To appear in *Connection Science*
Vol. 00, No. 00, Month 20XX, 1–26

Investigation on Particle Swarm Optimisation for Feature Selection on High-dimensional Data: Local Search and Selection Bias

Binh Tran, Bing Xue*, Mengjie Zhang and Su Nguyen

School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Feature selection is an essential step in classification tasks with a large number of features, such as in gene expression data. Recent research has shown that particle swarm optimisation (PSO) is a promising approach to feature selection. However, it also has potential to get stuck in local optima, especially for gene selection problems with a huge search space. Therefore, we developed a PSO algorithm (PSO-LSRG) with a fast "local search" combined with a *gbest* resetting mechanism as a way to improve the performance of PSO for feature selection. Furthermore, since many existing PSO based feature selection approaches on the gene expression data have feature selection bias, i.e. no unseen test data is used, two sets of experiments on 10 gene expression datasets were designed: *with* and *without* feature selection bias. As compared to the standard PSO, the PSO with *gbest* resetting or local search only, PSO-LSRG obtained a substantial dimensionality reduction and a significant improvement for the classification performance in both sets of experiments. PSO-LSRG outperforms the other three algorithms when feature selection bias exists. When there is no feature selection bias, PSO-LSRG selects the smallest number of features in all cases, but the classification performance is slightly worse in a few cases, which may be caused by the overfitting problem. This shows that feature selection bias should be avoided when designing a feature selection algorithm to ensure its generalisation ability on unseen data.

Keywords: Feature selection; particle swarm optimisation; digh-dimensional data; classification;

1. Introduction

Feature selection is an important pre-processing step in many machine learning and data mining tasks, such as classification in which a large number of features/variables are typically included in the dataset and "the curse of dimensionality" frequently happens (H. Liu & Yu, 2005). By selecting only a small subset of relevant features and removing the redundant/irrelevant features, feature selection can reduce the dimensionality, speed up the classification process, simplify the learnt classifier, and maintain or even improve the classification performance (Guyon & Elisseeff, 2003). Feature selection is also called gene selection in microarray data in which thousands or even tens of thousands of features (i.e. genes) are involved in a dataset. Selecting key features/genes in microarray data is much more important and necessary than other problems since some classification algorithms even cannot get a visible solution due to the high-dimensional huge data space. This paper focuses mainly on feature selection for microarray gene expression

*Corresponding author. Email: Bing.Xue@ecs.vuw.ac.nz

data.

There have been a large number of feature selection approaches (Xue et al., 2013; Guyon & Elisseeff, 2003; Kohavi & John, 1997; H. Liu & Yu, 2005) in the past decades. Existing feature selection algorithms can be generally classified into filter and wrapper approaches according to the evaluation criterion (Dash & Liu, 1997; Kohavi & John, 1997). Wrapper approaches use a classification algorithm to measure the classification performance of the selected features as part of the evaluation criterion while filter approaches are independent of any classification algorithm. Wrapper approaches are generally better than filter approaches in terms of the classification performance for a particular classifier, but computationally more expensive since each evaluation involves training and testing processes of a classification algorithm (Dash & Liu, 1997; Kohavi & John, 1997).

Feature selection is a challenging task in which the number of possible solutions (feature subsets) is 2^n for a dataset with n features. A good search technique is essential to develop a feature selection approach, especially when n is large. Evolutionary computation (EC) techniques are well known for their powerful search ability (Engelbrecht, 2007). Particle swarm optimisation (PSO) (Kennedy & Eberhart, 1995; Shi & Eberhart, 1999) is an EC technique based on swarm intelligence and has shown to be able to solve problems with high-dimensionality (X. Li & Yao, 2009, 2012). PSO has been successfully used to solve feature selection tasks including on microarray data (Alba et al., 2007; Babaoglu et al., 2010; Chuang et al., 2008). Since the search space of a gene selection task is huge, which often leads to premature convergence, we recently proposed a new approach by incorporating "local search" that further tunned the good solutions obtained by PSO in a fast manner and a mechanism to reset the leader of the population (i.e. global best *gbest*) to avoid premature convergence (Tran et al., 2014a). The proposed approach improved the gene selection performance in terms of both the classification accuracy and the number of selected genes, but it was only tested on five datasets due to the page limit and had feature selection bias.

Feature selection bias is an important issue in feature selection (Ambroise & McLachlan, 2002; Ding & Peng, 2005; Singhi & Liu, 2006), which happens when the whole set of data is used during the feature selection process, i.e. no (test) data is unseen to feature selection (Kohavi & John, 1997; Singhi & Liu, 2006). Although experienced researchers (Ambroise & McLachlan, 2002; Ding & Peng, 2005; Kohavi & John, 1997; Singhi & Liu, 2006; Zhu et al., 2007) have discussed feature selection bias in detail, many existing approaches still have this issue, especially for wrapper approaches on microarray gene-expression data which usually have a small number of examples and an n-fold cross validation is needed. Since EC techniques, e.g. PSO, are stochastic approaches which need to perform multiple runs to test the performance, performing an n-fold cross validation and multiple evolutionary runs is time-consuming. This often leads to the problem of feature selection bias on microarray gene-expression data (Abedini et al., 2013; Ahmed et al., 2012; Alba et al., 2007; Babaoglu et al., 2010; Huang et al., 2007; Mishra et al., 2009; Mohamad et al., 2011, 2013; Santana et al., 2010; Shen et al., 2007; Yu et al., 2009) (or sometimes on other classification tasks (Bharathi P T, 2014; Chuang et al., 2008; Y. Li et al., 2009; Mukhopadhyay & Mandal, 2012; Oh et al., 2004)). Therefore, investigating the issue of feature selection bias on PSO for feature selection on gene data is an important task.

1.1. *Goals*

This paper aims at investigating a PSO based feature selection approach to gene expression data with thousands or tens of thousands of features, with the expectation of selecting only a small subset of features and achieving similar or significantly better classification performance than using all features. To achieve these two objectives, we extended our initial work recently published in a conference (Tran et al., 2014a) by examining the algorithms on problems with a wider range of difficulty with further discussions and analysis, and more importantly, investigating the issue of feature selection bias to compare and test the PSO based feature selection algorithms. The PSO algorithm (PSO-LSRG) with "local search" and the *gbest* resetting mechanism is compared with standard PSO, PSO with the *gbest* resetting mechanism only (PSO-RG) (Chuang et al., 2008), and PSO with "local search" only (PSO-LS) on ten different gene datasets. Specifically, we would like to answer the following questions:

- Can the proposed algorithm reduce the number of features and achieve similar or better classification performance than using all features ?
- Can the proposed algorithm outperform the standard PSO, PSO-LS and PSO-RG ?
- Can the same observation be obtained in the experiment designs with and without feature selection bias ?

1.2. *Major Contributions*

The major contributions of this paper are: (1) propose a feature selection approach to high-dimensional classification problems by developing a new local search, which can be used together with PSO to balance the global and local search to achieve better performance, and the local search was designed in a way to speed up the evaluation process to avoid high computational cost since the majority of the computational cost in wrapper feature selection is cost by the evaluations, (2) show the influence of the feature selection bias by testing the performance of the algorithms in the situations with and without feature selection bias.

1.3. *Organisation*

The remainder of the paper is organised as follows. Section 2 provides an overall view of feature selection methods and briefly introduces the standard PSO algorithm. Section 3 describes our PSO based feature selection approach. Sections 4 and 5 presents experimental results under the situations with and without feature selection bias. Section 6 provides conclusions and future work.

2. Background

This section reviews the background of PSO and related work on feature selection, including traditional (non-EC) methods and EC based methods, particularly PSO for feature selection on high-dimensional data. Since there have been a large number of papers on feature selection, which is impossible to fully covered in this paper, we focus on the most closely related work and readers are referred to recent surveys on feature selection for other important work in the field (Chandrashekar & Sahin, 2014; De La Iglesia, 2013; Hancer et al., 2015; Kundu & Mitra, 2015; Liu et al., 2010; Rauber et al., 2015; Tran et

al., 2014b; Vergara & Estévez, 2014; Xue et al., 2015; Zhai et al., 2014).

2.1. *Particle Swarm Optimisation*

PSO was developed by Kennedy and Eberhart (Kennedy & Eberhart, 1995) in 1995. It is inspired by social behaviours found in birds flocking or fish schooling. In PSO, a swarm consists of many individuals called particles communicating together to search for optimal solutions by moving in the search space. Each particle has a position and a velocity. The position presents a candidate solution of the problem and is usually an n-dimension vector of numerical values. The velocity which is also a numerical vector of n-dimension indicates the speed and direction that the particle should move in each direction. It is updated based on the personal best (*pbest*) position that the particle has been explored so far and the global best (*gbest*) position obtained by the whole population. Equations (1) and (2) are used to update the velocity and position of each particle.

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

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

where v_{id}^{t} and x_{id}^{t} are the velocity and the position of particle i in dimension d at the tth iteration, respectively. p_{gd} and g_{gd} are *pbest* and *gbest* values in dimension d. c_1 and c_2 are acceleration constants, and r_1 and r_2 are random values. w is the inertia weight used to control the impact of the last velocity on the current velocity. The velocity values are usually limited by a predefined maximum velocity, v_{max}, to the range $[-v_{max}, v_{max}]$.

2.2. *Traditional Methods for Feature Selection*

Since exhaustive search for the best feature subset is impractical in most situations, heuristic approaches have been applied to feature selection (Dash & Liu, 1997). Two typical greedy search methods are sequential forward selection (SFS) (Wang et al., 2014) and sequential backward selection (SBS) (Marill & Green, 1963). SFS or SBS gradually adds or removes features until the classification accuracy is not improved. In these algorithms, the search space is reduced from $O(2^N)$ to $O(N^2)$. However, both methods suffer from the so-called "nesting effect" because a feature which is selected or removed cannot be removed or selected in later stages. "plus-l-take-away-r" (Stearns, 1976) compromises these two approaches by applying SFS l times and then SBS r times. However, it is hard to determine appropriate values for l and r. To avoid this, sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS) were proposed in (Pudil et al., 1994). Both floating search methods are claimed to be better than the static sequential methods.

Linear forward selection (LFS) (Gutlein et al., 2009) also followed the sequential strategy but used a predefined number of features k to consider only k first ranked features in each forward selection step. Results showed that LFS was faster, selected smaller subsets and increased the accuracy in comparison to standard forward selection. These greedy methods have shown good results in many real-world applications. However, they usually face the problem of stagnation in local optima and intractable computation time when the number of features is very large, e.g. thousands of features. Therefore, it is necessary to have a more efficient global search approach.

2.3. *EC Techniques for Feature Selection*

Evolutionary computation (EC) is a group of population based techniques that tend to explore more distinct regions of the search space (Cortez, 2014). EC techniques have been recently applied to solve feature selection problems.

Among EC techniques, genetic algorithm (GA) is probably the first popular EC technique that has been applied in feature selection. Guided by Darwinian evolution principles, a GA starts with a population of candidate solutions, represented as chromosomes, and evolved better solutions by using genetic operators like crossover and mutation. Y. Li et al. (2009) proposed a multiple-population based GA for feature selection, where adaptive genetic operators were developed to improve its performance. Tested with different filter and wrapper measures, the proposed scheme was shown to be effective for feature selection. A GA and simulated annealing were combined together to solve feature selection problems in (Manimala et al., 2011), where the parameters of a support vector machine (SVM) were also optimised during the feature selection process.

GP can evolve computer programs to generate solutions. Each program is a tree consisting of inner nodes which are usually arithmetic operators, and leaf nodes which are either constants or variables. When using GP for feature selection, the variables are chosen from the original features. Selected features are the ones used as leaf nodes of a GP tree. GP has been used in both filter feature selection methods (Neshatian & Zhang, 2009, 2011) and wrapper feature selection methods (Davis et al., 2006). Neshatian & Zhang (2009) proposed a wrapper feature selection approach, where a variation of Naive Bayes (NB) classification algorithm was used for classification. Feature ranking for feature selection using GP was achieved in (Neshatian & Zhang, 2011), where features were ranked according to the frequency of appearance in the best GP individuals. Feature selection was achieved by using only the top ranked features for classification.

ACO is inspired by the special communication system using pheromone between real ants about favourable paths to food. The shortest path will be the one that has most pheromone. O'Boyle et al. (2008) proposed to use ACO to simultaneously select features and optimise the parameters of a SVM, where a weighting method was also proposed to determine the probability of an ant selecting a particular feature. Santana et al. (2010) compared the performance of ACO and GA based feature selection methods for ensemble classifiers. The results showed that ACO performed better when the number of individual classifiers is small while the GA performed better when this number is big. Chen et al. (2012) proposed a new representation scheme to reduce the size of the search space (i.e. graph), where each feature/node is only connected to the next nodes using two edges showing "selected" or "not selected". This representation scheme significantly reduced the total number of edges that the ACO algorithm had to traverse.

The key idea of PSO is learning from neighbours' experience through communication the global best (*gbest*) information, and learning from individual's own experience through *pbest*. Based on *gbest* and *pbest*, many improved PSO algorithms have been proposed to solve feature selection tasks. Geometric PSO algorithms were proposed for feature selection (Talbi et al., 2008), where the current position, *pbest*, and *gbest* of a particle were used as three parents in a three-parent mask-based crossover operator to create a new position for the particle instead of using the position updating equation. Xue et al. (2014) developed a new initialisation strategy to mimic the conventional SFS and SBS algorithms, which showed that good initialisation can significantly increase the performance of PSO for feature selection.

2.4. *PSO for Feature Selection on High-dimensional Datasets*

Recent research has shown that PSO is a promising approach to feature selection. However, it also easily gets stuck in local optima, especially for gene selection problems with a huge search space. Therefore, many strategies have been proposed to solve this problem.

One strategy was to resetting the *gbest* when its fitness was identical after a predefined number of iterations. Yang et al. (2008) applied a Boolean operator 'and(.)' which would 'and' each bit of the *pbest* of all particles to create a new binary string. This new binary string would replace the current *gbest*. Results illustrated that the proposed method usually achieved higher classification accuracy with fewer features than GA and PSO. Yang et al. (2008) showed that PSO had the potential to be superior to other compared methods. However, proposed PSOs were not compared with other variations of PSO. In (Chuang et al., 2008), *gbest* was reset to zero, which meant no feature was selected. Results illustrated that (Chuang et al., 2008) effectively reduced the number of needed features and achieved better classification accuracy than (Yang et al., 2008) in most cases.

Combining both wrapper and filter approach to improve PSO performance in feature selection is another popular strategy. In a wrapper method using PSO with ID3 (X. Liu & Shang, 2013), mutual information was used to score the relevance and redundancy of a feature within a feature subset. This score was then added to the velocity of the particle to give features with a higher score a higher chance to be selected. Results on ten UCI datasets with tens of features showed that the proposed method helped improve ID3 classification performance. However, applying this method on high-dimensional datasets may require a very high computation time and a large memory.

Two-stage approaches are also very popular, where the first stage is to reduce the number of features before applying PSO in the second stage. In (Mohamad et al., 2011), a gain ratio technique was used to choose 500 top-ranked genes/features. A speed concept was introduced to update the positions of the particles instead of velocity to increase the probability of not choosing a feature to reduce the number of features. In this way, the proposed algorithm obtained much smaller feature subsets than in (Chuang et al., 2008) and the other compared methods. However, this method also reduced the accuracy in cases which might require a relatively large number of features. In (Banka & Dara, 2015), a quartile based preprocessing was applied to eliminate irrelevant features in the first stage. Remaining features are further selected by a binary PSO algorithm which uses hamming distance as a proximity measure in the velocity updating formula. Results showed that the method outperformed other compared methods. However, its application on larger gene expression data with a larger number of classes may be hindered by its memory requirement.

In summary, although PSO or other EC techniques have shown some success on solving high-dimensional or large-scale feature selection tasks, there are still some potential limitations, such as time-consuming and stagnation into local optima. Meanwhile, since many high-dimensional datasets, especially gene expression data, typically have a small number of instances, n-fold cross validation is needed. How to perform n-fold cross validation in multiple EC runs is hard to design and computationally expensive. This often leads to the issue of feature selection bias (Abedini et al., 2013; Ahmed et al., 2012; Alba et al., 2007; Babaoglu et al., 2010; Huang et al., 2007; Mishra et al., 2009; Mohamad et al., 2011, 2013; Santana et al., 2010; Shen et al., 2007; Yu et al., 2009). Therefore, investigating an effective and efficient feature selection algorithm for high-dimensional data and the influence of feature selection bias are still open issues.

Algorithm 1: The Pseudo code of PSO-LSRG

```
begin
    initialise the position and velocity of each particle;
    while Maximum Iterations or the stopping criterion is not met do
        Collect the feature subsets selected by each particle;
        Use 1NN with LOOCV to evaluate the classification error rate of each feature subset ; // *as the fitness
        value of each particle*
        for i=1 to P do
            if fitness of particle i is better than that of pbest then
                Update the pbest of particle i;
                Perform "local search" on pbest;
                Update pbest if a better solution is found by "local search";
            end
        end
        Update the gbest of the swarm;
        if gbest is not improved for m iterations then
            Reset all gbest to a vector of all "O"s;
        end
        for i=1 to Population Size do
            for d=1 to Dimensionality do
                Update the velocity of particle i according to Equation (1);
                Update the position of particle i according to Equation (2);
            end
        end
    end
    Return the position of gbest (the selected feature subset);
    Return the classification performance of gbest;
end
```

3. Proposed Approach

The overall structure of PSO-LSRG can be seen in Figure 1 in which the two added steps to standard PSO are highlighted: a "local search" on *pbest* and a *gbest* resetting mechanism (Tran et al., 2014a). The right half of the figure gives details about the "local search" on *pbest*. The *gbest* resetting mechanism follows the literature (Chuang et al., 2008) to reset *gbest* to a solution in which no feature is selected. More details can be seen from the pesudo-code shown in Algorithm 1. The "local search" will be described in the remainder of this section.

In PSO-LSRG, the position of each particle represents a solution, i.e. a feature subset. The representation is a string of real-numbers with the length equivalent to the total number of features in the dataset. Each bit in the string corresponds to one feature and the possible value is in the range of [0, 1], which shows the probability of the feature being selected. A threshold θ is used to determine the selection of the feature. A feature is selected if its corresponding position value is larger than θ or not selected otherwise.

PSO-LSRG is a wrapper method which uses the classification performance of the k-nearest neighbour (KNN) as the fitness value, which can be seen in Equation (3).

$$Fitness = \frac{Number\ of\ Correctly\ Classified\ Instances}{Total\ Number\ of\ Instances} \tag{3}$$

3.1. *"Local Search" on pbest*

The "local search" is used as a complimentary search of the PSO based global search, which aims to exploit the search space around the obtained good solutions, *pbest*, to find better ones. As can be seen from Algorithm 1, when the PSO algorithm finds a better *pbest* for a particle, e.g. *pbest* is updated, the "local search" is applied to refine the updated/new *pbest*. If a better solution is found by the "local search", *pbest* will be

further updated as the new solution. Otherwise, *pbest* remains the same.

To perform "local search", the *pbest*, which is a real-number string, is converted into a binary string to show a feature subset in which "1"s represent corresponding selected features. The "local search" is a flipping mechanism, which flips the values in the binary string from "0" to "1" or from "1" to "0", i.e. a feature from being "not selected" to "selected" or from being "selected" to "not selected". The flipping is only performed on a small proportion of the binary string to ensure the "local search" to focus on the near surroundings of the *pbest*. Since this work targets on high-dimensional data with thousands or tens of thousands of features, only 2% of the bits/dimensions in the binary string of *pbest* are flipped in each step of the local search. After one flipping step, if the flipped *pbest* is better than the current *pbest*, *pbest* is updated. The flipping is repeated for a predefined number of times.

3.2. *Fast Fitness Evaluation in "Local Search"*

The introduction of the "local search" brings a number of extra fitness evaluations. In wrapper approaches, each evaluation requires a process of training and testing a classifier to get the classification performance as the fitness value. Therefore, most of the computational cost is spent on the fitness evaluations. In PSO-LSRG, each flipping step in the "local search" brings an extra calculation of the fitness value (i.e. the classification error rate), which may cause a significant increase in the computational cost.

To address this issue, a fast fitness evaluation strategy is proposed for the "local search", which is designed by considering the characteristics of KNN classification algorithm and the "local search". In KNN, the class label of an unseen instance is determined according to its distance to the training instances. The overall distance between the unseen instance and a training instance can be calculated based on the sum of their distances in each feature. Therefore, when adding or removing features, the overall distance value can be calculated/updated by plus or minus their distances in these features only. In the "local search", each flipping step changes only a small percentage (2%) of the total dimensions/features while 98% of them remain the same. Instead of re-calculating the distance between an unseen instance and a training instance, the fast fitness evaluation strategy only re-calculates the 2% of the total features. To achieve this, at the beginning of each "local search" run, all the distances between the unseen instance and the training instances are calculated based on the features selected by the given *pbest* and stored in a square matrix (distance[i][j]). Since this matrix is symmetric, $\frac{m(m+1)}{2}$ distances are calculated with m instances. When a flipping step is performed, using the distances stored in the matrix can speed up the computation of finding the nearest neighbours of a certain instance by recalculating only 2% of the dimensions.

3.3. *Illustration of the "Local Search"*

Figure 2 illustrates the process of the "local search" and the fast fitness evaluation process. Figure 2(a) shows an example of the flipping procedure, where *pbest* is converted to a binary string. Based on the randomly chosen flipping dimensions, e.g. (2, 4, 7, 8, 10), the current *pbest* can be flipped to obtain a new *pbest* position (flipped *pbest*).

Figure 2(b) shows an example of the fast fitness evaluation process between two instances: $Instance_i$ and $Instance_j$, where the overall distance is the sum of the squared distance between $Instance_i$ and $Instance_j$ in all dimensions/features. Suppose the overall distance between $Instance_i$ and $Instance_j$ according to the current *pbest* ($distance[i][j]$)

is 258, their overall distance according to the flipped *pbest* can be re-calculated by changing values in the five flipped dimensions. Features 4 and 7 are flipped from selected to not selected, so their distances will be subtracted from 258. Meanwhile, Features 2, 8 and 10 are flipped from not selected to selected and their distances will be added to 258. The new distance therefore is 258 - 16 - 64 + 25 + 9 +1 = 213. After calculating the overall distance with all the training instances, the KNN algorithm can quickly find the new nearest neighbour to determine the class label of the unseen instance. As a result, a significant amount of time can be saved in the "local search" process.

4. Experiment I

In this section, we first test the performance of the four algorithms in a commonly used way, where feature selection is performed on the whole dataset, i.e. with feature selection bias. The results can show the "training" performance and the search ability of the algorithms. The results will also be used to compare with that of without feature selection bias in the next section (Experiment II) to test the influence produced by feature selection bias.

4.1. *Design of Experiments*

To examine the performance of PSO-LSRG, it is compared with three other PSO algorithms for feature selection, which are the standard PSO (PSO), PSO with *gbest* resetting only (PSO-RG) (Chuang et al., 2008), and PSO with "local search" on *pbest* only (PSO-LS). Since previous research have shown that PSO based approaches can achieve better performance than traditional methods (Xue et al., 2013), this paper does not compare the PSO algorithms with traditional methods to save space. Ten gene expression datasets downloaded from http://gems-system.org/ are used to test the performance of the algorithms, where Table 1 shows a brief overview of these datasets. KNN (K=1, i.e. 1NN) with leave one out cross validation (LOOCV) is used to calculate the classification accuracy of the selected features, since the number of instances is very small compared with the number of features.

All the four algorithms share the same parameter settings for the PSO algorithm, which are $c1 = c2 = 2.0$, w linearly decreases from 0.9 to 0.4 (Shi & Eberhart, 1999). The swarm includes 100 particles and the maximum number of iterations is 70. Fully connected topology is used and the maximum velocity is 6.0. The threshold $\theta = 0.6$ (Xue et al., 2013) is used to determine whether a feature is selected.

In PSO-RG and PSO-LSRG, if *gbest* is not improved for three iterations (i.e. m=3), it is reset to all 0, which is the same as in (Chuang et al., 2008) for comparison purposes. In PSO-LS and PSO-LSRG, 100 times of flipping will be performed to find a better *pbest* in each "local search", where in each time, 2% of the dimensions will be flipped to create a new candidate solution.

The experiment of each algorithm on each dataset is conducted for 40 independent runs with different random seeds. A statistical significance test, Wilcoxon test, is performed to compare the classification performance of different algorithms, where the significance level is set as 0.05.

4.2. *Results and Discussions*

Table 3 shows the experimental results of the four PSO algorithms: PSO, PSO-RG, PSO-LS, and PSO-LSRG. "All" means all the available features are used for classification. "Ave-Size" shows the average number of features selected by each method over the 40 runs. "Best", "Mean" and "StdDev" show the best, the average, and the standard deviation of the classification accuracy achieved by each method in the 40 independent runs. The smallest average number of features and the best classification accuracy in each dataset are bold. The last three columns show the results of the significance tests comparing PSO-RG, PSO-LS, or PSO-LSRG with the other corresponding algorithms. For example, the "+" ("–") in the "T_{RG}" column means the corresponding method achieves significantly better (worse) classification performance than PSO-RG. "=" means they are similar. Similarly, "T_{LS}" or "T_{LSRG}" are the results of the Wilcoxon tests comparing the classification performance achieved by other methods and PSO-LS or PSO-LSRG, respectively. In general, the more "–"s, the better PSO-RG, PSO-LS, or PSO-LSRG is.

4.2.1. Results of PSO and PSO-RG

As can be seen from Table 3, PSO removed around half of the original features on all the ten datasets. With only the selected features, 1NN with LOOCV improved the classification performance over using all the original features in all cases. Using the feature subsets selected by PSO, the classification accuracy was increased roughly 20% on two datasets (9 Tumors and Brain Tumors 2), 10% on three datasets, and 7% on the remaining datasets. These results show that PSO can be a suitable search mechanism to solve feature selection tasks in high-dimensional classification problems.

According to Table 3, the symbols in the T_{RG} column are all "–", which show that the classification accuracy of feature subsets selected by PSO-RG is significantly better than that of using all features and those of PSO on all the ten datasets. Meanwhile, the average number of features selected by PSO-RG is always smaller than that of PSO. PSO-RG selected about 15% of the original features on four datasets. On the SRBCT and Leukemia 2 datasets, this number even further reduced to just 9%. The possible reason for this is that resetting *gbest* to 0 attracts all other particles moving toward this direction, resulting to smaller subsets.

4.2.2. Effect of "Local Search" on pbest (PSO-LS)

As can be seen from Table 3, in terms of the number of features, PSO-LS selected approximately 30% of all the original features. The average size in PSO-LS is smaller than PSO in all cases, and smaller than PSO-RG on three datasets. Meanwhile, PSO-LS achieved significantly better classification performance than using all features and the features selected by PSO on all the ten datasets. The classification accuracy of PSO-LS is significantly higher than PSO-RG on five datasets and similar on the other five datasets, which means that PSO-LS is similar or significantly better than PSO-RG in terms of the classification performance.

The overall results show that the "local search" on *pbest* helps the PSO algorithm better exploit around the obtained good solutions, which further lead the particles to search for solutions with better classification performance. Since the "local search" considers only the classification performance not the number of features and *gbest* resetting in PSO-RG facilitates particles search for smaller feature subsets, the number of features selected by PSO-LS is often larger than that of PSO-RG. Therefore, PSO-LSRG is expected to take the advantages of both the "local search" and the *gbest* resetting to improve the

classification performance and reduce the number of features.

4.2.3. Results of PSO-LSRG

According to Table 3, the average number of features selected by PSO-LSRG is much smaller than the total number of features, which is only 7% of the total number of features on SRBCT, 23% on Lung Cancer, and around 15% on other four datasets, i.e. DLBCL, Leukemia 1, Brain Tumor 1, and Brain Tumor 2. With only the small selected feature subsets, the classification performance was significantly improved over using all features in all cases.

In general, PSO-LSRG achieved the highest classification accuracies on eight out of the ten datasets and selected the smallest number of features on six datasets. Specifically, PSO-LSRG outperformed PSO in terms of both the classification performance and the number of features. PSO-LSRG achieved significantly better classification performance than PSO-RG on eight datasets and similar on the other two datasets. Meanwhile, the average number of features in PSO-LSRG is smaller than PSO-RG on eight datasets. Comparing PSO-LSRG with PSO-LS, PSO-LSRG achieved similar or significantly better classification performance than PSO-LS and selected a much smaller number of features on eight of the ten datasets. On the two remaining datasets, i.e. 9 Tumors and 11 Tumors, PSO-LSRG achieved slightly worse performance than PSO-LS. From Table 1, it can be observed that these two datasets include a relatively large number of classes and features, but a small number of instances, where the tasks are different from others. They may need longer time for the particles to search according to *gbest*, where resetting *gbest* to zero when it is not changed for three iterations stops particles further exploiting their current *gbest*. From this point of view, a better mechanism to evaluate the stagnation status of a PSO swarm is needed to improve the performance.

In conclusion, PSO-LSRG has synthesized the strength of both *gbest* resetting technique and local search mechanism to achieve the best performance in terms of classification performance and the number of selected features. The former enables PSO to explore toward smaller subsets in the search space when the swarm is close to a stagnation status. However, it also hinders particles from further exploiting the surrounding solutions when resetting *gbest* to null subset. The latter technique compromises this situation by giving PSO a chance to search for better solutions using a kind of "mutation" technique on the current good solutions (*pbest*). Therefore, combing these two techniques in PSO is a better way to balance the exploration and exploitation processes.

4.3. *Evolutionary Training Process*

To better observe the searching behaviour of the four PSO algorithms, Figure 3 shows the average classification accuracy of *gbest* in PSO, PSO-RG, PSO-LS, and PSO-LSRG over the 40 independent runs. The ten graphs correspond to the ten datasets used in the experiments. In each graph, the horizontal coordinate shows the number of iterations and the vertical coordinate shows the average classification accuracy of *gbest*.

As can be seen from Figure 3, by performing the "local search", PSO-LS can quickly find better solutions than PSO and PSO-RG at the beginning of the searching process. By applying the *gbest* resetting mechanism, after the first iteration, PSO-RG generally made more improvement than PSO and PSO-LS over the searching process. PSO-LSRG combining the "local search" and *gbest* resetting mechanism found better solutions at the beginning of the search and made significant improvement over the searching process, which lead to better solutions than the other three algorithms. The observations in Figure

3 are consistent with that from Table 3.

4.4. *Computational Cost*

Since the "local search" introduces extra fitness evaluations which requires longer computational time, the fast fitness evaluation mechanism was proposed to speed up the process. To see the effect of this fast mechanism on the computational cost of the "local search", Table 4 summaries the average CPU time used by each method in the 40 independent runs on the ten datasets, where the numbers are expressed in seconds.

All the four algorithms share the same number of particles and iterations, so they have the same number of fitness evaluations in the PSO search process, which is 7000 (100*70). However, since PSO-LS and PSO-LSRG involves a "local search" process, they involve a much larger number of evaluations than PSO and PSO-RG. According to Table 4, PSO-RG used the shortest time in almost all cases, where the main reason is that the number of features selected by PSO-RG is usually smaller than others. This confirms the significant influence of the size of feature subsets on the computational time in wrapper feature selection approaches.

In PSO-LS and PSO-LSRG, the "local search" adds a large number of fitness evaluations. Every time a particle reaches a new *pbest*, the "local search" repeats 100 steps of flipping to search for a better *pbest*. In other words, it will call 100 fitness evaluations. If in one PSO iteration only one-third of the population, which is about 30 particles in this experiment, can reach new *pbest*, PSO-LS and PSO-LSRG will have extra 3000 (100*30) evaluations than PSO and PSO-RG in each iteration. In total, PSO-LS and PSO-LSRG may have extra 210 000 (3000*70) fitness evaluation, which is about 3000 times more than PSO and PSO-RG. However, by reaching solutions with smaller subsets and the use of the fast fitness evaluation in "local search", the computational time of PSO-LS and PSO-LSRG is just about twice of PSO. This new strategy successfully reduces the running time of 1NN classifier. The quick evaluation time and selecting smaller subsets enable PSO-LS and PSO-LSRG to achieve better results within a reasonable time.

Note that the algorithms in this paper use 100 particles while they used only 30 particles in (Tran et al., 2014a), but since the 1NN algorithm was re-implemented to be faster in this paper, using 100 particles did not significantly increase the computational time. Meanwhile, since the newly implemented 1NN consumed much less time in each evaluation, the influence of feature subsets size in the efficiency here is not as significant as in (Tran et al., 2014a). Importantly, the results in Table 3 are better than the results in (Tran et al., 2014a) in terms of both the classification performance and the number of features, which shows that increasing the population size can increase the performance of the PSO based feature selection algorithms on high-dimensional classification tasks and the traditional setting of 30 particles is not necessarily good for such problems.

4.5. *Selected Informative Features/Genes*

Since PSO is a stochastic approach, the individual features selected from different independent runs could be different, even for the feature subsets including the same number of features and achieving the same classification performance. Therefore, to give an idea of which features/genes are important, we count the frequency of features being selected in the final feature subsets of multiple PSO-LSRG runs. The more frequent a feature being selected, the more informative it is, since only the most useful or informative features should be selected by the PSO-LSRG algorithm.

Since the ten datasets used in the experiments include thousands of features, it is almost impossible to present the frequency of all features on each dataset. Table 6 shows the ten most frequently selected features/genes on each dataset. Since the GEMS database does not provide the name of features in the data files (except for the Lung dataset), we presented the index of the features, where the index of the features are listed in a descending order by their frequency. It would be much more interesting to discover the biological finding of these genes, but since the original meanings of the features/genes in GEMS are not given, it is impossible to perform such research. In the future, we intend to collaborate with researchers from biology to deep analyse the biological finding of the selected genes.

4.6. *Further Discussions*

In this section, the re-substitution estimator was used to evaluate the performance of the feature selection algorithms, which is the same as in (Chuang et al., 2008) and many other existing papers (Abedini et al., 2013; Ahmed et al., 2012; Alba et al., 2007; Babaoglu et al., 2010; Huang et al., 2007; Mishra et al., 2009; Mohamad et al., 2011, 2013; Santana et al., 2010; Shen et al., 2007; Yu et al., 2009). The re-substitution estimator, in other words, means the whole dataset is used during the evolutionary feature selection process (as shown in Figure 4(a)). There is no separate unseen data to test the generality of the selected features. According to Ambroise & McLachlan (2002), there is a feature selection bias issue here, so one cannot claim that the selected features can be used for future unseen data.

Feature selection bias typically happens when the dataset includes only a small number of instances, especially on the gene expression data, where n-fold cross validation (10-CV) or LOOCV is needed. Figure 4 compares the structures experiments with and without feature selection bias. It can be seen that with selection bias, the algorithm reports the classification performance of the (inner) cross validation loop and "such results are optimistically biased and are a subtle means of training on the test set " (Kohavi & John, 1997). Therefore, the conclusions drawn from the re-substitution estimator with selection bias may be different from that without bias. This, however, has not been seriously investigated in EC for gene selection.

5. Experiment II

In this section, the second set of experiments have been conducted, where the feature selection bias issue is removed.

5.1. *Performance Evaluation*

To avoid feature selection bias and compare the performance of the algorithms with and without bias, the second set of experiments without feature selection bias have been conducted. Feature selection without bias often has a much more complicated structure, where two loops (an inner loop and an outer loop (Kohavi & John, 1997)) of cross validation may be involved. Figure 4(b) shows an example of the outer loop cross validation using 10-CV as an example, where each evaluation in the "Feature Selection on Training set " process often has an inner loop of cross validation to calculate the classification performance.

In the experiment II, an outer loop of 10-CV is used to measure the classification performance of all the methods as shown in Figure 4(b), where the test set is independent of the feature selection process. Figure 5 further illustrates the details in one of the 10 processes in 10-CV, where two stages are involved. In the first stage, only the training set, i.e. 9 folds, are used, where the details of "PSO for Feature Selection" can be shown by Figure 1 in PSO-LSRG. In the second stage, both the training set and test set are transformed by keeping only the selected features. A classification algorithm is trained on the transformed training set and the classification performance on the test set is reported as the final solution.

The datasets and parameter settings in Experiment II are the same as in Experiment I. All the algorithms have been performed for 40 independent runs, where each run includes 10 (fold) repetitions of PSO for feature selection for 10-CV. Therefore, 400 (40*10) runs of PSO have been performed in each method on each dataset.

5.2. *Results on Test Set*

Table 5 shows the experimental results of the four algorithms without feature selection bias. According to the results, PSO selected only about 47% of the available features and still achieved significantly better classification performance than using all features on seven out of the ten datasets and similar performance on the other three datasets. This again confirms the ability of PSO in solving feature selection tasks on high-dimensional classification tasks.

PSO-RG achieved significantly better or similar classification performance than using all features on eight out of the ten datasets. PSO-RG further reduced the number of features selected by PSO on all datasets, but its classification performance is worse on three datasets. PSO-LS substantially reduced the number of features to roughly a quarter on all the datasets, and achieved similar or significantly better classification performance than using all features on nine out of the ten datasets. PSO-LS selected a smaller number of features than PSO in all cases and achieved similar or significantly better classification performance than PSO on nine datasets. The performance of PSO-LS is worse than PSO-RG in terms of the number of features, but slightly better in terms of the classification performance. This is expected since PSO-LS focuses more on increasing the classification accuracy and PSO-RG focuses more on reducing the number of features. PSO-LSRG shows the best performance in terms of the number of features. For example, PSO-LSRG selected on average 15 features from the 2308 features on the SRBCT dataset, but significantly improved the classification performance over using all features.

On four datasets, Leukemia 1, Brain Tumor 1, Leukemia 2, and 11 Tumors, although the best accuracy PSO-RG are higher than using all features, the average accuracy is lower on two datasets and equal on the other two. A similar pattern can be seen in PSO-LSRG. PSO-LSRG also has the worst performance on these four datasets. On the other hand, PSO-LS still worked well on three out of the four datasets. The reason of this performance deterioration may come from the *gbest* resetting technique. Resetting *gbest* to zero when *gbest* is not improved for *three* iterations may not be an appropriate choice for these datasets. Again, a better mechanism should be used to determine an appropriate time to apply this technique as we have discussed in Section 4.2.3. In addition, resetting *gbest* to zero enables swarm to explore search space towards smaller subsets. However, when flying towards zero in all dimensions, particles may loose their already learnt knowledge about different features. Therefore, a more informative *gbest* resetting

technique is needed to improve the performance, which will be investigated in the future.

5.3. *Comparisons Between With and Without Feature Selection Bias*

The results in Table 3 have feature selection bias and the results in Table 5 do not have feature selection bias. Comparing Table 3 with Table 5, it can be seen that without feature selection bias, all the algorithms obtained lower test accuracy than with feature selection bias. More importantly, the comparison shows that the patterns found from the results with feature selection bias cannot be generalised on those without feature selection bias. This suggests that a feature selection method that was tested with feature selection bias might not be a good feature selection approach in practice.

Although feature selection bias should be avoided in most cases, there still can be situations where feature selection bias does not lead to misleading observations/conclusions. For example, in some engineering problems, the goal of feature selection is to find factors/features that significantly influence the model performance. There is no unseen data to be tested on and the whole set of data (with feature selection) can be safely used to find such important features. Feature selection bias is also not a problem in some biological analysis tasks in which feature selection is used to find key genes or proteins without the need of testing them on unseen data.

Note that although the classification accuracies in Table 5 are not as high as those in Table 3, the proposed algorithms still can be successfully used for feature selection in most cases to substantially reduce the number of features and achieve similar or even better classification performance than using all features.

6. Conclusions and Future Work

This paper investigated a PSO approach to feature selection on gene expression data with thousands or tens of thousands of features in the situations with and without feature selection bias. The goal was achieved by developing a PSO algorithm named PSO-LSRG with an efficient "local search" on *pbest* and a *gbest* resetting mechanism. PSO-LSRG was tested and compared with standard PSO, PSO with *gbest* resetting mechanism only (PSO-RG), and PSO with "local search" only (PSO-LS) on ten different gene expression datasets. Two sets of experiments were conducted, where the first set had feature selection bias using the whole set of data during the feature selection process while the second set used an outer 10-fold cross validation loop to avoid feature selection bias.

The results from the first set of experiments show that PSO can be successfully used to reduce the number of features and increase the classification performance. PSO-RG further reduced the number of features and increased the classification performance since resetting *gbest* to zeros leads the swarm searching for solutions with smaller sizes. PSO-LS with the "local search" further increased the classification performance. PSO-LSRG can utilise the advantages of the two mechanisms to further reduce the number of features and increase the classification performance in most cases. Although PSO-LS and PSO-LSRG using the "local search" mechanism have a much larger number of fitness evaluations than PSO and PSO-RG, their computational time is not as much longer as they should due mainly to the fast fitness evaluation procedure. The first set of experiments have feature selection bias and the patterns observed were slightly changed when feature selection bias was removed in the second set of experiments. PSO-LSRG still performed better than other algorithms in terms of the number of features, but not always better in terms of the classification performance possibly due to the overfitting problem.

Since the gene experssion data used in this work does not include the biological meaning of each feature/gene, the biological finding of the selected features was not analysed. In the future, we intend to collaborate with researchers from biology to obtain data with biological meanings clearly labelled and peform deep analysis on the biological finding of the selected genes. The number of features selected here is over one hundred, which may cost a large amount of time and money in real-world gene analysis. Therefore, how to further reduce the number of features without reducing the classification performance still needs investigation. We will also work on biomarker detection problems to analysis the selected individual features, which requires domain knowledge from experts. Furthermore, researchers have recently made big achievement on evolutionary computation for large-scale function optimisation (Omidvar et al., 2014; Cheng & Jin, 2015), where the tasks share some similarities and also have some differences from feature selection. We also intend to adapt ideas from evolutionary large-scale function optimisation to investigate promising methods for feature selection in the future.

References

Abedini, M., Kirley, M., & Chiong, R. (2013). Incorporating feature ranking and evolutionary methods for the classification of high-dimensional dna microarray gene expression data. *Australasian Medical Journal*, *6*(5), 272–279.

Ahmed, S., Zhang, M., & Peng, L. (2012). Genetic programming for biomarker detection in mass spectrometry data. In *Ai 2012: Advances in artificial intelligence* (Vol. 7691, pp. 266–278). Springer Berlin Heidelberg.

Alba, E., Garcia-Nieto, J., & Jourdan, L. (2007). Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms. In *Ieee congress on evolutionary computation (cec'07)* (pp. 284–290).

Ambroise, C., & McLachlan, G. J. (2002). Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, *99*(10), 6562–6566.

Babaoglu, I., Findik, O., & Ulker, E. (2010). A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine. *Expert Systems with Applications*, *37*(4), 3177 - 3183.

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

Bharathi P T, P. S. (2014). Differential evolution and genetic algorithm based feature subset selection for recognition of river ice type. *Journal of Theoretical and Applied Information Technology*, *7*(1), 254–262.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28.

Chen, B., Chen, L., & Chen, Y. (2012). Efficient ant colony optimization for image feature selection. *Signal Processing*, *93*, 1566–1576.

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

Chuang, L. Y., Chang, H. W., Tu, C. J., & Yang, C. H. (2008). Improved binary PSO for feature selection using gene expression data. *Computational Biology and Chemistry*, *32*(29), 29– 38.

Cortez, P. (2014). *Modern optimization with r*. Springer.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*(4), 131–156.

Davis, R. A., Charlton, A. J., Oehlschlager, S., & Wilson, J. C. (2006). Novel feature selection method for genetic programming using metabolomic 1h NMR data. *Chemometrics and Intelligent Laboratory Systems*, *81*(1), 50–59.

De La Iglesia, B. (2013). Evolutionary computation for feature selection in classification problems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *3*(6), 381–407.

Ding, C., & Peng, H. (2005). Minimum redundancy feature selection from microarray gene expression data. *Journal of Bioinformatics and Computational Biology*, *03*(02), 185–205.

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction (2. ed.)*. Wiley.

Gutlein, M., Frank, E., Hall, M., & Karwath, A. (2009). Large-scale attribute selection using wrappers. In *Ieee symposium on computational intelligence and data mining (cidm '09)* (pp. 332–339). IEEE.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *The Journal of Machine Learning Research*, *3*, 1157-1182.

Hancer, E., Xue, B., Karaboga, D., & Zhang, M. (2015). A binary {ABC} algorithm based on advanced similarity scheme for feature selection. *Applied Soft Computing*, *36*, 334 – 348.

Huang, J., Cai, Y., & Xu, X. (2007). A hybrid genetic algorithm for feature selection wrapper based on mutual information. *Pattern Recognition Letters*, *28*(13), 1825 - 1844.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Ieee international conference on neural networks* (Vol. 4, pp. 1942–1948).

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*, 273–324.

Kundu, P. P., & Mitra, S. (2015). Multi-objective optimization of shared nearest neighbor similarity for feature selection. *Applied Soft Computing*, *37*, 751 – 762.

Li, X., & Yao, X. (2009). Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In *Ieee congress on evolutionary computation (cec '09)* (p. 1546-1553).

Li, X., & Yao, X. (2012). Cooperatively coevolving particle swarms for large scale optimization. *IEEE Transactions Evolutionary Computation*, *16*(2), 210-224.

Li, Y., Zhang, S., & Zeng, X. (2009). Research of multi-population agent genetic algorithm for feature selection. *Expert Systems with Applications*, *36*(9), 11570–11581.

Liu, Motoda, H., Setiono, R., & Zhao, Z. (2010). Feature selection: An ever evolving frontier in data mining. In *Fsdm* (Vol. 10, p. 4-13). JMLR.org.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*(4), 491–502.

Liu, X., & Shang, L. (2013). A fast wrapper feature subset selection method based on binary particle swarm optimization. In *Evolutionary computation (cec), 2013 ieee congress on* (pp. 3347–3353).

Manimala, K., Selvi, K., & Ahila, R. (2011). Hybrid soft computing techniques for feature selection and parameter optimization in power quality data mining. *Applied Soft Computing*, *11*(8), 5485–5497.

Marill, T., & Green, D. (1963). On the effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, *9*(1), 11–17.

Mishra, D., Rath., A., Acharya., M., & Jena., T. (2009). Rough aco: A hybridized model for feature selection in gene expression data. *International Journal of Computer Communication and Technology*, *1*, 85-98.

Mohamad, M., Omatu, S., Deris, S., & Yoshioka, M. (2011). A modified binary particle swarm optimization for selecting the small subset of informative genes from gene expression data. *IEEE Transactions on Information Technology in Biomedicine*, *15*(6), 813-822.

Mohamad, M., Omatu, S., Deris, S., Yoshioka, M., Abdullah, A., & Ibrahim, Z. (2013). An enhancement of binary particle swarm optimization for gene selection in classifying cancer classes. *Algorithms for Molecular Biology*, *8*(1).

Mukhopadhyay, A., & Mandal, M. (2012). A hybrid multiobjective particle swarm optimization approach for non-redundant gene marker selection. In *Bic-ta (1)'12* (p. 205-216).

Neshatian, K., & Zhang, M. (2009). Dimensionality reduction in face detection: A genetic programming approach. In *24th international conference image and vision computing new zealand (ivcnz'09)* (pp. 391–396). IEEE.

Neshatian, K., & Zhang, M. (2011). Using genetic programming for context-sensitive feature

scoring in classification problems. *Connection Science*, *23*(3), 183-207.

O'Boyle, N., Palmer, D., Nigsch, F., & Mitchell, J. (2008). Simultaneous feature selection and parameter optimisation using an artificial ant colony: case study of melting point prediction. *Chemistry Central Journal*, *2*(1).

Oh, I.-S., Lee, J.-S., & Moon, B.-R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(11), 1424 -1437.

Omidvar, M., Li, X., Mei, Y., & Yao, X. (2014, June). Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, *18*(3), 378–393.

Pudil, P., Novovicova, J., & Kittler, J. V. (1994). Floating search methods in feature selection. *Pattern Recognition Letters*, *15*(11), 1119–1125.

Rauber, T., de Assis Boldt, F., & Varejao, F. (2015). Heterogeneous feature models and feature selection applied to bearing fault diagnosis. *IEEE Transactions on Industrial Electronics*, *62*(1), 637–646.

Santana, L., Silva, L., Canuto, A., Pintro, F., & Vale, K. (2010). A comparative analysis of genetic algorithm and ant colony optimization to select attributes for an heterogeneous ensemble of classifiers. In *Ieee congress on evolutionary computation* (pp. 1–8).

Shen, Q., Shi, W.-M., & Ye, W. K. B.-X. (2007). A combination of modified particle swarm optimization algorithm and support vector machine for gene selection and tumor classification. *Talanta*, *71*(4), 1679–1683.

Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimization. In *Ieee congress on evolutionary computation (cec'99)* (Vol. 3, pp. 1945–1950).

Singhi, S. K., & Liu, H. (2006). Feature subset selection bias for classification learning. In *Proceedings of the 23rd international conference on machine learning* (pp. 849–856). New York, NY, USA: ACM.

Stearns, S. (1976). On selecting features for pattern classifier. In *Proceedings of the 3rd international conference on pattern recognition* (pp. 71–75). Coronado, Calif, USA: IEEE Press.

Talbi, E. G., Jourdan, L., Garcia-Nieto, J., & Alba, E. (2008). Comparison of population based metaheuristics for feature selection: Application to microarray data classification. In *Ieee/acs international conference on computer systems and applications (aiccsa'08)* (pp. 45–52).

Tran, B., Xue, B., & Zhang, M. (2014a). Improved pso for feature selection on high-dimensional datasets. In *Simulated evolution and learning* (Vol. 8886, pp. 503–515). Springer International Publishing.

Tran, B., Xue, B., & Zhang, M. (2014b). Overview of particle swarm optimisation for feature selection in classification. In *Simulated evolution and learning* (Vol. 8886, pp. 605–617). Springer International Publishing.

Vergara, J., & Estévez, P. (2014). A review of feature selection methods based on mutual information. *Neural Computing and Applications*, *24*(1), 175–186.

Wang, K.-J., Chen, K.-H., & Angelia, M.-A. (2014). An improved artificial immune recognition system with the opposite sign test for feature selection. *Knowledge-Based Systems*, *71*(0), 126–145.

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

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

Xue, B., Zhang, M., & Browne, W. N. (2015). A comprehensive comparison on evolutionary feature selection approaches to classification. *International Journal of Computational Intelligence and Applications*, *14*(02), 1550008.

Yang, C. S., Chuang, L. Y., Ke, C. H., & Yang, C. H. (2008). Boolean binary particle swarm optimization for feature selection. In *Ieee congress on evolutionary computation (cec'08)* (pp. 2093–2098).

Yu, H., Gu, G., Liu, H., Shen, J., & Zhao, J. (2009). A modified ant colony optimization algorithm

for tumor marker gene selection. *Genomics, Proteomics & Bioinformatics*, *7*(4), 200–208.
Zhai, Y., Ong, Y.-S., & Tsang, I. (2014). The emerging "big dimensionality". *IEEE Computational Intelligence Magazine*, *9*(3), 14–26.
Zhu, Z., Ong, Y.-S., & Dash, M. (2007). Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognition*, *40*(11), 3236–3248.

Table 1. Datasets: the number of features, instances and classes

Dataset	**#Features**	**#Instances**	**#Classes**
SRBCT	2308	83	4
DLBCL	5469	77	2
9 Tumors	5726	60	9
Leukemia 1	5327	72	3
Brain Tumor 1	5920	90	5
Brain Tumor 2	10367	50	4
Prostate Tumor	10509	102	2
Leukemia 2	11225	72	3
11 Tumors	12533	174	11
Lung Cancer	12600	203	5

Table 2. Parameters in PSO-LSRG method

Parameter	**Parameter value**
Maximum iteration	70
Population size	100
Acceleration constants ($c1 = c2$)	2.0
Inertia weight (w)	$0.9 - 0.5 * \frac{current\ iteration}{max\ iteration}$
Maximum velocity (v_{max})	0.6
Threshold for selected feature (θ)	0.6
Communication topology	Fully connected
gbest resetting time	when *gbest* stays the same for 3 iterations
Local search try	100
Local search flipping percentage	2%

Table 3. Experimental results I

Dataset	**Method**	**Ave-Size**	**Best**	**Mean±StdDev**	T_{RG}	T_{LS}	T_{LSRG}
SRBCT	All	2308	91.57		–	–	–
	PSO	1091.35	100.00	99.34±0.60	–	–	–
	PSO-RG	206.25	**100.00**	**100.00**±0.00		=	=
	PSO-LS	667.50	**100.00**	**100.00**±0.00			=
	PSO-LSRG	**153.95**	**100.00**	**100.00**±0.00			
DLBCL	All	5469	87.01		–	–	–
	PSO	2639.95	98.70	97.92±0.82	–	–	–
	PSO-RG	828.15	**100.00**	99.22±0.71		=	–
	PSO-LS	1682.88	**100.00**	99.06±0.59			–
	PSO-LSRG	**817.55**	**100.00**	**99.90**±0.35			
9 Tumors	All	5726	53.33		–	–	–
	PSO	2821.15	80.00	75.08±2.23	–	–	–
	PSO-RG	2303.78	83.33	77.04±2.65		–	–
	PSO-LS	**1734.45**	**86.67**	**84.17**±1.77			+
	PSO-LSRG	1872.98	88.33	83.13±2.24			
Leukemia 1	All	5327	87.50		–	–	–
	PSO	2539.33	**100.00**	99.76±0.62	–	–	–
	PSO-RG	1028.70	**100.00**	**100.00**±0.00		=	=
	PSO-LS	1668.83	**100.00**	**100.00**±0.00			=
	PSO-LSRG	**711.90**	**100.00**	**100.00**±0.00			
Brain Tumor 1	All	5920	86.67		–	–	–
	PSO	2852.68	93.33	92.64±0.65	–	–	–
	PSO-RG	**939.15**	96.67	94.53±0.81		=	–
	PSO-LS	1770.83	96.67	95.03±0.91			–
	PSO-LSRG	1014.63	**97.78**	**96.03**±1.36			
Brain Tumor 2	All	10367	70.00		–	–	–
	PSO	5016.08	92.00	88.10±1.86	–	–	–
	PSO-RG	1826.03	**96.00**	92.75±1.41		–	–
	PSO-LS	3107.18	**96.00**	93.60±0.93			–
	PSO-LSRG	**1688.28**	**96.00**	**94.55**±1.11			
Prostate Tumor	All	10509	76.47		–	–	–
	PSO	5138.50	91.18	89.64±1.27	–	–	–
	PSO-RG	**1461.35**	**98.04**	93.83±2.20		=	–
	PSO-LS	3120.78	96.08	94.32±1.20			–
	PSO-LSRG	1899.15	**98.04**	**95.91**±1.13			
Leukemia 2	All	11225	93.06		–	–	–
	PSO	5279.35	**100.00**	98.12±0.97	–	–	–
	PSO-RG	**982.20**	**100.00**	**100.00**±0.00		=	=
	PSO-LS	3612.95	**100.00**	99.93±0.31			=
	PSO-LSRG	1005.70	**100.00**	**100.00**±0.00			
11 Tumors	All	12533	84.48		–	–	–
	PSO	6178.63	92.53	91.25±0.80	–	–	–
	PSO-RG	4798.98	95.40	92.77±1.00		–	–
	PSO-LS	**3819.18**	**97.70**	**95.96**±0.93			+
	PSO-LSRG	3969.58	96.55	95.32±0.71			
Lung Cancer	All	12600	90.15		–	–	–
	PSO	6124.35	97.04	96.08±0.44	–	–	–
	PSO-RG	3659.15	98.03	96.86±0.60		–	–
	PSO-LS	3976.43	98.03	97.51±0.46			=
	PSO-LSRG	**2874.80**	**98.52**	**97.64**±0.45			

Table 4. Average computation time (in seconds)

	PSO	PSO-RG	PSO-LS	PSO-LSRG
SRBCT	41.16	31.53	87.89	77.70
DLBCL	120.02	71.27	173.43	160.21
9 Tumors	81.27	67.01	151.94	151.79
Leukemia 1	33.34	68.60	139.55	117.95
Brain Tumor 1	170.43	91.44	270.02	233.02
Brain Tumor 2	104.87	59.34	172.52	160.05
Prostate Tumor	376.81	203.22	818.00	640.41
Leukemia 2	201.97	109.14	315.94	261.25
11 Tumors	1401.14	1016.37	2568.87	2210.78
Lung Cancer	1725.75	1165.99	2639.05	2714.76

Table 5. Experimental results II

Dataset	**Method**	**Ave-Size**	**Best**	**Mean±StdDev**	T_{PSO}	T_{RG}	T_{LS}	T_{LSRG}
	All	2308	91.53		–	–	–	–
	PSO	1097.07	96.53	94.01±1.33		=	=	=
SRBCT	PSO-RG	300.91	**97.78**	93.91±1.80			=	=
	PSO-LS	564.89	96.67	**94.48**±1.52				=
	PSO-LSRG	**15.01**	97.64	93.26±2.86				
	All	5469	86.79		–	–	–	–
	PSO	2630.98	93.21	88.20±2.27		=	=	=
DLBCL	PSO-RG	961.66	93.21	87.96±2.50			=	=
	PSO-LS	1433.77	**93.39**	**88.94**±2.61				=
	PSO-LSRG	**243.18**	91.96	88.00±2.22				
	All	5726	52.89		–	–	–	=
	PSO	2811.74	**66.89**	55.52±4.49		=	=	=
9 Tumors	PSO-RG	2291.53	64.48	**56.30**±4.35			=	=
	PSO-LS	1572.93	64.39	55.74±4.15				=
	PSO-LSRG	**1457.35**	62.88	54.50±4.70				
	All	5327	85.77		–	–	–	–
	PSO	2560.57	94.40	**90.98**±1.91		+	=	+
Leukemia 1	PSO-RG	1257.29	**96.90**	89.91±2.67			=	+
	PSO-LS	1418.89	95.89	90.69±2.52				+
	PSO-LSRG	**236.45**	93.21	87.84±2.89				
	All	5920	85.56		–	=	–	+
	PSO	2834.34	90.00	**87.22**±1.67		+	=	+
Brain Tumor 1	PSO-RG	1086.47	90.00	85.20±2.19			–	+
	PSO-LS	1584.28	**91.11**	86.45±1.67				+
	PSO-LSRG	**573.31**	88.89	83.72±2.29				
	All	10367	71.83		–	=	–	–
	PSO	5000.16	81.83	73.18±3.58		=	–	=
Brain Tumor 2	PSO-RG	1897.58	79.83	72.29±4.26			–	=
	PSO-LS	2909.04	81.83	**75.63**±3.58				=
	PSO-LSRG	**882.37**	**85.83**	73.85±4.57				
	All	10509	77.36		=	–	–	–
	PSO	5127.39	82.36	77.67±2.09		–	–	–
Prostate Tumor	PSO-RG	1846.29	**87.36**	**82.25**±2.75			+	=
	PSO-LS	2980.81	84.36	80.57±2.58				=
	PSO-LSRG	**972.25**	86.36	81.79±2.18				
	All	11225	93.04		–	+	=	+
	PSO	5286.72	**97.14**	**93.53**±2.03		+	=	+
Leukemia 2	PSO-RG	1549.13	95.71	91.05±2.54			–	+
	PSO-LS	3260.81	**97.14**	92.90±1.95				+
	PSO-LSRG	**186.27**	93.39	89.36±2.63				
	All	12533	**85.47**		=	+	+	+
	PSO	6151.74	88.28	85.10±1.86		=	=	=
11 Tumors	PSO-RG	4725.39	87.66	84.53±1.52			=	=
	PSO-LS	3572.28	87.14	84.52±1.55				=
	PSO-LSRG	**3283.47**	**88.38**	84.59±1.45				
	All	12600	90.14		=	–	=	=
	PSO	6136.85	**93.12**	90.48±1.34		=	+	=
Lung Cancer	PSO-RG	3693.10	**93.12**	**90.53**±1.31			+	=
	PSO-LS	3481.53	**93.12**	89.82±1.11				–
	PSO-LSRG	**2271.19**	92.62	90.38±1.13				

Table 6. Top ten most frequent features selected by PSO-LSRG

Dataset	**Feature index**
SRBCT	509, 742, 1955, 1601, 545, 1954, 276, 1389, 151, 1750
DLBCL	4385, 5452, 3127, 3515, 52, 3220, 733, 4644, 4606, 1559
9Tumors	4492, 5160, 3407, 1816, 1431, 3437, 4819, 1429, 5180, 5032
Leuk1	1342, 3029, 1415, 1330, 1365, 4201, 2525, 851, 1267, 1426
Brain1	4431, 467, 16, 3712, 485, 881, 3748, 4794, 903, 5175
Leuk2	6720, 8025, 10038, 10355, 9482, 7561, 7377, 7573, 29, 937
Brain2	4649, 1198, 10343, 8131, 3963, 6814, 2145, 5575, 7449, 8615
Prostate	9241, 10417, 9949, 120, 9038, 3200, 8529, 5660, 7310, 7652
11Tumors	631, 12215, 6821, 3946, 6822, 6925, 3624, 6140, 11320, 1465
Lung	4366 (**41469_at**), 1060 (**34091_s_at**), 7383 (**41231_f_at**), 8429 (**36105_at**), 7053 (**40422_at**), 4666 (**32052_at**), 4983 (**33322_i_at**), 7199 (**40808_at**), 8803 (**37319_at**), 3845 (**39990_at**)

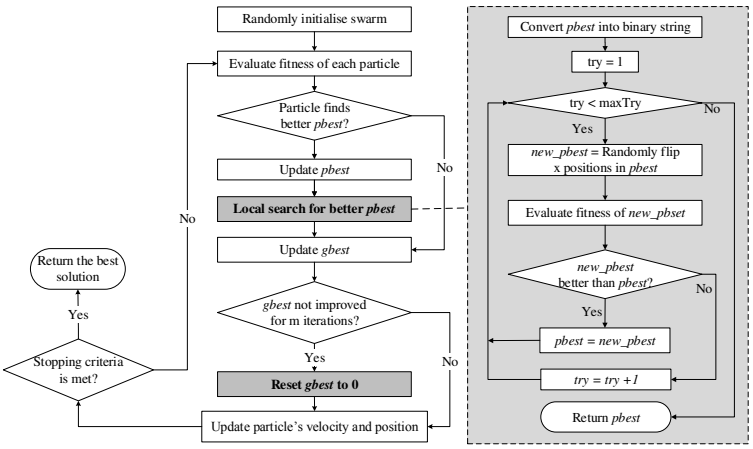

Figure 1. Flowchart of PSO-LSRG.

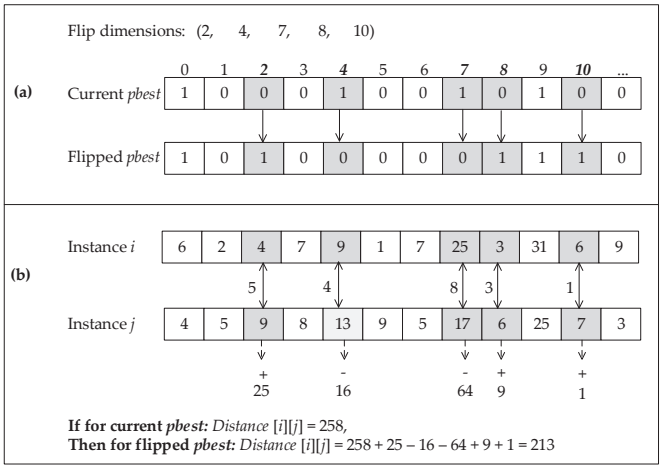

Figure 2. An example of one "local search" step and re-calculating instances' distance.

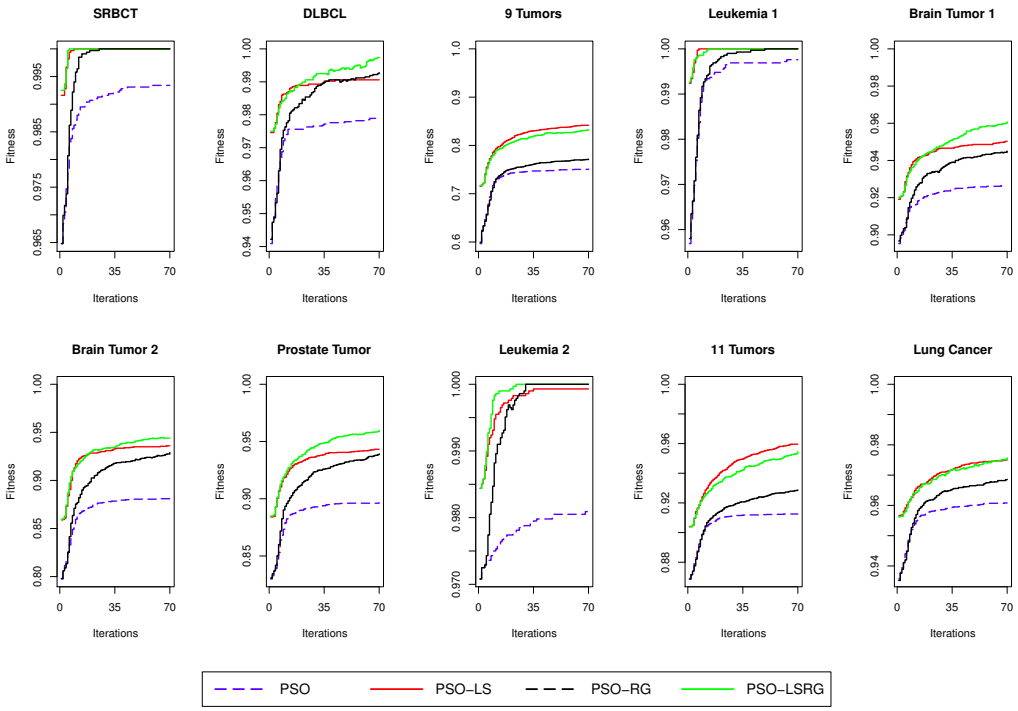

Figure 3. Average *gbest* fitness of PSO, PSO-RG, PSO-LS and PSO-LSRG (in colour).

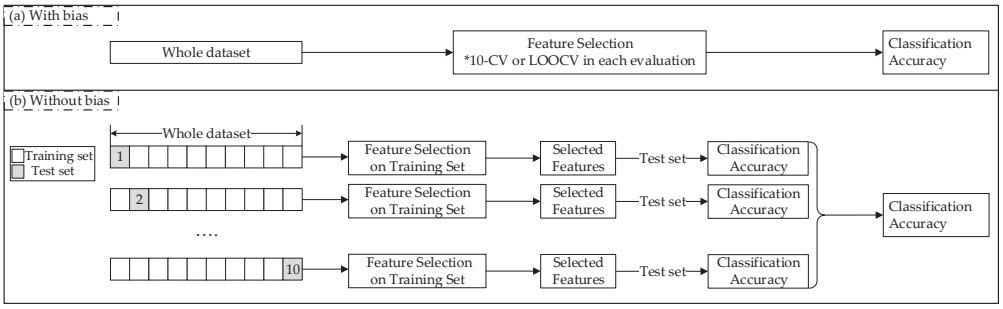

Figure 4. Structures of Experiments with and without feature selection bias.

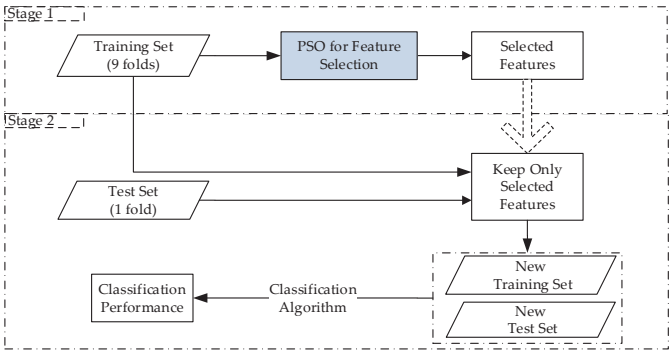

Figure 5. Overview of feature selection in one of the 10-fold cross validation.