Particle Swarm Optimisation for Feature Selection: A Hybrid Filter-Wrapper Approach

Tony Butler-Yeoman, Bing Xue, and Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
Email: {butlertony, Bing.Xue, Mengjie.Zhang}@ecs.vuw.ac.nz

***Abstract*—Feature selection is an important pre-processing step, which can reduce the dimensionality of a dataset and increase the accuracy and efficiency of a learning/classification algorithm. However, existing feature selection algorithms mainly wrappers and filters have their own advantages and disadvantages. This paper proposes two filter-wrapper hybrid feature selection algorithms based on particle swarm optimisation (PSO), where the first algorithm named FastPSO combined filter and wrapper into the search process of PSO for feature selection with most of the evaluations as filters and a small number of evaluations as wrappers. The second algorithm named RapidPSO further reduced the number of wrapper evaluations. Theoretical analysis on FastPSO and RapidPSO is conducted to investigate their complexity. FastPSO and RapidPSO are compared with a pure wrapper algorithm named WrapperPSO and a pure filter algorithm named FilterPSO on nine benchmark datasets of varying difficulty. The experimental results show that both FastPSO and RapidPSO can successfully reduce the number of features and simultaneously increase the classification performance over using all features. The two proposed algorithms maintain the high classification performance achieved by WrapperPSO and significantly reduce the computational time, although the number of features is larger. At the same time, they increase the classification accuracy of FilterPSO and reduce the number of features, but increased the computational cost. FastPSO outperformed RapidPSO in terms of the classification accuracy and the number of features, but increased the computational time, which shows the trade-off between the efficiency and effectiveness.**

I. Introduction

Classification tasks are often described by a large number of features, so as to represent the target concept as completely as possible. However, many features are redundant or irrelevant, resulting in noise in the dataset that reduces the performance of many classification algorithms [1]. Furthermore, the large number of features contributes to the "curse of dimensionality", a major problem for classification in general. Feature selection is the process of choosing a subset of the original features from data. The chosen feature subset should be small and accurately describe the target concept. As a preprocessing step, feature selection is a practical and well-known solution to the problems of high-dimensional data, resulting in a fast and better classification process.

Finding the optimal feature subset is a difficult task. The search cannot be performed exhaustively, as the search space contains 2^n possible feature subsets for a dataset with n features. Feature interaction problems, which occur frequently in classification tasks, increase the complexity of the search space. Most of the existing feature selection algorithms suffer from the problems of being stagnation into local optima. Evolutionary computation (EC) techniques are well-known for their global search ability, and have been used effectively to solve feature selection problems. Examples of these includes genetic algorithms (GAs) [2], genetic programming (GP) [3], and particle swarm optimisation (PSO) [4]. Compared with GAs and GP, PSO is easier to implement, has fewer parameters, computationally less expensive, and can converge more quickly. So, in recent years, PSO has attracted much attention from researchers for solving feature selection problems.

978-1-4799-7492-4/15/$31.00

Feature selection algorithms broadly fall into two categories, wrapper and filter approaches [1], which differ in their evaluations. Wrapper approaches use a learning/classification algorithm to evaluate the quality of a particular feature subset, while filter approaches use data-intrinsic measures (independent of any learning algorithm). Common measures for a filter approach include information theory, dependency, and consistency measures [1]. Wrapper approaches often yield better classification performance than filter approaches due to the direct link between the feature subsets and the classification algorithm. However, this comes with expensive computational cost. Filters are computationally cheap, but may not achieve good classification performance since the filter measure cannot perfectly reflect the performance of a classification algorithm. Therefore, finding a way to combine filters and wrappers into a single process to utilise their advantages and avoid their disadvantages is expected to improve the performance. Although there are a few works [5], [6] to combine wrappers with filters to improve the performance, all of them are performed by adding extra calculation to wrappers, which further increase the computational cost. Finding a way to combine filters and wrappers to maintain the classification performance of wrappers and simultaneously reduce the computational cost is still an open issue.

A. Goals

This paper aims to explore combinations of filter and wrapper methods in PSO for feature selection with the goal of maintaining the classification performance of wrappers while approaching the efficiency of filters. To achieve this goal, we will develop two new PSO-based feature selection algorithms, where in the first algorithm (named FastPSO), the majority of the evaluations are based on filters and the wrapper evaluations are performed only when a better (in terms of the filter measure) solution is found. The second algorithm (named RapidPSO) further reduces the times of wrapper evaluations by partially applying the wrapper evaluations when better

solutions are found. The two proposed algorithms will be evaluated and compared with a PSO-based pure filter method and a PSO-based pure wrapper method on a number of benchmark datasets. Specifically, we will investigate:

- whether FastPSO and RapidPSO can successfully reduce the number of features and maintain or even increase the classification accuracy over using all features,
- whether FastPSO can maintain the classification performance achieved by the PSO-based wrapper approach and use a significantly shorter time,
- whether RapidPSO can further reduce the computational time over FastPSO and still achieved better classification performance than the PSO-based filter algorithm, and

II. Background

A. Particle Swarm Optimisation

PSO is an EC technique proposed by Kennedy and Eberhart in 1995 [7]. PSO maintains a population of *particles*, called a *swarm*, each of which encodes a candidate solution. PSO initialises each particle in the swarm to a random position in the space, and iterates the position of each particle based on the experience of the particle and its neighbours. The position of particle i is represented by a vector $x_i = (x_{i,1}, \ldots, x_{i,n})$ where n is the dimensionality of the search space. The velocity is represented by a similar vector $v_i = (v_{i,1}, \ldots, v_{i,n})$ where each component of the vector is limited to a predefined range $[-v_max, v_max]$. The best previous position of particle i is recorded as the personal best, $pbest_i = (p_{i,1} \ldots p_{i,n})$, and the best position found by the whole population is recorded the global best, $gbest = (g_1 \ldots g_n)$. PSO updates the velocity and position of each particle according to the following equations:

$$x_{i,d}^{t+1} = x_{i,d}^{t} + v_{i,d}^{t+1} \tag{1}$$

$$v_{i,d}^{t+1} = w \cdot v_{i,d}^{t} + c_1 \cdot r_{1,i} \cdot (p_{i,d} - x_{i,d}^{t}) + c_2 \cdot r_{2,i} \cdot (g_d - x_{i,d}^{t}) \tag{2}$$

where d with $0 < d \leq n$ denotes the dimension of in the position or velocity vector, and t represents the t-th iteration. w is a predefined constant for the inertia weight, and c_1 and c_2 are predefined acceleration constants. Each $r_{1,i}$ and $r_{2,i}$ are random values uniformly distributed over $[0, 1]$. The original PSO is applicable to real-valued search spaces. However, feature selection, along with many other problems, occur in a discrete search space and require a discrete search algorithm. Binary PSO (BPSO) was an discrete version of the PSO algorithm [8], where the values of all the position vectors (i.e. x_i, $pbest_i$, and $gbest_i$) are restricted to 0 or 1. Equation 2 is still used to update the velocity. Each component in the velocity is transformed to (0,1), which indicates the probability of the corresponding component in the position vector being 1. BPSO updates the position of each particle according to the following equation:

$$x_{i,d} = \begin{cases} 1, & rand() < \dfrac{1}{1 + e^{-v_{i,d}}} \\ 0, & otherwise \end{cases} \tag{3}$$

B. Information Theory

The tools of information theory [9] are the principal methods to measure the information content of random variables, which can be used to measure the quality of feature subsets. A core information measure is *entropy*, $H(X)$, which measures the uncertainty of a discrete random variable X. It is defined as:

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \cdot log\ p(x) \tag{4}$$

where $\mathcal{X}$ is the set of values that X can take. *Conditional entropy* measures the remaining uncertainty in a discrete random variable X when another Y is known. This is defined as:

$$H(X|Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \cdot log\ p(x|y) \tag{5}$$

If Y completely determines X then the conditional entropy is zero, indicating no further information is required to fully describe X. On the other hand, if $H(X|Y) = H(X)$ then X and Y are completely independent, i.e. no extra information about X is gained from knowing Y.

Mutual information, $I(X;Y)$, determines how much information can be gained about Y given knowledge of X, which is defined as:

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \cdot log\ \frac{p(x, y)}{p(x)p(y)} \end{aligned} \tag{6}$$

If knowing X gives no extra information about Y, i.e. the two variables are independent, then $I(X;Y)$ will be zero. Otherwise, $I(X;Y)$ will be large if the two variables are closely related.

C. PSO for Feature Selection

EC techniques have been broadly applied to feature selection problems, typically PSO [10], [11], [12], GAs [2], [13], and GP [3]. Due to page limit, this section focuses mainly on PSO for feature selection and other EC-based works can be seen from [14], [15].

PSO for Wrapper Feature Selection: A myriad of BPSO approaches have been applied to feature selection, a selection of which are featured here. Chuang et al. [16] proposed a feature selection method that resets $gbest$ to the zero vector if it is unchanged for too many iterations. In a similar vein, Yang et al. [17] propose that $gbest$ of a particle is forcibly changed if unchanged for three iterations. The new $gbest$ is formed from a combination of the individual $pbest$s of relevant particles. Experiments show that the new algorithm often performs better than the standard BPSO algorithm. Xue et al. [4] present a multi-objective to approach to feature selection, optimising both classification accuracy and subset size. The results show one method in particular outperforming both traditional PSO feature selection algorithms, and some other common multi-objective methods, in terms of accuracy. Liu et al. [18] have studied *multi-swarm* PSO (MSPSO) for feature

Algorithm 1 Pseudo-code of FastPSO

f: the fitness value of filter evaluation, Equation (7);
w: the fitness value wrapper evaluation, Equation (8);
randomly initialise the particles in the swarm;
evaluate the f and w values for each particle;
initialise the $pbest$ of each particle and calculate their corresponding f' and w' values;
for each iteration **do**
    for each particle p **do**
        $f \leftarrow$ filter evaluation of position of p;
        if $f > f'$ **then**
            $w \leftarrow$ wrapper evaluation of position of p;
            if $w > w'$ **then**
                update $pbest$ of p to the position of p;
                update f' and w';
    for each particle p **do**
        update the $gbest$ of p;
    for each particle p **do**
        update the velocity of p using Equation (2) ;
        update the position of p using Equation (3);
collect the features selected by $gbest$ and calculate its classification performance on the test set;
return the $gbest$, the features selected by $gbest$, the training and testing classification performance.

selection, where experiments indicate IFS performs better than both traditional PSO- and GA-based feature selection in terms of classification accuracy and subset size.

PSO for Filter Feature Selection: A number of different fitness criteria have been used to propose filter approaches. Peng et al. [19] have proposed an information-theoretic filter approach, called Min-Redundancy Max-Relevance (mRMR), based on maximising relevancy and minimising redundancy. Due to its roots in information theory, mRMR is computationally inexpensive. Cervante et al. [20] have also proposed information-theoretic fitness functions for use in feature selection, evaluating groups instead of pairs of features. Wang at al. [21] proposed a PSO based filter approach based on rough set theory, which reduced the dimensionality but suffered from the problem of high computational cost.

III. Proposed Hybrid PSO Approach

Wrapper approaches have the advantages of obtaining high classification performance, but computationally expensive. Filter approaches are computationally cheap, but the filter measure might not fully reflect the classification performance (of a particular classifier). To utilise their advantages, we propose two hybrid PSO feature selection algorithms, in which both filter and wrapper approaches are used together to evaluate the fitness of feature subsets.

A. New Algorithm 1: FastPSO

To combine filter and wrapper evaluations into the search process of PSO for feature selection, we proposed a new algorithm where the majority of the evaluations are based on the filter fitness function and only a small proportion of the evaluations are based on the wrapper fitness function. Since filter evaluations are fast, we expect this new algorithm to be fast and also effective in terms of the classification performance due to the use of the wrapper fitness function. This algorithm is called FastPSO.

Algorithm 2 Pseudo-code of RapidPSO

initialise the swarm, f, w, f' and w' values;
for each iteration **do**
    for each particle p **do**
        $f \leftarrow$ filter evaluation of position of p;
        if $f > f'$ **then**
            if $(f - f') > L$ **then**
                update $pbest$ of p to the position of p;
                update f';
            else
                $w \leftarrow$ wrapper evaluation of position of p;
                if $w > w'$ **then**
                    update $pbest$ of p to the position of p;
                    update f' and w';
    $U \leftarrow$ list of all particles with updated $pbest$s;
    sort U in a descending order of filter evaluations;
    for the first u particles of U **do**
        ensure the wrapper evaluations are performed on their $pbest$s;
    update $gbest$ according to the best wrapper evaluation of the $pbest$ of particles in U;
    for each particle p **do**
        update the velocity of p using Equation (2);
        update the position of p using Equation (3);
return the $gbest$, the features selected by $gbest$, the training and testing classification performance.

FastPSO is guided by a two-step evaluation procedure involving two fitness functions, i.e. Equation (7) for filter and Equation (8) for wrapper. The evaluations are mainly based on the filter fitness function and the wrapper fitness function is introduced when a better solution is found during the search process. This is achieved by observing the change of the filter evaluation values, and when a particle finds a better $pbest$, the wrapper fitness function will be introduced. Algorithm 1 shows the pseudo-code of the FastPSO algorithm. FastPSO follows the basic structure of a PSO algorithm, but the key idea relies on the two-step evaluation. In each iteration, the filter fitness function is first applied to evaluate the current position of each particle. Only if the particle's position is better than its $pbest$ in terms of the filter evaluation value, the wrapper fitness function is applied to evaluate the classification performance of the particle's current position. If the particle's current position is better than its $pbest$ in terms of the wrapper evaluation value, i.e. the classification accuracy, the $pbest$ is updated to the particle's current position. $gbest$ is updated if any updated $pbest$ is better than the current $gbest$ in terms of the wrapper evaluation. During the search process, each $pbest$ and $gbest$ will have two fitness values, a filter evaluation and wrapper evaluation, which is different from standard PSO for feature selection.

B. New Algorithm 2: RapidPSO

To further investigate the combination of filter and wrapper, we develop a new strategy to reduce the use of the wrapper fitness function during the search process of PSO for feature selection. The second new algorithm is called RapidPSO here.

Algorithm 2 shows the pseudo-code of RapidPSO, which shows that the filter fitness function is always performed in each evaluation, the same as FastPSO. The key idea in RapidPSO is the use of the wrapper fitness function, which

is shown from Line 5 to Line 16, which includes two specific situations that need wrapper evaluations. The first situation is when the filter fitness value of the current particle is better than its $pbest$, which is similar to FastPSO. However, instead of always introducing the wrapper fitness function, RapidPSO considers a special case where the improvement of the filter evaluation is larger than a certain amount L. We assume that the large improvement in the filter evaluation is highly likely to cause an improvement in the wrapper evaluation. Therefore, the wrapper evaluation is not needed to perform. $pbest$ of the particle is directly update to the particle's current position, which further reduces the probability of performing the wrapper evaluations. This can be seen from Line 6 to Line 9. For the first situation, there may be some $pbest$s without a corresponding wrapper fitness value, which is hard to guarantee that $gbest$ is the solution with the highest classification performance. So the wrapper evaluation is considered in the second situation, which is to ensure that the best $pbest$s have their corresponding wrapper fitness values stored. This is achieved by sorting all the $pbest$s in a descending order according to their filter-evaluation values, and perform the wrapper evaluation on the top ranked $pbest$ that do not have a wrapper fitness value.

The difference between FastPSO and RapidPSO can be seen by comparing Algorithms 1 with 2. RapidPSO performs a "blind $pbest$ update" step (i.e. without wrapper evaluations) as shown from Line 6 to Line 9, which reduces the number of wrapper evaluations over FastPSO. Consequently, RapidPSO performs the wrapper evaluation to ensure that (only) the good $pbest$s have a classification accuracy, which is shown from Line 14 to Line 17 in Algorithm 2.

C. Fitness Functions

a) Filter Evaluation: The goal of using filter approaches is to speed up the fitness evaluation procedures. So a computationally cheap measure, mutual information, is employed here to form the filter evaluation, which is based on the idea of the Min-Redundancy Max-Relevance (mRMR) criterion [19]. The filter fitness function aims to maximise the relevance of the selected features to the class labels and also to minimise the redundancy among the selected feature subset. Furthermore, minimising the size of the feature subset is also considered in the fitness function with a small weighting factor, which should be small enough to only be significant in tie-breaker situations.

Equation (7) shows the filter fitness function, where D is the relevancy of the selected features, R is the redundancy among the selected features, $|X|$ represents the size of the feature subset, and α is the weighting factor. Equation (7) is a maximisation function. PSO using Equation (7) as the fitness function for feature selection is called FilterPSO here. Note that FilterPSO is similar to the algorithm proposed in [20], but with the extra component of $\alpha \cdot |X|$ to ensure that if two feature subsets have the same value for $(D - R)$, the smaller subset will have a better fitness value.

$$F_1(X) = D - R - \alpha \cdot |X| \tag{7}$$

where

$$D = \sum_{x \in X} I(x; c); \qquad R = \sum_{x,y \in X} I(x; y)$$

where x, y are individual features in X, and c is the class label.

b) Wrapper Evaluation: The wrapper fitness function is to maximise the classification accuracy of the selected feature subset, which is calculated by Equation (8). The accuracy is calculated by using the number of correctly classified instances divided by the total number of instances. $\alpha \cdot |X|$ is the same as in Equation (7). PSO using Equation (8) as the fitness function for feature selection is called WrapperPSO.

$$F_2(X) = Accuracy - \alpha \cdot |X| \tag{8}$$

IV. Theoretical Analysis

In this section, we take FastPSO as an example to analyse and approximate the computational complexity. This analysis is limited to the number of *wrapper evaluations* since they usually take the majority of the computational cost in feature selection approaches, and ignores the size of the feature subset in those evaluations since it is dataset dependent.

We aim to approximate the mean number of wrapper evaluations performed by FastPSO in each run. The core problem is that of the probability of a wrapper evaluation being required at the i-th iteration of a particular particle, based on which it is easy to calculate the overall speed improvement of FastPSO over the standard wrapper approach, i.e. WrapperPSO. For a given particle and iteration, a wrapper evaluation is performed if, and only if, the filter evaluation of that particle's position is an improvement over the filter evaluation of this particle's $pbest$. So the problem further reduces to find the probability of the filter evaluation of a particle being better than that of its $pbest$ at the i-th iteration. However, this is almost an impractical task since the filter evaluation function, F_1, is highly dependent on both the dataset and on past filter evaluations. This is certainly far too complicated to analyse, so some assumptions must be made.

Suppose F_1 follows continuous distribution that supports (a subset of) the range $[0, 1]$. Furthermore, suppose that each evaluation of F_1 is independent of all others. Let p be a particular particle and m be a particular iteration, and let $X_1 \dots X_m$ be random variables representing the evaluations of F_1 at each iteration 1 to i respectively. The probability that, at the m-th iteration, a wrapper evaluation will be required is the probability that X_m is larger than all $X_1 \dots X_{m-1}$ (F_1 is a maximisation function). We claim the following:

$$P(X_m > X_1 \dots X_{m-1}) = \frac{1}{m} \tag{9}$$

This is proven using the Inclusion-Exclusion Principle [22]. Let E_i be the event that X_i is all the largest value among $X_1 \dots X_m$. At least one E_i must hold:

$$P(E_1 \cup \dots \cup E_m) = 1 \tag{10}$$

Since the distribution of F is continuous, no two of the events can be simultaneously true and so $P(E_i \cap E_j) = 0$ for all distinct i, j. By the Inclusion-Exclusion Principle:

$$P(E_1 \cup \ldots \cup E_m) = \sum_{i=1}^{m} P(E_i) - \sum_{i<j} P(E_i \cap E_j) + \ldots \quad (11)$$

However, as the probability of any two events occurring simultaneously is 0, all but the first sum evaluate to 0. Hence:

$$1 = P(E_1 \cup \ldots \cup E_m) = \sum_{i=1}^{m} P(E_i) \quad (12)$$

but since each random variable X_i is i.i.d. $P(E_i) = P(E_j)$ for all i, j. Therefore:

$$1 = \sum_{i=1}^{m} P(E_i) = mP(E_n) \quad (13)$$

So $P(E_m) = P(X_m > X_1, \ldots, X_{m-1}) = \frac{1}{m}$.

Let M be the total number of iterations, and P be the number of particles in the swarm. Under the assumption that each X_i is an i.i.d. random sample, the average probability of a wrapper evaluation is:

$$P(evaluation) = \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{1}{i} \quad (14)$$

Therefore, making the expected number of wrapper evaluations:

$$M \cdot P \cdot \frac{1}{M} \cdot \sum_{i=1}^{M} \frac{1}{i} = PH_M \quad (15)$$

where H_m is the m-th harmonic number. Hence, with 30 particles and 50 iterations, the expected number of wrapper evaluations is roughly 135.

This provides a lower bound on the true mean of the number of wrapper evaluations required, no matter what the distribution of fitness values is. However, this is clearly an unrealistic model of the algorithm. The evaluations of F_1 are *not* i.i.d. because previous evaluations will inform the velocity of the particle and thus affect future evaluations. Thus, any increase in number of evaluations above PH_M is the result of PSO performing a more highly informed search than pure random sampling.

V. Experimental Design

To test FastPSO and RapidPSO, they are compared with WrapperPSO and FilterPSO. WrapperPSO, FastPSO, RapidPSO, and FilterPSO represents four algorithms, where the number of wrapper evaluations is decreased from always performing to not performing at all. The four algorithms are compared on nine datasets chosen from the UCI machine learning repository [23], where the details are shown in Table I. The nine datasets were chosen to represent a range of features, instances, and classes that the algorithms can be applied to. For each dataset, the instances are randomly divided into $2/3$ as the training set and $1/3$ for as the test set such that the class distribution is approximately maintained [15]. In order to maintain reasonably balanced classes, the Gas 6 dataset has been created from the Gas Sensor Array Drift dataset by taking batch 6 and removing all data with class labels 3, 4, and 6. The Isolet 5 dataset is created by using only the validation and training set of the Isolet 5 dataset since the provided test set does not include the class labels.

TABLE I. Datasets

Dataset	# Features	# Instances	# Classes
Wine	13	178	3
WDBC	30	569	2
Ionosphere	34	351	2
Splice	61	3190	4
Hill-Valley	100	606	2
Gas 6	128	1694	3
Musk 1	166	476	2
Madelon	500	2600	2
Isolet 5	617	1599	26

To perform the wrapper evaluation, a classification algorithm is needed to calculate the classification accuracy. There are many options, such as K-nearest neighbour (KNN), Decision Trees, Support Vector Machines, and Naive Bayes. KNN was chosen with $k = 1$ (1NN) due to its simplicity and widely use in existing papers [4], [16]. Since some datasets have a relatively small number of instances in the training set, each wrapper evaluation uses 10-fold cross validation on the training set to avoid feature selection bias. The mutual information based filter evaluation works on discrete rather than continuous features. To accommodate this, discretised versions of each dataset are used for the experiments, the procedure is as follows. The range upon which the values of a feature lie is divided into 20 bins with each bin, except perhaps the last, containing an equal number of values. If a feature takes on fewer than 20 values total, it is left unchanged.

All four tested algorithms, WrapperPSO, FilterPSO, FastPSO, and RapidPSO, use the following the parameters suggested in [24]: inertia weight $w = 0.7298$, acceleration constants $c_1 = c_2 = 1.49618$, maximum velocity $v_{max} = 6$, population size $P = 30$ and maximum iterations $T = 50$. Fully connected topology is used in all the four algorithms. Furthermore, $\alpha = 10^{-8}$, the extra parameters of RapidPSO are set to $L = 0.1$ and $u = 3$. Each algorithm was applied to each dataset for 30 independent runs. A non-parametric statistical significance test, Wilcoxon test, was used to compare the classification accuracy achieved by using all features for classification, and that of the feature subsets selected by WrapperPSO, FastPSO, RapidPSO, and FilterPSO.

VI. Results and Discussions

Table II summarises the classification accuracy and the feature subset size of the four algorithms and that of original feature set, where "mean $\pm$ stdev" shows the average and the standard deviation of the the results from the 30 independent runs. Table III shows the results of the statistical significance test between each pair of algorithms in terms of the classification accuracy and the number of features. Table IV summarises the number of wrapper evaluations and the computational time used by the four algorithms.

A. Results of WrapperPSO and FilterPSO

According to Tables II and III, WrapperPSO substantially reduced the feature subset size, which is more than half in almost all cases. With the small subsets selected

TABLE II. EXPERIMENTAL RESULTS

Dataset	Method	Size mean ± stdev	Accuracy (%) mean ± stdev	Accuracy(%) Best
Wine	All	13	96.61	96.61
	WrapperPSO	7.3 ± 0.8	97.40 ± 2.30	100.00
	FastPSO	7.8 ± 1.1	98.25 ± 1.86	100.00
	RapidPSO	9.0 ± 1.5	97.91 ± 2.12	100.00
	FilterPSO	8.5 ± 3.5	94.29 ± 5.03	100.00
WDBC	All	30	97.89	97.89
	WrapperPSO	14.5 ± 2.1	97.11 ± 0.85	98.42
	FastPSO	17.8 ± 2.1	97.56 ± 0.72	99.47
	RapidPSO	19.6 ± 3.1	97.53 ± 0.69	98.95
	FilterPSO	27.7 ± 0.7	97.88 ± 0.35	98.95
Ionosphere	All	34	82.05	82.05
	WrapperPSO	11.0 ± 1.8	84.70 ± 2.86	89.74
	FastPSO	15.9 ± 2.5	84.10 ± 2.84	88.03
	RapidPSO	19.4 ± 4.0	82.68 ± 2.29	86.32
	FilterPSO	30.5 ± 0.7	81.17 ± 1.77	86.32
Splice	All	60	74.39	74.39
	WrapperPSO	23.6 ± 3.1	78.42 ± 1.05	81.10
	FastPSO	28.5 ± 3.2	77.80 ± 1.17	79.68
	RapidPSO	28.2 ± 3.4	77.49 ± 1.24	80.43
	FilterPSO	38.1 ± 3.4	76.22 ± 1.02	78.73
Hill-Valley	All	100	55.45	55.45
	WrapperPSO	50.4 ± 6.2	54.12 ± 1.38	58.17
	FastPSO	56.1 ± 4.3	54.13 ± 1.14	56.44
	RapidPSO	61.0 ± 5.7	54.00 ± 1.55	56.68
	FilterPSO	48.9 ± 6.2	54.53 ± 1.26	57.43
Gas 6	All	128	99.82	99.82
	WrapperPSO	42.0 ± 2.4	99.86 ± 0.13	100.00
	FastPSO	55.4 ± 3.8	99.88 ± 0.12	100.00
	RapidPSO	63.2 ± 6.4	99.91 ± 0.09	100.00
	FilterPSO	66.5 ± 5.8	99.87 ± 0.11	100.00
Musk1	All	166	76.10	76.10
	WrapperPSO	81.9 ± 7.0	79.41 ± 2.16	85.53
	FastPSO	91.1 ± 7.8	78.99 ± 1.84	82.39
	RapidPSO	94.4 ± 9.0	78.41 ± 1.96	83.02
	FilterPSO	82.9 ± 5.9	78.05 ± 3.02	85.53
Madelon	All	500	52.60	52.60
	WrapperPSO	245.5 ± 9.6	54.08 ± 1.46	57.32
	FastPSO	258.5 ± 8.8	54.41 ± 1.86	58.13
	RapidPSO	268.6 ± 11.0	54.00 ± 2.08	57.55
	FilterPSO	307.5 ± 3.9	52.48 ± 1.59	55.71
Isolet 5	All	617	77.12	77.12
	WrapperPSO	306.6 ± 11.4	78.01 ± 1.04	79.81
	FastPSO	316.1 ± 12.3	77.71 ± 1.03	79.62
	RapidPSO	326.8 ± 10.6	77.33 ± 0.88	78.65
	FilterPSO	369.9 ± 4.8	75.87 ± 1.13	78.27

TABLE III. SIGNIFICANCE TESTS

Dataset	Method A	Method B					
		WrapperPSO		RapidPSO		FastPSO	
		Acc	Size	Acc	Size	Acc	Size
Wine	All	◇	◇	◇	◇	◇	◇
	FilterPSO	◇	◇	◇	≈	◇	≈
	FastPSO	∗	◇	≈	∗		
	RapidPSO	≈	◇				
WDBC	All	∗	◇	∗	◇	∗	◇
	FilterPSO	∗	◇	∗	◇	∗	◇
	FastPSO	∗	◇	≈	∗		
	RapidPSO	∗	◇				
Ionosphere	All	◇	◇	≈	◇	◇	◇
	FilterPSO	◇	◇	◇	◇	◇	◇
	FastPSO	≈	◇	∗	∗		
	RapidPSO	◇	◇				
Splice	All	◇	◇	◇	◇	◇	◇
	FilterPSO	◇	◇	◇	◇	◇	◇
	FastPSO	◇	◇	≈	≈		
	RapidPSO	◇	◇				
Hillvalley	All	∗	◇	∗	◇	∗	◇
	FilterPSO	≈	≈	≈	∗	≈	∗
	FastPSO	≈	◇	≈	∗		
	RapidPSO	≈	◇				
Gas 6	All	◇	◇	◇	◇	◇	◇
	FilterPSO	≈	◇	◇	◇	≈	◇
	FastPSO	≈	◇	≈	∗		
	RapidPSO	∗	◇				
Musk1	All	◇	◇	◇	◇	◇	◇
	FilterPSO	◇	≈	≈	∗	≈	∗
	FastPSO	≈	◇	≈	≈		
	RapidPSO	◇	◇				
Madelon	All	◇	◇	◇	◇	◇	◇
	FilterPSO	◇	◇	◇	◇	◇	◇
	FastPSO	≈	◇	≈	∗		
	RapidPSO	≈	◇				
Isolet 5	All	◇	◇	≈	◇	◇	◇
	FilterPSO	◇	◇	◇	◇	◇	◇
	FastPSO	≈	◇	∗	∗		
	RapidPSO	◇	◇				

∗: Method A is significantly better than B, i.e. higher accuracy or smaller size;
◇: Method A is significantly worse than B, i.e. lower accuracy or larger size;
≈: Methods A and B are similar to each other, i.e. similar accuracy or size.

by WrapperPSO, the classification performance of 1NN was significantly increased in 7 out of the 9 datasets. The results suggest that PSO guided by the classsification performance can effectively explore the seach space of the feature selection problems to reduce the dimensionality and increase the classification performance.

As a filter approach, FilterPSO maintained similar classification performance to using all features, but reduced the number of features in all cases. This indicates that FilterPSO is an effective algorithm for feature selection. However, there is still a statistically significant difference in classification accuracy, favouring WraperPSO, on the Wine, Ionosphere, Splice, Musk 1, Madelon, and Isolet 5 datasets. The filter approach generated significantly larger feature subsets on all but the Hill-Valley and Musk 1 datasets, for which FilterPSO produced anomalously small subsets. On some datasets, this size difference is very large, for example 61% larger on the Splice dataset. The best accuracy found, while not a particularly important statistic, tends to be higher in WrapperPSO. The results show that WrapperPSO directly used the accuracy as the fitness function can achieved better classification performance than FilterPSO.

B. Results of FastPSO

Tables II and III shows that the number of features selected by FastPSO is much smaller (around half) than the total number of features, and maintain or even significantly increase the classification accuracy on 8 out of the 9 datasets.

Compared with WrapperPSO, FastPSO performed statistically at *least* as well as WrapperPSO in terms of accuracy on all but the Splice dataset. FastPSO also produced larger feature subsets, but the size increase is often small, in the range of 10% in most cases. Comparing FastPSO with FilterPSO, FastPSO outperformed FilterPSO in terms of the classification performance and the feature subset size in most cases. The comparisons show that FastPSO combined the filter and wrapper evaluations together to guide the search process of PSO for feature selection can achieve better performance than FilterPSO and maintain the classification accuracy as high as WrapperPSO. Although the number of features is larger than WrapperPSO, the computational time is expected to be much shorter in FastPSO because of the majority of evaluations are filter evaluations. The results suggest that FastPSO could be

useful in situations where accuracy and speed are important but subset size is not critical.

C. Results of RapidPSO

According to Tables II and III, the classification performance of RapidPSO is similar or significantly better than using all features in 7 out of the 9 datasets and reduce the number of features in all cases.

RapidPSO found subsets with significantly worse accuracy than WrapperPSO on the Ionosphere, Splice, Musk 1, and Isolet 5 datasets, but similar on the remaining five datasets. As expected RapidPSO performed at least as well as FilterPSO except for WDBC, in which the difference is very small. The comparison between FastPSO and RapidPSO is fairly even, they are not distinguishable in terms of accuracy on any datasets except for Ionosphere and Isolet 5, which FastPSO slightly outperforms RapidPSO. Following the trend, RapidPSO tends to produce larger feature subsets than WrapperPSO and FastPSO, but smaller subsets than FilterPSO; with Hill-Valley and Musk 1 being the exceptions due to their unusually small FilterPSO subset sizes. This increase is generally around 20%, with larger subsets being found on the WDBC, Ionosphere, and Gas 6 datasets.

D. Analysis on Computational Cost

Table IV summarises the computational cost used by the four PSO-based algorithms, which includes the CPU time and the number of wrapper evaluations. The filter evaluation is very cheap and the wrapper evaluation is very expensive since each evaluation needs a 10-fold cross validation process for classification to get the accuracy.

Table IV shows that on all the 9 datasets, the number of wrapper evaluations decreases from WrapperPSO, FastPSO, RapidPSO to FilterPSO, which leads to the decrease in the computational time. Specifically, FastPSO shows an improvement between 18% and 66% in running time over WrapperPSO on every dataset. The algorithm is, as expected, still much slower than FilterPSO. RapidPSO universally runs more quickly than both WrapperPSO and FastPSO, while running more slowly than FilterPSO. The improvement of RapidPSO over FastPSO is mixed, taking between less than half and almost equal amounts of time. The analysis performed in Section IV gives a lower bound on the number of wrapper evaluations required by FastPSO, but the experiments show that the algorithm requires several times this number. This difference is due to FastPSO not performing i.i.d. random sampling of the search space, which suggests that FastPSO is performing a much more informed search. RapidPSO uses far fewer wrapper evaluations than FastPSO on average in most tests, indicating that it may have earned its title. However, in general, the number of wrapper evaluations that RapidPSO requires decreases more than the actual running time when compared to FastPSO. This could be due to the sorting process required in each iteration in RapidPSO, and/or because RapidPSO tends to evaluate larger subsets leading to more computationally expensive wrapper evaluations.

The results in Tables II, III and IV show that by combining filter and wrapper evaluations together into the search process of PSO, it can maintain the classification performance and reduce the computational cost, although the cost is a slightly increase in the feature subset size. The feature subset size could be reduced by using a new fitness function or another filter measure, which is an interesting direction for future work.

TABLE IV. EXPERIMENTAL EFFICIENCY RESULTS

Dataset	Method	Wrapper evaluations mean (± stdev)	Time (ms) mean ± stdev
Wine	WrapperPSO	1500.0	1881 ± 397
	FastPSO	321.4 ± 35.1	695 ± 99
	RapidPSO	113.5 ± 12.1	433 ± 64
	FilterPSO	0.0	81 ± 26
WDBC	WrapperPSO	1500.0	15384 ± 1629
	FastPSO	387.0 ± 51.2	5308 ± 918
	RapidPSO	136.8 ± 14.8	2486 ± 370
	FilterPSO	0.0	571 ± 77
Ionosphere	WrapperPSO	1500.0	7207 ± 922
	FastPSO	437.6 ± 63.5	2826 ± 474
	RapidPSO	137.6 ± 21.6	1362 ± 235
	FilterPSO	0.0	375 ± 50
Splice	WrapperPSO	1500.0	991025 ± 109723
	FastPSO	254.1 ± 17.4	196257 ± 23957
	RapidPSO	201.9 ± 16.5	157885 ± 19392
	FilterPSO	0.0	1062 ± 53
Hill-Valley	WrapperPSO	1500.0	51520 ± 6277
	FastPSO	453.5 ± 60.5	29622 ± 3168
	RapidPSO	129.6 ± 11.9	18971 ± 1675
	FilterPSO	0.0	13883 ± 1335
Gas 6	WrapperPSO	1500.0	168903 ± 4096
	FastPSO	555.7 ± 64.8	111303 ± 13553
	RapidPSO	123.6 ± 20.1	49983 ± 4127
	FilterPSO	0.0	33847 ± 2098
Musk 1	WrapperPSO	1500.0	43513 ± 2770
	FastPSO	468.4 ± 67.0	28218 ± 2637
	RapidPSO	166.6 ± 24.6	20235 ± 2442
	FilterPSO	0.0	14187 ± 1718
Madelon	WrapperPSO	1500.0	8404234 ± 132761
	FastPSO	608.9 ± 71.1	5370957 ± 715122
	RapidPSO	325.7 ± 43.3	3214323 ± 477870
	FilterPSO	0.0	1078671 ± 205485
Isolet 5	WrapperPSO	1500.0	2352318 ± 217667
	FastPSO	517.2 ± 51.5	1310517 ± 101113
	RapidPSO	284.5 ± 43.7	997554 ± 110503
	FilterPSO	0.0	650762 ± 108362

TABLE V. RESULTS OF SFS AND SBS

Method	Wine		WDBC		Ionosphere		Splice	
	Size	Accuracy	Size	Accuracy	Size	Accuracy	Size	Accuracy
SFS	4	91.63	5	95.26	4	80.34	6	89.98
SBS	6	98.31	21	96.84	22	83.76	50	73.44
Method	**Hillvalley**		**Gas 6**		**Musk1**			
	Size	Accuracy	Size	Accuracy	Size	Accuracy		
SFS	7	51.98	4	99.82	9	76.10		
SBS	90	54.21	8	99.12	145	76.73		

E. Further Comparisons with Traditional Methods

The results of two traditional algorithms, sequential forward selection (SFS) [25] and sequential backward selection (SBS) [26], are shown in Table V. SFS and SBS are deterministic algorithms and produce a single solution on each dataset. Since the experiments on Madelon and Isolet 5 cannot finish within two days, the results are not listed in the table. For the seven datasets in Table V, SFS selected a smaller number of features than FastPSO and RapidPSO and SBS usually selected more features, but FastPSO and RapidPSO obtained higher classification accuracies than SFS and SBS on almost all datasets.

VII. CONCLUSIONS AND FUTURE WORK

The goal of this paper is to investigate a PSO approach to combine filters and wrappers for feature selection to re-

duce the dimensionality of the data and maintain or increase the classification performance. To achieve this goal, two algorithms FastPSO and RapidPSO have been proposed by introducing wrapper evaluations into PSO-based filter algorithms, and FastPSO has more wrapper evaluations than RapidPSO. FastPSO ad RapidPSO are tested and compared with WraperPSO and FilterPSO on a number of benchmark problems of varying difficulty. The experimental results show that FastPSO ad RapidPSO reduced the dimensionality of the data and achieved the same or better classification performance than using all the available features. WrapperPSO achieved the highest classification accuracy and selected the smallest number of features, but used the longest computational time, while FilterPSO used the shortest time, but is the worst in terms of the classification performance and the subset size. FastPSO ad RapidPSO obtained similar classification performance to WrapperPSO in most cases, used a shorter time, but selected a slightly larger number of features. Meanwhile, FastPSO ad RapidPSO outperformed FilterPSO in terms of the classification performance and the number of features, but worse in terms of the computational cost. The theoretical analysis provides a lower bound of the number of wrapper evaluations for FastPSO, and the actual computational cost is higher than the lower bound, showing that FastPSO performed a much more informed search. FastPSO achieved better classification performance than RapidPSO with slightly higher computational cost and larger feature subset size. In addition, FastPSO and RapidPSO achieved higher classification performance than LFS and SBS. Although their sizes are larger than SFS, classification performance is usually more important than the number of features.

Several questions have been raised in the process of this research. Perhaps most obviously, there appears to be a limit on the classification accuracy of many datasets. This could be due to interaction between the classification algorithm and the data, or a characteristic of the dataset itself. Knowing *why* these limits seem to exist may give insight into the performance of the feature selection algorithms and the classification algorithm.

ACKNOWLEDGMENT

We would like to thank Dr Ivy Liu and Dr Mark Johnston from School of Mathematics, Statistics and Operations Research at Victoria University of Wellington for their help in the theoretical analysis of the algorithm.

REFERENCES

[1] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 4, pp. 131–156, 1997.

[2] F. Lin, D. Liang, C.-C. Yeh, and J.-C. Huang, "Novel feature selection methods to financial distress prediction," *Expert Systems with Applications*, vol. 41, no. 5, pp. 2472–2483, 2014.

[3] K. Neshatian and M. Zhang, "Improving relevance measures using genetic programming," in *European Conference on Genetic Programming (EuroGP 2012)*, ser. Lecture Notes in Computer Science, vol. 7244. Springer, 2012, pp. 97–108.

[4] B. Xue, M. Zhang, and W. N. 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.

[5] 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 (CEC'14)*, 2014, pp. 3111 – 3118.

[6] 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.

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

[8] ——, "A discrete binary version of the particle swarm algorithm," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, 1997, pp. 4104–4108.

[9] C. Shannon and W. Weaver, *The Mathematical Theory of Communication*. Urbana:The University of Illinois Press, 1949.

[10] 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.

[11] L. Cervante, B. Xue, L. Shang, and M. Zhang, "A dimension reduction approach to classification based on particle swarm optimisation and rough set theory," in *25nd Australasian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 7691. Springer, 2012, pp. 313–325.

[12] B. Xue, M. Zhang, and W. Browne, "Novel initialisation and updating mechanisms in PSO for feature selection in classification," in *Applications of Evolutionary Computation*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7835, pp. 428–438.

[13] E. Martinez, M. M. Alvarez, and V. Trevino, "Compact cancer biomarkers discovery using a swarm intelligence feature selection algorithm," *Computational Biology and Chemistry*, vol. 34, no. 4, pp. 244–250, 2010.

[14] B. Tran, B. Xue, and M. Zhang, "Overview of particle swarm optimisation for feature selection in classification," in *Simulated Evolution and Learning*, ser. Lecture Notes in Computer Science. Springer, 2014, vol. 8886, pp. 605–617.

[15] B. Xue, "Particle swarm optimisation for feature selection," Ph.D. dissertation, Victoria University of Wellington, Wellington, New Zealand, 2014.

[16] 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.

[17] 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 (CEC'08)*, 2008, pp. 2093–2098.

[18] Y. Liu, G. Wang, H. Chen, and H. Dong, "An improved particle swarm optimization for feature selection," *Journal of Bionic Engineering*, vol. 8, no. 2, pp. 191–200, 2011.

[19] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[20] L. Cervante, B. Xue, M. Zhang, and L. Shang, "Binary particle swarm optimisation for feature selection: A filter based approach," in *IEEE Congress on Evolutionary Computation (CEC'12)*, 2012, pp. 881–888.

[21] X. Wang, J. Yang, X. Teng, W. Xia, and R. Jensen, "Feature selection based on rough sets and particle swarm optimization," *Pattern Recognition Letters*, vol. 28, no. 4, pp. 459–471, 2007.

[22] W. Szpankowski, *Inclusion-Exclusion Principle*. John Wiley & Sons, Inc., 2001, pp. 49–72.

[23] K. Bache and M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[24] M. Clerc and J. Kennedy, "The particle swarm– explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[25] A. Whitney, "A direct method of nonparametric measurement selection," *IEEE Transactions on Computers*, vol. C-20, no. 9, pp. 1100–1103, 1971.

[26] T. Marill and D. Green, "On the effectiveness of receptors in recognition systems," *IEEE Transactions on Information Theory*, vol. 9, no. 1, pp. 11–17, 1963.