhythmia	All	279	94.79				
	BPSOfs	99.7	95.37	94.93 ± 0.2288	+		16.97
	PBPSOfs	63.42	95.86	95.55 ± 0.1657	+	+	12.44
Madelon	All	500	83.24				
	BPSOfs	243.2	87.69	85.93 ± 0.7274	+		991.6
	PBPSOfs	212.42	90.55	88.91 ± 0.6204	+	+	953.63
Isolet5	All	617	99.15				
	BPSOfs	225.15	99.28	99.14 ± 0.0826	=		384.16
	PBPSOfs	169.35	99.49	99.36 ± 0.0643	+	+	328.34
Multiple Features	All	649	99.36				
	BPSOfs	237.05	99.51	99.41 ± 0.0533	+		692.54
	PBPSOfs	176.15	99.63	99.52 ± 0.0542	+	+	551.44

Training Performance. Table 4 shows the experimental results of PBPSOfs and BPSOfs from the training process. The average size of the feature subsets in Table 4 is the same as in Table 2 because they are the same feature subsets, but their classification performances are different because they are used on different sets of data, i.e. the training set and test set, respectively. The average computational time of PBPSOfs and BPSOfs are also listed in the last column of Table 4, where the numbers are expressed in minutes.

From Table 4, it can be seen that both PBPSOfs and BPSOfs can reduce the number of features and maintain or even increase the training classification accuracy in almost all cases. Comparing PBPSOfs with BPSOfs, for **all** the fourteen datasets, PBPSOfs selected a smaller number of features than BPSOfs. On 12 out of the 14 datasets, the training classification accuracy of PBPSOfs is similar or significantly better than BPSOfs. Particularly, on **all** the nine datasets with more than 30 features/dimensions, PBPSOfs achieved significantly better performance than BPSOfs, i.e., selected a much smaller number of features and achieved significantly better classification accuracy. For some datasets, e.g. Lung, PBPSOfs has the problem of over-fitting and we will address this in future work.

The results show that PBPSOfs with the newly developed updating mechanisms can improve the search ability over BPSOfs, especially for the high-dimensional problems, where the search space is larger and more complex than low-dimensional problems.

4.3 Analysis on Computational Time

According to Table 4, it can be seen that on datasets with a small number of features or instances, both PBPSOfs and BPSOfs can finish the evolutionary feature selection process in a very short time, which is even less than one minute on the Zoo and Lung datasets. In **all** the 14 benchmark problems, PBPSOfs used a shorter time than BPSOfs. There are two main reasons. The first reason is that the newly developed PBPSO has a simpler updating equation than the standard BPSO. The second reason is that as wrapper approaches, the majority part of the computational time in PBPSOfs and BPSOfs are used on fitness evaluations, which needs to calculate the classification error rate of the selected features. For the same dataset, a large number of selected features needs longer time to calculate the error rate than a small number of selected features. Since PBPSOfs and BPSOfs have the same number of evaluations during the evolutionary training process and PBPSOfs usually selected a smaller number of features, PBPSOfs is faster than BPSOfs.

5 Conclusions and Future Work

This paper developed a new probability based updating mechanism based on which a new BPSO named PBPSO was proposed. A new feature selection approach named PBPSOfs was developed to maximise the classification accuracy and minimise the number of features. PBPSOfs was examined and compared with a standard BPSO based feature selection approach (BPSOfs) and two traditional feature selection algorithms on 14 datasets of varying difficulty. The experimental results show that PBPSOfs achieved better performance than the two traditional feature selection algorithms. Meanwhile, PBPSOfs outperformed BPSOfs in terms of both the classification performance and the number of features, especially on high-dimensional problems. The performances of PBPSOfs and BPSOfs in the training process show that PBPSO as a general method has better optimisation capability than standard BPSO. Additionally, PBPSOfs is computationally less expensive than BPSOfs due to its simple calculation of the new updating mechanism and selecting a smaller number of features. Overall, the newly developed PBPSO algorithm outperformed the standard BPSO in terms of both the effectiveness and the efficiency.

This work mainly focuses on binary problems, but in future, we will investigate a new PSO algorithm for general discrete problems. We will also further investigate and improve the performance of BPSO by developing new updating mechanisms and new representation or encoding schemes. From the classification point of view, the proposed algorithm may have over-fitting problems, which will also be addressed in future. Meanwhile, we also intend to develop a multi-objective feature selection approach to find a set of trade-off solutions to meet different requirements in real-world applications.

References

1. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis **1**(4) (1997) 131–156
2. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks. Volume 4. (1995) 1942–1948
3. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: IEEE International Conference on Evolutionary Computation (CEC'98). (1998) 69–73
4. Engelbrecht, A.P.: Computational intelligence: an introduction (2. ed.). Wiley (2007)
5. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation. Volume 5. (1997) 4104–4108
6. Xue, B., Zhang, M., Browne, W.: Particle swarm optimization for feature selection in classification: A multi-objective approach. IEEE Transactions on Cybernetics **43**(6) (2013) 1656–1671
7. Chuang, L.Y., Chang, H.W., Tu, C.J., Yang, C.H.: Improved binary PSO for feature selection using gene expression data. Computational Biology and Chemistry **32**(29) (2008) 29– 38
8. Xue, B., Cervante, L., Shang, L., Browne, W.N., Zhang, M.: A multi-objective particle swarm optimisation for filter based feature selection in classification problems. Connection Science (2012)
9. Sudholt, D., Witt, C.: Runtime analysis of binary PSO. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO'08), New York, NY, USA, ACM (2008) 135–142
10. Khanesar, M., Teshnehlab, M., Shoorehdeli, M.: A novel binary particle swarm optimization. In: Mediterranean Conference on Control Automation (MED'07). (2007) 1–6
11. Xue, B., Zhang, M., Browne, W.N.: New fitness functions in binary particle swarm optimisation for feature selection. In: IEEE Congress on Evolutionary Computation (CEC'12). (2012) 2145–2152
12. Gutlein, M., Frank, E., Hall, M., Karwath, A.: Large-scale attribute selection using wrappers. In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM '09). (2009) 332–339
13. Caruana, R., Freitag, D.: Greedy attribute selection. In: International Conference on Machine Learning (ICML'94), Morgan Kaufmann (1994) 28–36
14. Whitney, A.: A direct method of nonparametric measurement selection. IEEE Transactions on Computers **C-20**(9) (1971) 1100–1103
15. Marill, T., Green, D.: On the effectiveness of receptors in recognition systems. IEEE Transactions on Information Theory **9**(1) (1963) 11–17
16. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update. SIGKDD Explorations **11** (2009) 931–934
17. Frank, A., Asuncion, A.: UCI machine learning repository (2010)
18. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence **97** (1997) 273–324