Particle Swarm Optimisation for Feature Selection in Classification

by

Bing Xue

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2014

Abstract

Classification problems often have a large number of features, but not all of them are useful for classification. Irrelevant and redundant features may even reduce the classification accuracy. *Feature selection* is a process of selecting a subset of relevant features, which can decrease the dimensionality, shorten the running time, and/or improve the classification accuracy. There are two types of feature selection approaches, i.e. *wrapper* and *filter* approaches. Their main difference is that wrappers use a classification algorithm to evaluate the goodness of the features during the feature selection process while filters are independent of any classification algorithm. Feature selection is a difficult task because of feature interactions and the large search space. Existing feature selection methods suffer from different problems, such as stagnation in local optima and high computational cost. Evolutionary computation (EC) techniques are well-known global search algorithms. *Particle swarm optimisation* (PSO) is an EC technique that is computationally less expensive and can converge faster than other methods. PSO has been successfully applied to many areas, but its potential for feature selection has not been fully investigated.

The overall goal of this thesis is to investigate and improve the capability of PSO for feature selection to select a smaller number of features and achieve similar or better classification performance than using all features. This thesis investigates the use of PSO for both wrapper and filter, and for both single objective and multi-objective feature selection, and also investigates the differences between wrappers and filters.

This thesis proposes a new PSO based wrapper, single objective feature selection approach by developing new initialisation and updating mechanisms. The results show that by considering the number of features in the

initialisation and updating procedures, the new algorithm can improve the classification performance, reduce the number of features and decrease computational time.

This thesis develops the first PSO based wrapper multi-objective feature selection approach, which aims to maximise the classification accuracy and simultaneously minimise the number of features. The results show that the proposed multi-objective algorithm can obtain more and better feature subsets than single objective algorithms, and outperform other well-known EC based multi-objective feature selection algorithms.

This thesis develops a filter, single objective feature selection approach based on PSO and information theory. Two measures are proposed to evaluate the relevance of the selected features based on each pair of features and a group of features, respectively. The results show that PSO and information based algorithms can successfully address feature selection tasks. The group based method achieves higher classification accuracies, but the pair based method is faster and selects smaller feature subsets.

This thesis proposes the first PSO based multi-objective filter feature selection approach using information based measures. This work is also the first work using other two well-known multi-objective EC algorithms in filter feature selection, which are also used to compare the performance of the PSO based approach. The results show that the PSO based multi-objective filter approach can successfully address feature selection problems, outperform single objective filter algorithms and achieve better classification performance than other multi-objective algorithms.

This thesis investigates the difference between wrapper and filter approaches in terms of the classification performance and computational time, and also examines the generality of wrappers. The results show that wrappers generally achieve better or similar classification performance than filters, but do not always need longer computational time than filters. The results also show that wrappers built with simple classification algorithms can be general to other classification algorithms.

List of Publications

1. Bing Xue, Mengjie Zhang, Will N. Browne. "Particle Swarm Optimisation for Feature Selection in Classification: A Multi-Objective Approach". IEEE Transactions on Systems, Man, and Cybernetics (Part B). (13 Dec 2012 published online). DOI:10.1109/TSMCB.2012.2227469.

2. Bing Xue, Mengjie Zhang, Will N. Browne. "Particle Swarm Optimisation for Feature Selection in Classification: Novel Initialisation and Updating Mechanisms". Applied Soft Computing. (Accepted, 30-09-2013).

3. Bing Xue, Liam Cervante, Lin Shang, Will N. Browne, Mengjie Zhang. "A Multi-Objective Particle Swarm Optimisation for Filter Based Feature Selection in Classification Problems". Connection Science. Vol. 24, No. 2-3, 2012. pp. 91-116 DOI: 10.1080/09540091.2012.737765.

4. Bing Xue, Liam Cervante, Lin Shang, Will N. Browne, Mengjie Zhang. "Multi-Objective Evolutionary Algorithms for Filter Based Feature Selection in Classification". International Journal on Artificial Intelligence Tools. Vol. 22, Issue 04, August 2013. pp. 1350024-1 – 1350024-31. DOI:10.1142/S0218213013500243.

5. Bing Xue, Mengjie Zhang, Will N. Browne. "Novel Initialisation and Updating Mechanisms in PSO for Feature Selection in Classification". Proceedings of the 16th European Conference on Applications of Evolutionary Computation (EvoApplications 2013). Lecture

Notes in Computer Science. Vol. 7835. Vienna, Austria, April 3-5, 2013. pp. 428-438.

6. Bing Xue, Mengjie Zhang, Yan Dai, Will N. Browne. "PSO for Feature Construction and Binary Classification". Proceedings of 2013 Genetic and Evolutionary Computation Conference. Amsterdam, The Netherlands. July 06-10, 2013. pp. 137-144.

7. Bing Xue, Mengjie Zhang, Will N. Browne. "Multi-Objective Particle Swarm Optimisation (PSO) for Feature Selection". Proceedings of 2012 Genetic and Evolutionary Computation Conference (GECCO 2012). ACM Press. Philadelphia, USA. 7-11 July 2012. pp. 81-88.

8. Bing Xue, Mengjie Zhang and Will N. Browne. "New Fitness Functions in Binary Particle Swarm Optimisation for Feature Selection". Proceedings of 2012 IEEE Congress on Evolutionary Computation. Australian. 10-17 June, 2012. IEEE Press. pp. 2145-2152.

9. Bing Xue, Mengjie Zhang, Will N. Browne. "Single Feature Ranking and Binary Particle Swarm Optimisation Based Feature Subset Ranking for Feature Selection". Proceedings of the Thirty-Fifth Australasian Computer Science Conference (ACSC 2012). Melbourne, Australia. 30 Jan - 2 Feb 2012. pp. 27-36.

10. Mitchell C. Lane, Bing Xue, Ivy Liu and Mengjie Zhang. "Particle Swarm Optimisation and Statistical Clustering for Feature Selection". Proceedings of the 26th Australasian Joint Conference on Artificial Intelligence (AI2013) Lecture Notes in Computer Science. Dunedin, New Zealand. 1-6 December 2013. (To appear)

11. Aneta Stevanovic, Bing Xue and Mengjie Zhang. "Feature Selection Based on PSO and Decision-Theoretic Rough Set Model". Proceedings of 2013 IEEE Congress on Evolutionary Computation. Cancun, Mexico. 20-23 June, 2013. IEEE Press. pp. 2840–2847.

12. Liam Cervante, Bing Xue, Lin Shang and Mengjie Zhang. "Binary Particle Swarm Optimisation and Rough Set Theory for Dimension Reduction in Classification. Proceedings of 2013 IEEE Congress on Evolutionary Computation. Cancun, Mexico. 20-23 June, 2013. IEEE Press. pp. 2428–2435.

13. Liam Cervante, Bing Xue, Lin Shang, Mengjie Zhang. "A Multi-Objective Feature Selection Approach Based on Binary PSO and Rough Set Theory". Proceedings of the 13th European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2013). Lecture Notes in Computer Science. Vol. 7832. Vienna, Austria. 3-5 April 2013. pp. 25-36.

14. Liam Cervante, Bing Xue, Lin Shang and Mengjie Zhang. "A Dimension Reduction Approach to Classification Based on Particle Swarm Optimisation and Rough Set Theory". Proceedings of the 25th Australasian Joint Conference on Artificial Intelligence. Lecture Notes in Artificial Intelligence.Vol. 7691. Springer. Sydney, Australia, December 2012. pp. 313-325.

15. Bing Xue, Liam Cervante, Lin Shang and Mengjie Zhang. "A Particle Swarm Optimisation Based Multi-Objective Filter Approach to Feature Selection for Classification". Proceeding of the 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2012). Lecture Notes in Artificial Intelligence. Vol. 7458. Kuching, Sarawak, Malaysia. 3-7 September 2012. pp. 673-685.

16. Liam Cervante, Bing Xue and Mengjie Zhang. "Binary Particle Swarm Optimisation for Feature Selection: A Filter Based Approach". Proceedings of 2012 IEEE Congress on Evolutionary Computation. Brisbane, Australian. 10-17 June, 2012. IEEE Press. pp. 881-888.

Acknowledgments

I would like to express my sincere gratitude to those who gave me the assistance and support during my PhD study.

My deepest gratitude goes to my supervisors, Prof Mengjie Zhang and Dr Will N. Browne. Prof. Mengjie Zhang has spent dedicated time and efforts to train my research skills, and provided constructive and challenging feedbacks to improve my research work. This thesis would not have been possible without his encouragement, motivation, inspiration, and guidance. Dr Will N. Browne is always nice to talk to, and the discussions with him always bring me to a further thinking of my research. I am also grateful to him for helping me improve my research writing skills.

I am grateful for the Victoria Doctoral Scholarship, the Marsden Fund of New Zealand (VUW0806), the University Research Fund (200457/3230), School of Engineering and Computer Science and Faculty of Engineering at Victoria University of Wellington for their financial support over the past three years.

I wish to thank my friends in the Evolutionary Computation Research Group (ECRG) for creating an active and interesting research environment. Thank you to my officemates Su and Juan for their help in my study and lots of lovely jokes. Thank you to my friends Haneim and Wenlong for sharing their delicious food.

Last but not least, I wish to thank my family for their love, encouragement and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces this thesis. It starts with the problem statement, then outlines the motivations, the research goals, the major contributions and the organisation of the thesis.

1.1 Problem Statement

Classification is an important task in machine learning and data mining, which aims to classify each instance in the data into different groups. The feature space of a classification problem is a key factor influencing the performance of a classification/learning algorithm [2]. Without prior knowledge, it is difficult to determine which features are useful. Therefore, a large number of features are usually introduced into the dataset, including relevant, irrelevant and redundant features. However, irrelevant and redundant features are not useful for classification. Their presence may mask or obscure the useful information provided by relevant features, and hence reduces the quality of the whole feature set [3]. Meanwhile, the large number of features causes one of the major obstacles in classification known as "the curse of dimensionality" [4]. Therefore, **feature selection** is proposed to increase the quality of the feature space, reduce the number of features and/or improve the classification performance [5, 6, 7].

Feature selection (See Figure 1.1) aims to select a subset of relevant features that are necessary and sufficient to describe the target concept [1]. By reducing the irrelevant and redundant features, feature selection could decrease the dimensionality, reduce the amount of data needed for the learning process, shorten the running time, simplify the structure and/or improve the performance of the learnt classifiers [1]. Naturally, an optimal feature subset is the smallest feature subset that can obtain the optimal performance, which makes feature selection a multi-objective problem [8]. Note that feature selection algorithms choose a subset of features from the original feature set, and do not create new features.

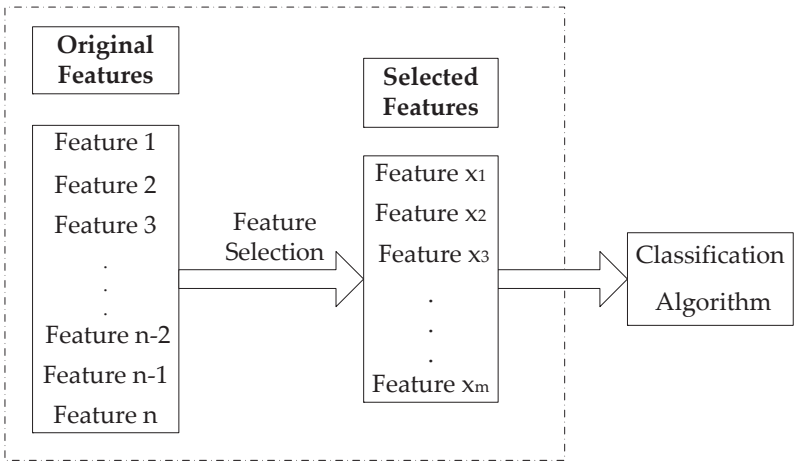

Figure 1.1: Feature selection, where m and n are constant, $1 \leq m < n$, and {Feature x_1 ... Feature x_m} $\in$ {Feature 1, ... Feature n } without replacement.

Existing feature selection methods can be broadly classified into two categories: filter approaches and wrapper approaches [5, 1]. Wrapper approaches include a classification algorithm as a part of the evaluation function to determine the goodness of the selected feature subsets. Filter approaches use statistical characteristics of the data for evaluation and the feature selection search process is independent of any classification algorithm. Filter approaches are computationally less expensive and more

general than wrapper approaches while wrappers are better than filters in terms of the classification performance [1].

Feature selection is a difficult task. Although many approaches have been proposed, most of them still suffer from the problems of stagnation in local optima and high computational cost due mainly to the large search space. Therefore, an efficient global search technique is needed to address feature selection tasks. Evolutionary computation (EC) techniques are a group of powerful global search algorithms, which have been successfully applied to many areas [9]. Particle swarm optimisation (PSO) [10, 11] is a relatively recent EC technique based on swarm intelligence. The underlying concept of PSO is that knowledge is optimised by social interactions, where the "thinking"/interaction is not only personal but also social [10, 12]. Compared with other EC techniques, such as genetic algorithms (GAs) and genetic programming (GP), PSO is easier to implement, has fewer parameters and can converge more quickly [9]. PSO has been recently applied to solve feature selection problems [13, 14, 15], but its potential on feature selection has not been fully investigated.

1.2 Motivations

1.2.1 Challenges of Feature Selection

Feature selection is a difficult problem [16, 17], especially when the number of available features is large. The task is challenging due mainly to two reasons, which are feature interaction and the large search space.

Feature interaction (also called epistasis [18]) happens frequently in classification tasks. There can be two-way, three-way or complex multi-way interactions among features. On one hand, a feature, which is weakly relevant or even completely irrelevant to the target concept by itself, can significantly improve the classification accuracy if it is complementary to other features. Therefore, the removal of such features may also miss the

optimal feature subsets. On the other hand, an individually relevant feature can become redundant when working together with other features. The selection/use of such features brings redundancy, which may deteriorate the classification performance.

In feature selection, the size of the search space grows exponentially with respect to the number of available features in the dataset (2^n possible subsets for n features) [19]. In most cases, it is practically impossible to exhaustively search all the candidate solutions. To better address this problem, a variety of search techniques have been applied to feature selection [1, 19]. However, existing methods still suffer from the problem of stagnation in local optima and/or high computational cost.

1.2.2 Why PSO

PSO is an effective and efficient global search technique [9, 12]. It is a suitable algorithm to address feature selection problems because of the following reasons:

- The representation of PSO is appropriate to feature selection tasks. PSO uses an array or a vector to encode each individual/particle in the swarm, which is a natural way to encode the candidate solution of a feature selection problem, where the dimensionality is the number of features and the value in each dimension shows whether the corresponding feature is selected;

- Feature selection has a large search space, which often causes the problem of becoming stuck in local optima in existing methods. So it needs a global search technique. EC methods are well-known for their global search ability. PSO is an EC algorithm that is able to effectively search large spaces to find optimal or near-optimal solutions;

- The large search space in feature selection also causes the problem of

high computational cost, especially for wrapper approaches. PSO is argued to be computationally less expensive than other EC methods;

- Compared with other EC methods, PSO is easier to implement, has fewer parameters and can converge faster; and

- There have been a limited number of studies on PSO for feature selection, which have shown that PSO has the potential to address feature selection problems. However, the capability/capacity of PSO for feature selection has not been fully investigated.

1.2.3 Limitations of Existing Work

PSO has been used for feature selection, but PSO has never been investigated/modified according to the characteristics of the feature selection task. There are many important factors in PSO, which can be investigated for a particular task to improve the performance, such as the initialisation strategy and the updating mechanisms. Therefore, tuning the PSO algorithm according to the objectives of the feature selection task can further improve the classification performance and/or reduce the number of features.

Feature selection is a multi-objective problem. It has two main objectives, which are to maximise the classification accuracy (minimise the classification error rate) and minimise the number of features. These two objectives are usually conflicting to each other and the optimal decision needs to be made in the presence of a trade-off between them. Treating feature selection as a multi-objective problem can obtain a set of non-dominated feature subsets to meet different requirements in real-world applications. However, there are rare studies treating feature selection as a multi-objective problem [20, 21]. Although PSO has been shown to be successful in addressing many multi-objective problems [9], PSO has never been applied to multi-objective feature selection tasks.

Most of the existing PSO based feature selection algorithms are wrapper approaches, which are argued to be less general, that is the selected features may obtain low performance in classification algorithms rather than the internal classification algorithm used in the evaluation function. Meanwhile, as each evaluation involves a training and testing classification process, wrappers are usually computationally expensive. Filter approaches are argued to be more general and computationally less expensive. However, there are very few studies on using PSO for filter feature selection and no work on using PSO for filter based multi-objective feature selection.

Wrapper and filter approaches are argued to have their own advantages and disadvantages. Wrappers can achieve better classification performance than filters, but filters are computationally less expensive and more general than wrappers. However, no thorough investigations have been made on how much difference there probably is between the two approaches in terms of the classification performance and the computational cost.

1.3 Goals

The overall goal of this thesis is to investigate/improve the capability of PSO for feature selection and propose a new approach to the use of PSO for feature selection in classification problems to reduce the number of features and achieve similar or even better classification performance than using all the original features. To achieve this overall goal, a set of research objectives have been established to guide this research, which can be seen as follows.

1. Develop a new *wrapper* based *single objective* feature selection method by proposing new initialisation and updating mechanisms in PSO. The proposed algorithm is expected to select a small number of features and achieve similar or better classification performance than

using all features and outperform existing PSO based feature selection algorithms.

In existing work, PSO has never been tuned to the feature selection task. The initialisation strategy in PSO can improve its performance. Gutierrez et al. [22] show that the influence of the initialisation strategy varies in different tasks. However, no existing initialisation strategies are specifically proposed for the feature selection problems. Meanwhile, in standard PSO, the personal and global best are updated solely based on the fitness value of the particles (i.e. classification performance in feature selection problems). For two feature subsets with the same classification performance but different numbers of features, the traditional updating mechanism does not tend to choose the smaller feature subset.

2. Develop a new *wrapper* based *multi-objective* feature selection algorithm using a multi-objective PSO algorithm. The proposed algorithm is expected to evolve a set of non-dominated feature subsets with a smaller number of features and better classification performance than using all features, and outperform other well-known EC based multi-objective feature selection methods.

   EC techniques are particularly suitable for multi-objective optimisation because they use a population of candidate solutions and are able to find multiple non-dominated solutions in a single run. PSO has been successfully used to solve many multi-objective problems, but has never applied to multi-objective feature selection. The proposed PSO based multi-objective feature selection algorithm aims to optimise the two main objectives, i.e. maximising the classification performance and minimising the number of features. It is expected to select a set of non-dominated solutions rather than a single solution, which can assist users in choosing their preferred solutions to meet different requirements in real-world applications.

3. Develop a new *filter* based *single objective* feature selection approach based on PSO and an information theory measure to evaluate the goodness of the selected features. The proposed algorithm is expected to efficiently select a small feature subset with which different classification algorithms can achieve similar or even better classification performance than using all features.

   Filter approaches are argued to be computationally less expensive and more general than wrapper approaches. The main reason is that filters use statistical characteristics of the data as the evaluation measure rather than using a classification algorithm as in wrappers. The evaluation measure is a key factor in a filter approach. Information theory is one of the most important theories that are capable to measure the relevance between features and class labels [1]. However, no work has been conducted to investigate the use of information theory in PSO based feature selection.

4. Develop a *filter* based *multi-objective* feature selection approach based on multi-objective PSO and the information theory based measure. The proposed approach is expected to efficiently select a set of non-dominated feature subsets to achieve similar or even better classification performance than using all features.

   EC techniques are particularly suitable for multi-objective optimisation. However, multi-objective PSO and other two well-known evolutionary multi-objective algorithms, non-dominated sorting based multi-objective genetic algorithm II (NSGAII) [23] and strength Pareto evolutionary algorithm 2 (SPEA2) [24] have not been applied to filter based multi-objective feature selection. In this objective, multi-objective PSO, NSGAII and SPEA2 will be used to develop filter based multi-objective methods. The proposed multi-objective PSO based algorithm is expected to achieve better performance than the PSO based single objective method, the NSGAII and SPEA2 based

multi-objective methods.

5. Further investigate the differences between wrapper and filter feature selection methods in terms of the computational time, the generality and classification performance.

   Wrappers are usually considered to be less general than filter approaches. To examine the generality of wrapper approaches, this objective will investigate whether the feature subset selected by using a certain classification algorithm during the training process can improve the performance of other classification algorithms. Meanwhile, it is also needed to test the difference on the classification performance and the computational time between wrapper and filter approaches, which can provide more information to help users make a choice between filter and wrapper methods according to their requirements on the classification performance and the computational time.

1.4 Major Contributions

This thesis makes the following major contributions.

1. The thesis proposes a PSO based feature selection approach based on a new initialisation strategy and a new personal and global best updating mechanism. The new initialisation strategy simulates conventional forward and backward feature selection methods [1]. The new updating mechanism considers both the classification performance and the number of features. In addition, a new PSO based two-stage feature selection method is proposed as a baseline algorithm. Experimental results show that the two-stage algorithm can achieve better performance than traditional feature selection methods and existing PSO based methods. With the new initialisation and updating mechanisms, the proposed method further improves

the performance in terms of the classification performance, the number of features and the computational time.

Part of this contribution has been published in:

Bing Xue, Mengjie Zhang and Will N. Browne. "New Fitness Functions in Binary Particle Swarm Optimisation for Feature Selection". Proceedings of 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 2145-2152.

Bing Xue, Mengjie Zhang, Will N. Browne. "Novel Initialisation and Updating Mechanisms in PSO for Feature Selection in Classification". Proceedings of the 16th European Conference on Applications of Evolutionary Computation (EvoApplications 2013). Lecture Notes in Computer Science. Vol. 7835. Vienna, Austria, April 3-5, 2013. pp. 428-438.

Bing Xue, Mengjie Zhang, Will N. Browne. "Particle Swarm Optimisation for Feature Selection in Classification: Novel Initialisation and Updating Mechanisms". Applied Soft Computing. (Accepted, 30-09-2013).

2. This thesis proposes the first PSO based multi-objective, wrapper feature selection approach. The proposed algorithm aims to maximise the classification performance and minimise the number of features. Experimental results show that the proposed algorithm can successfully obtain a set of feature subsets that have a smaller number of features and achieve similar or better classification performance than using all features. The proposed algorithm outperforms PSO based single objective algorithms and three other multi-objective EC techniques based wrapper algorithms, which are NSGAII, SPEA2 and Pareto archived evolutionary strategy (PAES).

   Part of this contribution has been published in:

   Bing Xue, Mengjie Zhang, Will N. Browne. "Multi-Objective Particle

Swarm Optimisation (PSO) for Feature Selection". Proceedings of 2012 Genetic and Evolutionary Computation Conference (GECCO 2012). ACM Press. Philadelphia, USA. 7-11 July 2012. pp. 81-88.

Bing Xue, Mengjie Zhang, Will Browne. "Particle Swarm Optimisation for Feature Selection in Classification: A Multi-Objective Approach". IEEE Transactions on Systems, Man, and Cybernetics (Part B). (13 Dec 2012 published online). DOI: 10.1109/TSMCB.2012.2227469.

3. This thesis proposes a filter based single objective feature selection approach using PSO and new information theory based measures. The new measures are used to evaluate the relevance between features and the class labels, and the redundancy amongst the selected features, which indicates the classification performance and the number of features, respectively. New fitness functions using such relevance and redundancy measures are developed to guide PSO to search for the optimal feature subsets. Experimental results show that the proposed algorithms can select a small number of features to achieve similar or even better classification performance than using all features, and outperform two traditional filter feature selection algorithms and even a traditional wrapper method.

   Part of this contribution has been published in:

   Liam Cervante, Bing Xue and Mengjie Zhang. "Binary Particle Swarm Optimisation for Feature Selection: A Filter Based Approach". Proceedings of 2012 IEEE Congress on Evolutionary Computation. IEEE Press. 2012. pp. 881-888.

4. This thesis proposes the first multi-objective, filter feature selection approach using PSO, NSGAII or SPEA2. The proposed algorithms maximise an information measure representing the classification performance and minimise the number of features. Experimental results show that all the proposed multi-objective algorithms can obtain a set of feature subsets, which include a smaller number of features

and achieve similar or even better classification performance than using all features. The proposed multi-objective algorithms outperform traditional filter algorithms and PSO based single objective algorithms. The PSO based multi-objective approach achieves slightly better classification performance than the NSGAII and SPEA2 based algorithms.

Part of this contribution has been published in:

Bing Xue, Liam Cervante, Lin Shang, Will N. Browne, Mengjie Zhang. "A Multi-Objective Particle Swarm Optimisation for Filter Based Feature Selection in Classification Problems". Connection Science. Vol. 24, No. 2-3, 2012. pp. 91-116 DOI: 10.1080/09540091.2012.737765.

Bing Xue, Liam Cervante, Lin Shang and Mengjie Zhang. "A Particle Swarm Optimisation Based Multi-Objective Filter Approach to Feature Selection for Classification". Proceeding of the 12th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2012). Lecture Notes in Artificial Intelligence. Vol. 7458. Kuching, Sarawak, Malaysia. 3-7 September 2012. pp. 673-685.

Bing Xue, Liam Cervante, Lin Shang, Will N. Browne, Mengjie Zhang. "Multi-Objective Evolutionary Algorithms for Filter Based Feature Selection in Classification". International Journal on Artificial Intelligence Tools . Vol. 22, Issue 04, August 2013. pp. 1350024-1 – 1350024-31. DOI: 10.1142/S0218213013500243.

5. This thesis investigates the differences between wrapper and filter approaches in terms of the classification performance and the computational time, and examines the generality of different wrappers. The experimental results show that wrapper approaches generally achieve better or similar (but not worse) classification performance than filters, but wrapper approaches do not necessarily always need longer computational time than filter approaches. Wrapper approaches were previously claimed not general to different classification algo-

rithms, but this thesis finds that wrappers built with a simple classification algorithm can be general to other classification algorithms.

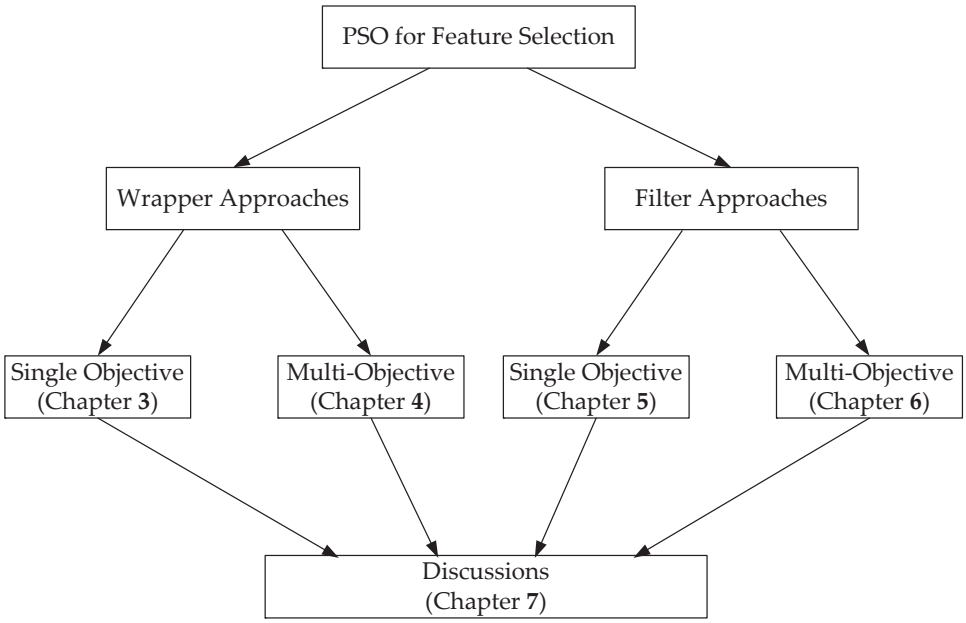

Figure 1.2: The overall structure of the contributions.

1.5 Organisation of the Thesis

The remainder of this thesis is organised as follows. Chapter 2 presents the literature review of related work. The main contributions of the thesis are presented in Chapters 3-7, which can be seen in Figure 1.2. Each chapter addresses one of the research objective. Chapter 8 concludes the thesis.

Chapter 2 presents essential background and basic concepts of machine learning and classification, feature selection, evolutionary computation particularly PSO, multi-objective optimisation and information theory. It reviews typical related work in feature selection using conventional methods and evolutionary computation techniques. It also visits advances in

feature selection using PSO and other EC methods. It then discusses open questions and current challenges that form the motivations of this thesis.

Chapter 3 proposes new initialisation and updating mechanisms in PSO for feature selection. It discusses the influence of the initialisation, updating mechanism and the fitness function on the classification performance and the number of features selected. The chapter then proposes new algorithms and examines their performance against traditional methods and existing PSO based algorithms. A set of experiments are conducted on commonly used classification problems of varying difficulty. The results are then presented and analysed.

Chapter 4 proposes a novel wrapper based multi-objective feature selection algorithm using multi-objective PSO, which aims to maximise the classification accuracy and minimise the number of features. The proposed algorithm is then examined and compared with single objective algorithms, and three other EC based multi-objective algorithms, i.e. NSGAII, SPEA2 and PAES in terms of the number of features, the classification performance and the computational time.

Chapter 5 develops a new filter algorithm based on PSO and information theory. It discusses the capability of information theory in measuring the relevance between class labels and a groups of features, and the redundancy with the features. New information based measures are then developed to form the fitness function of the new PSO based feature selection algorithm. The performance of the new algorithm is examined on a set of datasets and the classification performance is evaluated using different classification algorithms to test its generality.

Chapter 6 proposes new filter based multi-objective algorithms using multi-objective PSO and information based measures. It also proposes two other EC based multi-objective algorithms using NSGAII and SPEA2. All these algorithms aim to the information theory based relevance measure and minimise the number of features. Their performances are then examined and compared with PSO or GA based single objective algorithms on

datasets of varying difficulty.

Chapter 7 discusses and tests the advantages and disadvantages of both filter and wrapper approaches. Based on the experimental results, the classification performance, the computational time and the generality of filter and wrapper approaches are compared and discussed.

Chapter 8 summaries the work and draws overall conclusions of the thesis. Key research points and the contributions of the thesis are ascertained. It also suggests some possible future research directions.

1.6 Benchmark Datasets

Throughout this thesis, the proposed PSO based feature selection algorithms are evaluated on a number of benchmark classification tasks of varying difficulty.

The datasets are summarised in Table 1.1. These datasets are carefully choosen from the UCI Repository of Machine Learning Databases [25]. The datasets are selected to have different numbers of features (from 13 to 649), classes (from 2 to 19) and instances (from 32 to 44473), and different data type (continuous and categorical). The datasets are used as representative samples of the problems that the proposed algorithms can address.

Table 1.1: Datasets

Dataset	Number of Features	Number of Classes	Number of Instances	Data Type
Wine	13	3	178	Continuous
Australian	14	2	690	Continuous
Zoo	17	7	101	Continuous
Vehicle	18	4	846	Continuous
German	24	2	1000	Continuous
Wisconsin Breast Cancer-Diagnostic (WBCD)	30	2	569	Continuous
Ionosphere (Ionosp)	34	2	351	Continuous
Lung Cancer (Lung)	56	3	32	Continuous
Sonar	60	2	208	Continuous
Movementlibras (MoveLib)	90	15	360	Continuous
Hillvalley	100	2	606	Continuous
Musk Version1(Musk1)	166	2	476	Continuous
Madelon	500	2	4400	Continuous
Isolet5	617	2	1559	Continuous
Lymphography (Lymph)	18	4	148	Categorical
Mushroom	22	2	5644	Categorical
Spect	22	2	267	Categorical
Leddisplay	24	10	1000	Categorical
Dermatology	34	6	366	Categorical
Connect4	42	3	44473	Categorical
Soybean Large	35	19	307	Categorical
Chess	36	2	3196	Categorical
Splice	60	3	3190	Categorical
Statlog	36	6	6435	Categorical
Waveform	40	3	5000	Categorical

Chapter 2

Literature Review

This chapter provides a review on the literature that forms the background and supports the motivations of the thesis. This chapter covers essential background and basic concepts of machine learning and classification, feature selection, evolutionary computation (particularly PSO), multi-objective optimisation and information theory. It reviews typical related work in feature selection using conventional methods and evolutionary computation techniques.

2.1 Machine Learning and Classification

Machine learning is a major research area of artificial intelligence. It is concerned with the design, analysis, implementation, and application of programs that are capable of learning in the environment [26, 27, 28]. A machine learning system is expected to be able to automatically improve its performance at a certain task as it gains more experience [28].

Mitchell [28] provided a widely quoted definition of machine learning:

> "computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with

experience **E**" .

Machine learning algorithms use a feedback mechanism to change their behaviour (learn). Depending on the type of feedback, machine learning algorithms can be classified into three main categories: supervised learning, unsupervised learning and reinforcement learning [2].

- In supervised learning, the learner is learning with labelled examples or instances. The desired outputs for a problem are known in advance. The goal is to learn a function that maps inputs to the desired outputs. Classification is a typical form of supervised learning. Given a set of instances represented by features or attributes and corresponding class labels, classification involves learning a model to correctly predict the class membership of each instance [29].

- In unsupervised learning, the learner is learning with examples which are not labelled. This means that there are no correct answers from which the learner can explicitly learn [30]. It attempts to find inherent patterns that can then be used to determine groups for the given instances. An example of unsupervised learning is clustering, where the learner must explore underlying structures or correlations in the data to learn relationships rather than rules.

- In reinforcement learning, desired outputs are not directly provided. Every action of the learner has different impact to the environment, and the environment provides feedback on the goodness of its action in the form of rewards and punishments. The learner learns based on the rewards and punishments that it receives from the environment [31].

Classification is one of the major tasks in machine learning, which refers to the process of assigning a given piece of input data (an instance or example) to one of the given categories/classes [29]. A learnt classifier is needed for classification. The classifier is learnt by a learning/classification

algorithm, also called a classifier inducer, which is a supervised learning algorithm. The learning algorithm uses a set of examples to learn a classifier that is expected to correctly predict the class label of unseen (future) instances [29]. The learnt classifier takes the values of the features or attributes of an object as input and the predefined class labels for the object as output. The set of class labels is defined as part of the problem (by users).

A typical classification example is the email spam-catching system, which is important and necessary in real-world applications. Given a set of emails marked as "spam" and "non-spam" , the learner will learn the characteristics of the spam emails and then the learnt classifier is able to process future email messages to mark them as "spam" or "non-spam".

2.1.1 Training and Testing

Common to classification problems are the processes of training and testing. The process by which a learning algorithm (classifier inducer) uses observations to learn a new classifier is called the *training* process. The process by which the learnt classifier is tested on unseen observations is called the *testing* process [28]. During the training process, the classifier is learnt from a collection of observations from the problem domain called instances, which is called the *training set*. The algorithm learns important knowledge or rules in the training set by building models and adjusting the corresponding parameters. The performance of the algorithm is then evaluated on the *test set*, which is also a collection of instances in the same problem domain, but these are not used and remain *unseen* during the training process.

The learning ability of classification algorithms is usually examined by applying them to a set of benchmark problems. Benchmark problems are usually chosen from datasets that are publicly accessible to researchers (e.g. UCI Machine Learning Repository [25]) so that results can be verified and the performance can be checked. A dataset usually has a train-

ing set and a test set. In such problems, the learning algorithm then becomes two-fold: to discover or learn different kinds of knowledge or rules from the training set, and apply these rules to the test set to measure the learnt model. However, many benchmark problems do not have a specific test set or some of them only have a small number of available instances in the dataset. To evaluate the performance of a classifier on these problems, it is necessary to use some resampling methods, such as n-fold cross-validation [32].

In n-fold cross-validation, a dataset is randomly partitioned into n folds (partitions) and the folds are near-equal size. In n-fold cross-validation, the folds are selected so that the proportion of instances from different classes remains the same in all folds. Subsequently, a single fold of the n folds is retained as the test set for testing the learnt classifier, and the remaining $(n-1)$ folds are used as the training set. The cross-validation process is then repeated n times, with each of the n folds used only once as the test set. The n results from the n experiments are then averaged to produce a single estimate of the classification performance. The advantage of such a method is that all instances are used for both training and testing, and each instance is used for testing only once. Generally, a larger n will produce an estimate with smaller bias because of the higher proportion of instances in the training set, but potentially higher variance (on top of being computationally expensive) [32]. Leave-one-out cross-validation (LOOCV) is an extreme case of n-fold cross-validation, which uses a single observation from the dataset as the test set, and the remaining instances as the training set. This is the same as a n-fold cross-validation with n being equal to the total number of instances in the dataset. Note that n-fold cross-validation is usually used when the number of instances in the entire dataset is small.

2.1.2 Data Representation

In classification, each instance in the dataset is presented to a classification algorithm using a representation system. The quality of the representation system is of key importance in the majority of classification algorithms. The most commonly used representation system is *feature-value* [6]. In this system, each instance is represented in the form of a vector of values for the features defined in the problem domain. The datasets (including both the training set and the test set) are usually represented in the form of a table, where each row is an instance and each column represents a different feature in the problem domain. The quality of the feature space defined in the problem domain, which usually involves the number of features and their relevance to the desired task, significantly influences the performance of a classification algorithm.

2.1.3 Classification Algorithms

Many different learning/classification algorithms (classifiers) have been proposed in machine learning. Four most commonly used classifiers, which will be used in this thesis, are reviewed in this section. They are K-nearest neighbour (KNN), decision tree classifiers (DT), support vector machines (SVMs), and Bayesian classifiers.

K-nearest Neighbour Classifier (KNN)

The K-nearest neighbour classifier (KNN) [33] is a type of instance-based learning algorithm. When using KNN for classification, it calculates the distances between an instance in the test set and every instance in the training set. KNN assigns the test instance to the class that is the most common amongst its k nearest neighbours, where k is a positive integer, typically small. If $k = 1$, the test instance is simply assigned to the class of the single nearest neighbour.

Euclidean distance, Manhattean distance, Minkowski Distance and other distance measures can be used to measure the distance between the test instance and the training instances in KNN [26]. In KNN, there is no explicit training process or it is very minimal. In other words, KNN does not use the training data points to do any generalisation. The training data in KNN is needed during the testing process, which is in contrast to other techniques like SVM, where the training set and all non-support vectors (hyperplanes) can be safely discarded.

KNN does not make any assumptions on the underlying data distribution. In real-world applications, most of the datasets do not obey the typical theoretical assumptions (e.g. Gaussian mixtures, linearly separable or independent features), which are needed in certain classifiers [26]. Therefore, KNN is a simple learning algorithm, but works well in practice. However, for a large training set, KNN requires large memory and is very time-consuming to make a decision [34].

Decision Tree Classifiers (DT)

Decision tree (DT) learning is an algorithm to approximate discrete-valued functions [28]. DT classifiers partition the input training data into smaller subsets by producing rules or decisions, also called nodes, which maximise the information gained [28].

A decision tree is usually learnt or built by recursively selecting the best feature/attribute to split the training data and expanding the leaf nodes of the tree until the stopping cirterion is met. The main problem in learning a decision tree is to determine which feature should be tested at each node of the tree. Most decision tree learning algorithms employ a top-down greedy search through the space of possible decision trees. The choice of the best split condition is determined by comparing the impurity of child nodes and also depends on which impurity measurement is used. Common used methods, such as classification and regression tree (CART) [35], the iterative dichotomiser 3 algorithm (ID3) [36], and the C4.5 algorithm

[37], employ an entropy function to measure the homogeneity of examples and choose the best node at each stage.

The learnt decision tree can be expressed as a set of 'if-then' decision rules to improve human readability. It is a hierarchy of nodes, where leaves represent the class labels and branches represent conjunctions of features that lead to those class labels. Instances in the test set are classified by sorting them down the tree from the root to certain leave nodes. For a given instance, the classification process starts at the root node by testing the value of the feature at the root node and then moves to one of the child nodes. Then the process is repeated for the subtree rooted at the new node.

Decision trees are easy to understand and interpret when the trees are small. People are able to understand decision tree models after a brief explanation. Important insights can be generated based on experts' domain knowledge and adding possible scenarios. A disadvantage of decision trees is that they are weak in separating non-rectangular areas in the input space [37], which creates two-way or multi-way feature interactions (See the challenges discussed in Section 1.2).

Support Vector Machines (SVMs)

Support vector machines (SVMs) are a popular machine learning method. They are based on the concept of decision planes that define decision boundaries. The main idea of SVMs is to use a kernel function to map the input data to a higher-dimensional space, where the instances are linearly separable. In the high-dimensional space, SVMs construct a hyperplane or a set of hyperplanes, which are used to create decision boundaries for classification [38]. SVMs are inherently two-class classifiers. Each hyperplane is expected to separate between a set of instances having two classes. Instances are classified based on what side of these hyperplanes they fall on. SVMs aim to maximise the distances between the hyperplanes and the nearest training data points of any class (so-called functional margin),

since in general the larger the margin the lower the generalisation error of the classifier [38].

SVMs were primarily designed for binary classification. Different methods have been developed to use SVMs for multiple (C) class classification [39]. A common way is to build C "one-versus-rest" classifiers (commonly referred to as "one-versus-all" classification) [40], and to choose the classifier that classifies the instances with the greatest margin. Another way is to build a set of "one-versus-one" (binary) classifiers [41], and to choose the class that is selected by the most classifiers.

A particular advantage of SVMs over other learning algorithms is that they are based on sound mathematics theory and can be analysed theoretically using concepts from the computational learning theory [42]. From a practical point of view, the most serious disadvantage of SVMs is the high algorithmic complexity and extensive memory requirements in large-scale tasks [43].

Bayesian classifiers

Bayesian classifiers are probabilistic methods for classification. Their assumptions are that the behaviour of data (input-output relationships) can be captured in probability distributions and features or attributes of the problem are statistically independent [28]. A Bayesian algorithm stores a simple probabilistic summary for each class and this summary contains the conditional probability of each feature or attribute value given the class, as well as the probability (or base rate) of the class [44].

Naïve Bayes (NB) classifiers are the most common and straightforward Bayesian classifiers [45]. It has been shown that NB classifiers are quite competitive with other classifiers, such as DT and neural networks (NN) [46]. NB classifiers make significant use of the assumption that all input features are conditionally independent, i.e. assuming that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class label.

An advantage of NB is that it only requires a small amount of training data to estimate the parameters (means and variances of the variables) necessary for classification. However, the assumption of features being conditional independent to each other can not be applied to many real-world problems, where there are interdependency between the input features.

2.2 Feature Selection

In classification, features/attributes are used to describe the instances in the datasets. Since useful features are usually unknown in advance, a large number of features are introduced to describe the instances in the datasets. Clearly, not all of the features are useful for classification. Irrelevant or redundant features may even reduce the classification performance [1]. Classification algorithms often suffer from the problem of "the curse of the dimensionality" [19] because of the large number of features in the dataset. Therefore, feature selection, also known as variable selection or attribute selection, is proposed as a data pre-processing step to reduce or eliminate irrelevant and redundant features.

Feature selection is defined by many researchers from different points of view, but most of them are similar in intuition and/or content [1]. The following lists those that are conceptually different and cover a range of definitions.

- Improving classification accuracy: feature selection is to choose a subset of features for improving the classification performance or reducing the complexity of the model without significantly decreasing the classification accuracy of the classifier built using only the selected features [47].

- Approximating original class distribution: feature selection is to select a subset of features such that the resulting class distribution, given only the values of the selected features, is as close as possible

to the original class distribution given by all the available features [47].

- Classical: feature selection is to select m features from n original features, $m < n$, such that the value of a criterion function is optimised over all subsets of size m [48].

- Idealised: feature selection is to find the minimally sized feature subset that is necessary and sufficient to describe the target concept [49].

Note that the second definition emphasises the class distribution of the training set, whereas the third definition emphasises on selecting the best combination of m features based on a certain criterion. Overall, feature selection is the process of finding a small subset of original features that is necessary and sufficient to solve a classification problem. Naturally, the optimal feature subset is the smallest subset that can obtain the highest classification performance, which makes feature selection a multi-objective problem [8, 20], i.e. to minimise the number of features and to maximise the classification performance.

Feature selection leads to dimensionality reduction by reducing or eliminating irrelevant and redundant features from the dataset, which in turn improves the classification performance and makes the learning and execution processes faster. Models constructed using a smaller number of features are typically easier to interpret and visualize [1, 19].

2.2.1 General Feature Selection Process

Generally, there are five basic steps in a typical feature selection algorithm [1] (see Figure 2.1).

1. A feature selection algorithm starts with a initialisation procedure.

   The initalisation procedure is the first step of a feature selection algorithm and it is based on all the original features in the problem. For

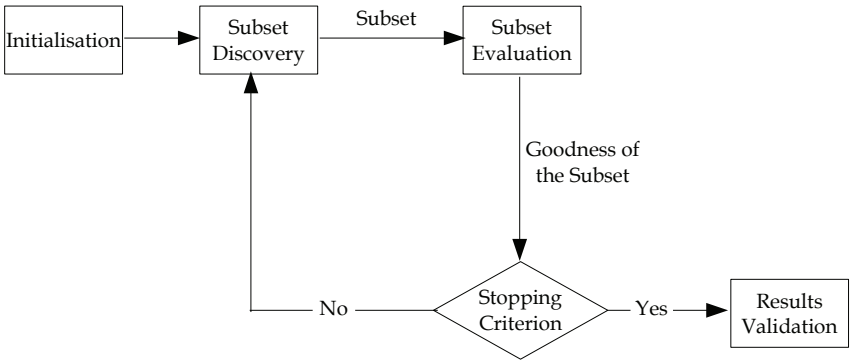

Figure 2.1: General feature selection process [1].

example, in a PSO based feature selection algorithm, the dimensionality of the search space is usually set as the total number of all the available features in this procedure.

2. A discovery procedure to generate candidate feature subsets.

   It is a search procedure [50], which can start with no features, all features, or a random subset of features. Many search techniques, including conventional methods and evolutionary techniques, are applied in this feature subset search step to search for the best subset of features.

3. An evaluation function to measure the feature subset.

   Feature subsets produced by the search procedure will be examined by a evaluation function to determine their goodness. The evaluation function plays an important role in a feature selection algorithm, because it helps guide the algorithm to search for the optimal feature subset.

4. Based on given criteria to decide when to stop.

   Stopping criteria can be based on the search procedure or the evaluation function. Criteria based on the search procedure can be whether

a predefined number of features are selected and whether a predetermined maximum number of iterations have been reached. Evaluation based criteria include whether addition or deletion of any feature does not produce a better subset and whether the optimal subset according to certain evaluation functions has been obtained. The loop continuous until the stopping criterion is satisfied.

5. A validation procedure to check whether the subset is valid.

   The validation procedure is not part of the feature selection process itself, but a feature selection algorithm must be validated. The selected feature subset will be validated on the test set. The results are compared with previously established results or the results of predefined benchmark techniques.

Two key factors in a feature selection algorithm are the *search strategy* and the *evaluation criterion*. The search space of a feature selection problem has 2^n possible points/solutions, where n is the number of available features. The algorithm explores the search space of different feature combinations to find the best feature subset. However, the size of the search space is huge, especially when the number of features is large. This is one of the main reasons making feature selection a challenging task. In many situations, it is impractical to search the whole space exhaustively [5]. Therefore, an efficient and effective global search technique is needed to find the optimal feature subset.

The evaluation criterion is used to form the evaluation function, which determines the goodness of the selected feature subset and leads the search of the algorithm. The optimal feature subset is always relative to a certain evaluation function. For a feature selection problem, the optimal feature subset chosen using one evaluation function may not be the same feature subset chosen using another evaluation function. Because of the feature interaction problem, the feature subset that can achieve the highest classification performance is usually a group of complementary features.

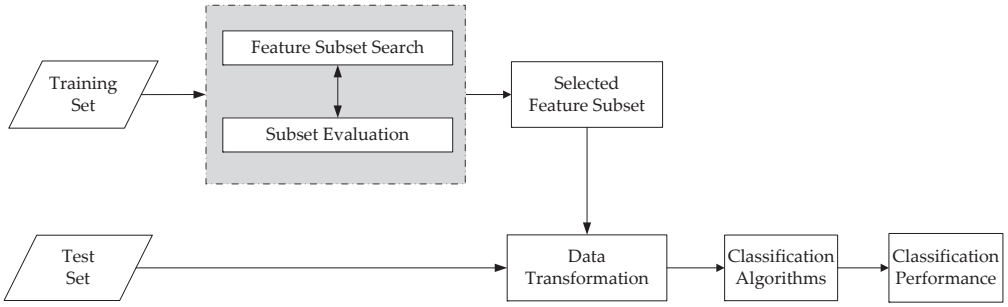

Figure 2.2: A filter feature selection algorithm in which the features are filtered independently the classification algorithm.

A good evaluation function is expected to guide the algorithm to search for such a complementary subset of features. Based on whether a learning/classification algorithm is used in the evaluation function, the existing feature selection methods can be broadly classified into two categories: filter approaches and wrapper approaches [50]. A filter feature selection algorithm is independent of any classification algorithm, whereas wrappers use a classification algorithm in the evaluation function.

2.2.2 Filter vs Wrapper Approaches

Figure 2.2 shows the diagram of a feature selection system taking a filter algorithm. In filter algorithms, the search process is independent of any classification algorithm. The goodness of feature subsets are evaluated based on a certain criterion like distance measure, information measure and consistency measure [1].

Figure 2.3 shows the diagram of a wrapper feature selection algorithm. In a wrapper model, the feature selection algorithm exists as a wrapper around a classification algorithm and the classification algorithm is used as a "black box" by the feature selection algorithm [51]. The performance of the classification algorithm is used in the evaluation function to evaluate the goodness of feature subsets and guide the search.

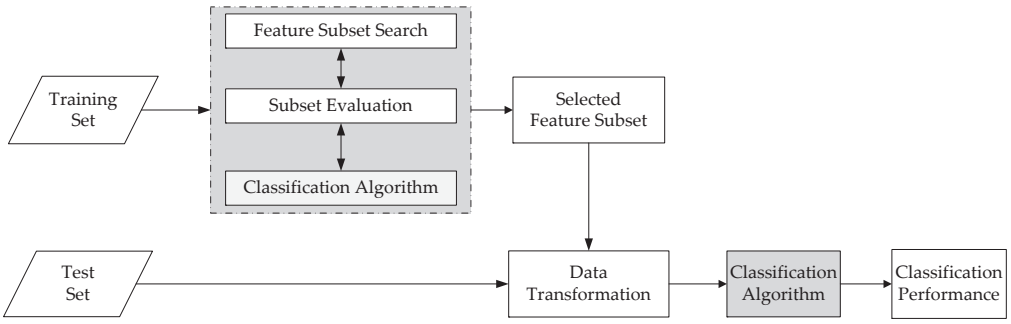

Figure 2.3: A wrapper feature selection algorithm which exists as a wrapper around the classification algorithm.

Filter algorithms are argued to be computationally less expensive and more general than wrapper algorithms [52, 51], but filter algorithms totally ignore the performance of the selected feature subset on the classification algorithm, which usually leads to lower performance than wrapper algorithms on a particular classification algorithm [51]. Compared with filter algorithms, wrappers usually produce better classification performance because of the interaction between the classification algorithm and the selected feature subsets during the feature selection process [5]. However, wrapper feature selection algorithms are usually computationally more expensive than filters because each evaluation of a candidate solution needs a learning/classification algorithm to be trained and tested [53].

2.2.3 Single Feature Ranking and Feature Construction

Single Feature Ranking

Single feature ranking is a relaxed version of feature selection [19]. Single feature ranking is computationally cheap, because it only requires the computation of the relative importance of the individual features and sub-

sequently sorting them [54]. In single feature ranking, a score denotes the relative importance of a single feature, which is measured by a predefined criterion. All the features are ranked according to the score and then feature selection can be accomplished by selecting a small number of top-ranked features. Normally, users specify the number of top-ranked features they need according to their requirements. There are also analytical methods to determine the best number of features [55].

Many measures have been proposed to evaluate the relative importance of each feature in single feature ranking algorithms, including information gain, gain ratio, mutual information and so forth [56]. Most single feature ranking methods fall into the filter approach category and not much work has been conducted on wrapper based single feature ranking [57, 58]. However, existing single feature ranking algorithms only measure the goodness of a single feature, not taking into account the interaction between groups of features [19, 59]. The combination of top-ranked features may still have redundancy. The combination of one or more top-ranked features and one or more low-ranked features are more likely to be complementary to each other and can typically achieve better classification performance. Therefore, in this thesis, single feature ranking for feature selection is not considered.

Feature Construction

In classification, the quality of representation of a dataset can also be improved by feature construction. Feature construction is a means of enhancing the quality of the feature space by constructing new high-level features [6, 60]. Different from selecting a subset of original features in feature selection, feature construction creates one or more new features to better describe the problem and reduce the complexity [61]. A constructed feature is usually a function of original low-level features and mathematical/logical operators. The constructed feature(s) should be able to discover the hidden relationship amongst the original low-level features to

increase the classification performance.

Based on whether a classification algorithm is included in the feature construction process, feature construction approaches (like feature selection methods) can be divided into two categories, which are wrapper approaches and filter approaches [6]. In wrapper approaches, feature construction and learning are integrated into one single algorithm, where new features are constructed within the learning process of the classification algorithm. However, wrapper feature construction approaches have a disadvantage of losing the generality and increasing the processing time in general [62]. In filter approaches, the process of feature construction is a separate, independent preprocessing stage and the new features are constructed before the classification algorithm is applied. Recently, more feature construction methods fall into this category than wrapper category because of computational efficiency and generality of filter approaches [62, 63, 64].

Feature construction can be used to reduce the dimensionality and increase the classification performance, but feature selection has an advantage of keeping the original features, which facilitates domain experts to understand and interpret of the problem. Therefore, this thesis will only focus on feature selection problems, which are important and challenging tasks needing further investigation.

2.3 Particle Swarm Optimisation (PSO)

2.3.1 Evolutionary Computation (EC)

Evolutionary computation (EC) is an area of computational intelligence, which is inspired by the principles of biological evolution. In general, EC consists of evolutionary algorithms (EAs), swarm intelligence (SI) and other techniques.

EC techniques often perform well approximating solutions in differ-

ent types of problems because they do not make any assumption about the underlying fitness landscape. Therefore, these techniques have shown successes in a variety of fields, ranging from practical applications in industry to leading-edge scientific research [9].

Evolutionary Algorithms

Evolutionary algorithms (EAs) are a subset of evolutionary computation, which are population based meta-heuristic optimisation algorithms. An EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection. In EAs, each candidate solution of the optimisation problem is represented as an individual in the population. The fitness function determines the goodness of each individual. Evolution of the population then takes place through the repeated application of selection and genetic operators, e.g. reproduction, mutation, and crossover. Four important EAs, which are genetic algorithms, genetic programming, evolutionary strategies and evolutionary programming, are briefly discussed here:

Genetic Algorithms. Genetic algorithms (GAs) [65] are evolutionary algorithms that simulate the process of natural selection, which are possibly the first algorithmic models developed to simulate genetic systems [66]. In GAs, candidate solutions of the problem are encoded as a population of chromosomes. A standard representation of each chromosome is as a fixed-length array of bits (bitstrings). The population is evolved to search for the optimal solution by applying genetic operators. The main driving operators of a GA is selection (to model survival of the fittest) and recombination through application of a crossover operator (to model reproduction). Compared to analytical optimisation methods like gradient based optimisation, GAs are less likely to be trapped in local optima. They, however, tend to be computationally expensive.

Genetic Programming. Genetic programming (GP) [67] was developed to evolve executable computer programs [68]. Similar to GAs, GP

concentrates on the evolution of genotypes. The main difference between the two paradigms is in the representation scheme. In GAs, each chromosome is represented as an array (or string). In GP, each individual is a computer program, which often uses a variable-length tree-like representation. A population of computer programs is optimised according to a fitness landscape, which determines a program's ability to perform a given task. GP has been a successful technique for getting computers to automatically solve problems without having to tell them explicitly how [9].

Evolutionary Strategies. Evolution strategies (ESs) [69] are in many ways very similar to GAs, but ESs consider both genotypic and phenotypic evolution and the emphasis is toward the phenotypic behavior of individuals [9]. In ESs, each individual is represented by a fixed-length real-valued vector. The vector includes the genetic building blocks and a set of strategy parameters. The strategy parameters simulate the behavior of that individual in its environment. Evolution then consists of evolving both the genetic characteristics and the strategy parameters, where the evolution of the genetic characteristics is controlled by the strategy parameters. Another difference between ESs and other EC algorithms is that ESs typically uses self-adaptive mutation rates. The selection of survivals in ESs is deterministic, that is, once the genetic operators are applied, a number of individuals with highest fitness are selected for the population in the next generation.

Evolutionary Programming. Evolutionary programming (EP) [70] is similar to GP, but the structure of the program is fixed. A population of chromosomes in EP is used to evolve finite-state machines (FSMs) [9]. Each FSM is in fact a program. A sequence of symbols that have been observed up to the current time is fed to each FSM. The fitness of an individual is evaluated by its ability in predicting future symbols. Like other EAs, EP uses fitness values to select individuals and then applies some evolutionary operators to find other solutions. Different from GAs, EP

applies two evolutionary operators, namely variation through application of mutation and selection operators. The recombination operators are not used with original EP.

Swarm Intelligence

Swarm intelligence (SI) is inspired by the collective intelligence of decentralised, self-organized systems [12] . A swarm is a population of interacting individuals that is able to optimise global objectives through collaborative search of the space. The intelligence relies on the networks of interactions among individuals, and between individuals and the environment. There is a general stochastic (or chaotic) tendency in a swarm for individuals to move toward a centre of mass in the population on critical dimensions, resulting in convergence on an optimum [12]. Two main techniques in swarm intelligence are particle swarm optimisation and ant colony optimisation.

Particle Swarm Optimisation. Particle swarm optimisation (PSO) [10] is a swarm intelligence algorithm inspired by the social behaviour of birds flocking or fish schooling [10]. In PSO, each candidate solution of the problem is represented as a particle, which is encoded by a vector or an array. Particles move in the search space to search for the optimal solutions. During the movement, each particle can remember its best experience. The whole swarm searches for the optimal solution by updating the position of each particle based on the best experience of its own and its neighbouring particles [12]. PSO is a simple but powerful search technique, which has been successfully applied to solve problems in a variety of areas [9, 71, 72, 73].

Ant Colony Optimisation. Ant colony optimisation (ACO) [74, 75] takes inspiration from the behaviour of real ants seeking the shortest path between their colony and a source of food [76]. Candidate solutions of the problem are represented as ants in the population. These ants deposit pheromone on the ground in order to mark their favourable path

that should be followed by other members of the colony. The best solution is the "path" that has the most pheromone. This mechanism allows the algorithms explicitly use the elements of previous solutions, which is the characteristic of ACO algorithm.

Besides, there are also some other popular EC methods, such as learning classifier systems (LCS) [77, 78], differential evolution (DE) [79, 80], artificial immune systems (AIS) [81, 82] and evolutionary multi-object optimisation algorithms (EMO) [83].

2.3.2 Standard Particle Swarm Optimisation (PSO)

PSO is an EC technique proposed by Kennedy and Eberhart in 1995 [10, 11]. PSO simulates the social behaviours such as birds flocking and fish schooling. In PSO, a population, also called a *swarm*, of candidate solutions are encoded as particles in the search space. PSO starts with the random initialisation of a population of particles. Particles move in the search space to search for the optimal solution by updating the position of each particle based on the experience of its own and its neighbouring particles. During the movement, the current position of particle i is represented by a vector $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$, where D is the dimensionality of the search space. The velocity of particle i is represented as $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previous position of a particle is recorded as the personal best called *pbest* and the best position obtained by the swarm so far is the global best called *gbest*. PSO searches for the optimal solution by updating the position and the velocity of each particle according to the following equations:

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_{1i} * (p_{id} - x_{id}^{t}) + c_2 * r_{2i} * (p_{gd} - x_{id}^{t}) \tag{2.2}$$

where t denotes the tth iteration in the evolutionary process. $d \in D$ denotes the dth dimension in the search space. w is inertia weight, which is

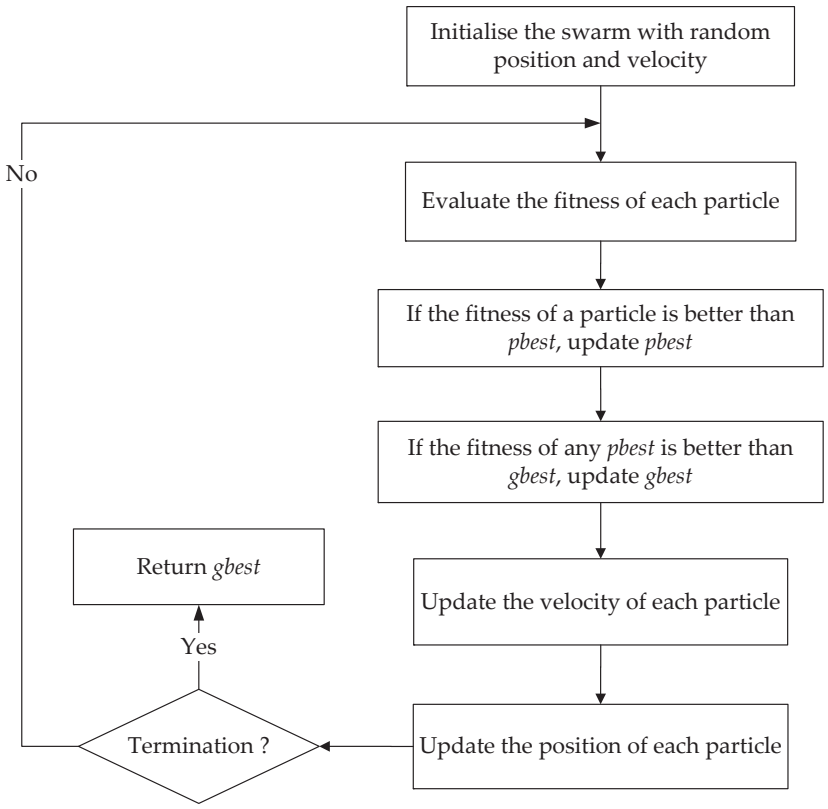

Figure 2.4: The flowchart of PSO

employed to control the impact of the previous velocities on the current velocity. c_1 and c_2 are acceleration constants. r_{1i} and r_{2i} are random values uniformly distributed in [0, 1]. p_{id} and p_{gd} represent the elements of *pbest* and *gbest* in the dth dimension. v_{id} is limited by a predefined maximum velocity, $v_{max,d}$ to $[-v_{max}, v_{max}]$ according to the following equation [9].

$$v_{id} = \begin{cases} v_{id}, \text{ if } |v_{id}| \leq v_{max,d} \\ v_{max,d}, \text{ if } v_{id} > v_{max,d} \\ -v_{max,d}, \text{ if } v_{id} < -v_{max,d} \end{cases} \tag{2.3}$$

Figure 2.4 and Algorithm 1 show the flowchart and the pseudo-code of PSO, where each particle is assumed to take the entire population as its

topological neighbours. First, each particle is initialised with a random velocity and a random position in a D-dimensional search space. Second, the fitness of each particle is evaluated by a predefined fitness function, and then based on $pbest$ and $gbest$, the velocity and the position of each particle are updated according to Equation 2.2 and 2.1 to search for the possible best solution. During the search process, if the fitness of the particle is better than that of $pbest$, then its position will be saved to replace/update the $pbest$. If the fitness of any $pbest$ in the population is better than $gbest$, the $gbest$ will be replaced by this $pbest$. The algorithm iteratively updates the position and velocity values of each particle to search for the best solution of the problem until a predefined stopping criterion is met. The stopping criterion can be a maximum number of iterations or a satisfactory fitness value.

2.3.3 Binary Particle Swarm Optimisation (BPSO)

PSO was originally proposed as an optimisation technique to address continuous problems. However, many optimisation problems, such as feature selection, occur in a space featuring discrete, where there are qualitative distinctions between variables and between levels of variables. To extend the use of the PSO algorithm, Kennedy and Eberhart [84] developed a binary particle swarm optimisation (BPSO) to solve discrete problems. The position of each particle is encoded by a binary string. The values in x_{id}, p_{id} and p_{gd} are restricted to 1 or 0. The velocity in BPSO represents the probability of an element in the position taking value 1. Equation (2.2) is still applied to update the velocity. A sigmoid function $s(v_{id})$ is introduced to transform v_{id} to the range of (0, 1). BPSO updates the position of each particle according to the following formulae:

$$x_{id} = \begin{cases} 1, \text{ if } rand() < s(v_{id}) \\ 0, \text{ o}therwise \end{cases} \tag{2.4}$$

Input : w: inertia weight; c_1, c_2: acceleration constants
v_{max}: maximum velocity; D: dimension of search space
$|Swarm|$: the population size
T: the maximum number of iterations
Output: $gbest$
best fitness value

```
1  begin
2      randomly initialise the position and velocity of each particle;
3      while T or other the stopping criterion is not met do
4          evaluate fitness of each particle;
5          for i=1 to |Swarm| do
6              if fitness of x_i is better than that of pbest_i then
7                  pbest_i = x_i ;        // Update the pbest of particle i
8              if fitness of pbest_i is better than that of gbest then
9                  gbest = pbest_i ;      // Update the gbest of particle i
10         for i=1 to |Swarm| do
11             for d=1 to D do
12                 update the velocity of particle i according to Equation 2.2;
13                 update the position of particle i according to Equation 2.1;
14     return gbest and its fitness value;
```

Algorithm 1: Pseudo-code of PSO

where

$$s(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{2.5}$$

where $rand()$ are numbers randomly selected from a uniform distribution in [0,1].

2.3.4 Initialisation and Updating Mechanisms in PSO

In PSO, the initialisation and updating mechanisms are two of the important aspects that can significantly influence its performance. Recently,

many initialisation strategies and updating mechanisms for $pbest$ and $gbest$ have been proposed in PSO to improve its performance.

Richards and Ventura [85] proposed an initialisation strategy for PSO based on centroidal Voronoi tessellations (CVTs) [86], which is a mathematical way of dividing space into a number of regions to initialise the particles' positions using evenly distributed points to cover the search space. Comparisons on eight benchmark functions showed that the CVTs based initialisation strategy can improve the performance of PSO when the dimensionality is 50.

Parsopoulos and Vrahatis [87] applied the Nonlinear Simplex method (NSM) [88] to generate initial particles in PSO. Experiments showed that the NSM based initialisation can improve the performance of PSO on 14 benchmark functions. However, some parameters of PSO used in the experiments are different from common settings, such as $c_1 = c_2 = 5$, which are usually set as values between 1 and 2 [10, 89]. This is a possible reason why the standard PSO performed poorly. Since NSM was slow and can be applied only to the problems with low dimensionality, this initialisation method may not be appropriate for feature selection problems, where the dimensionality is typically large.

Jabeen et al. [90] proposed an opposition based initialisation strategy in PSO. In this initialisation strategy, a population of random particles was firstly generated and the opposition of each particle was calculated according to the opposition based learning. Particles in both the randomly generated population and its opposition population were treated as candidate particles. The fitness of all the candidate particles was evaluated and the fitter ones were selected as the initial population of PSO. The experiments showed that the proposed opposition based PSO achieves better performance than the random initialisation method. Later, Wang et al. [91] proposed an initialisation method based on space transformation search (STS) strategy by evaluating the solutions in both the original and the transformed spaces to get a better set of initial particles. Experimental

results showed that the proposed initialisation strategy outperformed the traditional random initialisation and the opposition-based population initialisation. Gutierrez et al. [22] assumed that uniformly distributed particles in the initialisation was able to improve the performance of PSO. Three different initialisation strategies, the orthogonal array initialisation, a chaotic technique and the opposition-based initialisation in PSO were compared on problems with high dimensional search space. The experiments showed that the three initialisation strategies performed differently on different problems. However, all these strategies are general methods and no existing initialisation strategies are specifically proposed for feature selection problems.

Some researchers also work on developing new $gbest$ updating mechanisms. Wang et al. [92] applied a dynamic Cauchy mutation to $gbest$, where if the new $gbest$ was better after the application of the mutation operator then it was replaced by the mutated $gbest$. However, the proposed algorithm does not work well on multi-modal problems and it may also not perform well for feature selection problems, which have many local optima in the search space. Chuang et al. [93] proposed a $gbest$ updating mechanism, but it simply set $gbest$ to zero and could not be applied to $pbest$.

Chen et al. [94] proposed a preferential velocity-updating mechanism in PSO to avoid premature convergence, where the particles far away from $gbest$ were updated with more preference on the second term in velocity updating (Equation 2.2) and less preference on the third term. On the other hand, the particles close to $gbest$ were updated with more preference on the third term and less preference on the second term. The experiments showed that this algorithm achieved better performance than other conventional PSO variants on 14 benchmark functions. Qi and Ding [95] adapted an updating mechanism to PSO to avoid the problem of premature convergence. In this updating mechanism, a random dimension of each $pbest$ was re-set if a convergence measure was lower than a pre-

defined threshold. In six of the eleven tested cases, this PSO algorithm outperforms other methods used in the experiments.

Existing works have shown that the performance of PSO can be improved by developing new initialisation and updating mechanisms for $gbest$ and $pbest$. However, there has been no existing work that proposes an initialisation or a $gbest$ and $pbest$ updating mechanism in PSO specifically for feature selection to date.

2.4 Multi-Objective Optimisation

Multi-objective problems happen wherever optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives [96]. Multi-objective optimisation involves minimising or maximising multiple conflicting objective functions. In mathematical terms, the formulae of a minimisation problem with multiple objective functions can be written as follows:

$$minimise \;\; F(x) = [f_1(x), f_2(x), \; ... \; , f_K(x)] \tag{2.6}$$

subject to:

$$g_i(x) \leq 0, \; i = 1, 2, \; ... \; m \tag{2.7}$$

$$h_i(x) = 0, \; i = 1, 2, \; ... \; l \tag{2.8}$$

where x is the vector of decision variables, $f_i(x)$ is a function of x, K is the number of objective functions to be minimised, $g_i(x)$ and $h_i(x)$ are the constraint functions of the problem. m and l are integer numbers.

In multi-objective optimisation, the quality of a solution is explained in terms of trade-offs between conflicting objectives. Let y and z be two solutions of the above K-objective minimisation problem. If the following conditions are met, one can say y dominates z or y is better than z:

$$\forall i : f_i(y) \leq f_i(z) \quad and \quad \exists j : f_j(y) < f_j(z) \tag{2.9}$$

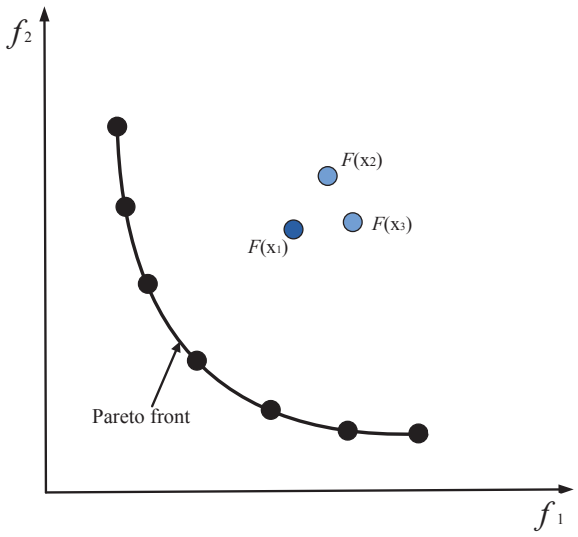

Figure 2.5: A minimisation problem with two objective functions.

where $i, j \in \{1, 2, 3, ...K\}$.

Take a two-objective minimisation problem (shown in Figure 2.5) as an example, x_1 dominates both x_2 and x_3. For the case that neither x_2 dominates x_3 nor x_3 dominates x_2, x_2 and x_3 are called non-dominated solutions or trade-off solutions of each other. When a solution is not dominated by any other solutions, it is referred as a Pareto-optimal solution. The set of all Pareto-optimal solutions forms the trade-off surface in the search space, called Pareto front.

Feature selection has two main conflicting objectives, which are minimising both the number of features and the classification error rate. Therefore, feature selection can be expressed as a two-objective minimisation problem.

2.4.1 Evolutionary Multi-Objective Optimisation

Evolutionary computation techniques are particularly suitable for multi-objective optimisation because they use a population of candidate solu-

tions and are able to find multiple non-dominated solutions in a single run. Therefore, evolutionary computation techniques have recently gained more attention than many other techniques to address multi-objective problems. Evolutionary multi-objective algorithms (EMO) can be generally classified into three main types depending on how the objectives are treated in the evolutionary search process, which are a *priori*, a *posteriori* and *progressive* objective articulation [83].

Priori articulation

In a priori articulation, before the search process starts, the multiple objectives of the problem are combined or aggregated together according to pre-defined objective preference. The evolutionary process is guided by the combined objectives and returns a single solution, which is expected to be optimal according to this objective prioritisation. The focuses of this type of algorithms are how to combine or aggregate the objectives, and how to determine the relative importance prior to the search.

The simplest and most common method is to use a linear aggregating function to combine the multiple objectives into a single fitness function. A weighting factor is usually assigned to each objective to specify its relative importance. Take a K-objective minimisation problem as an example, the linearly aggregated fitness function can be expressed by Equation 2.10.

$$minimise \;\; F(x) = minimise \sum_{k=1}^{K} w_k f(x)_k \tag{2.10}$$

where w_k represents the weighted relative importance for the kth objective. Usually, $0 \leq w_k \leq 1$ and $\sum w_k = 1$.

The priori articulation is simple and easy to implement and has been used in many existing works, including feature selection, where there are two objectives of minimising the classification error rate and the number of features [97, 98]. However, a major limitation of this technique is that the relative importance of each objective has to be specified before the optimi-

sation process starts. In many real-world problems, it is not a trivial task to determine the suitable values of the weight factors. Bad combination of the objectives leads to poor solutions and in many cases, the objectives cannot be linearly interrelated. Another major limitation is that only one solution is returned, which means that any change to the objective preference needs a new optimisation process to obtain the desired solution.

Posteriori articulation

Posteriori articulation approaches assume that the objectives cannot be interrelated with each other. Each objective is treated separately in the optimisation process. They focus on finding a set of the trade-off solutions along the objectives, i.e. the Pareto front. The users therefore have multiple choices and they can choose any solution from the Pareto front according to their own requirements.

As stated previously, prior articulation methods have two major limitations. Most EMO algorithms belong to a posteriori approach [83]. The search of such EMO algorithms aims to push the frontier of trade-off solutions (Pareto front) closer toward a point which is optimal on all objectives.

This requires two major adaptations in EMO compared to a single objective EC algorithm. The first one is to adapt the evolutionary search algorithm to evolve a set of non-dominated solutions in parallel, where the Pareto front is the output. In contrast, a single objective EC algorithm focuses on finding only a single best solution and only a single solution is returned. The second one is to incorporate the concept of Pareto dominance in the fitness evaluation, which measures the performance of each individual based on all the objectives and relative to all others in the population.

Progressive articulation

Progressive articulation approaches integrate a posteriori EMO search with objective preference during the optimisation process in an iterative and in-

teractive way [83]. Progressive articulation has been found very useful in *many*-objective (more than 2 objectives) problems [99].

This thesis uses concepts from a priori and a posteriori objective articulation. More detailed discussions on EMO approaches can be seen in [83, 9, 100, 101].

2.4.2 Typical Evolutionary Multi-Objective Algorithms

In recent years, a large number of EMO algorithms have been proposed, which are posteriori algorithms and aim to evolve a set of Pareto non-dominated solutions [83, 9]. All these algorithms use a Pareto-based fitness evaluation scheme, which treats each objective as a separate entity during the evolutionary search process. Typical EMO algorithms will be reviewed in this section.

Non-dominated sorting based multi-objective genetic algorithm II (NSGAII). NSGAII is one of the most popular evolutionary multi-objective algorithms proposed by Deb et al. [23]. The main principle of NSGAII is the use of a fast non-dominated sorting technique and a diversity preservation strategy. The fast non-dominated sorting technique is used to rank the parent and offspring populations to different levels of non-dominated solution fronts. A density estimation based on the crowding distance is adopted to keep the diversity of the population. More details can be seen in the literature [23].

Strength Pareto evolutionary algorithm 2 (SPEA2). SPEA2 is a popular evolutionary multi-objective algorithm proposed by Zitzler et al. [24]. The main principle is the fine-gained fitness assignment strategy and the use of an archive truncation method. In SPEA2, the fitness of each individual is the sum of its strength raw fitness and a density estimation. A new population is constructed by the non-dominated solutions in both the original population and the archive. When the number of non-dominated

solutions is larger than the population size, the archive truncation method is applied to determine whether a non-dominated solution should be selected or not according to the distance to its *k*th nearest neighbour. More details can be seen in the literature [24].

Pareto archived evolution strategy (PAES). PAES is an evolutionary multi-objective algorithm proposed by Knowles and Corne [102]. The authours claimed that PAES may represent the simplest possible non-trival algorithm capable of generating diverse solutions in the Pareto front. The main idea of PAES is the use of a local search and the use of an archive of previously found non-dominated solutions. PAES was proposed as a baseline approach for Pareto multi-objective algorithms. More details about PAES can be seen in the literature [102].

Multi-objective PSO (MOPSO). PSO has been widely used for multi-objective optimisation [100, 83, 9]. The key issue in using PSO for multi-objective optimisation is how to determine a $gbest$ for each particle since there is no longer a single better solution, but a set of non-dominated solutions. Researchers have proposed many different ways to address this problem [100]. Two of them are non-dominated sorting multi-objective PSO (named NSPSO) [103] and multi-objective PSO based on crowding, mutation and ϵ-dominance (named CMDPSO) [104]. NSPSO is based on the idea of non-dominated sorting mechanism in NSGAII, where the non-dominated solutions are kept in the swarm. The non-dominated solutions are ranked according to a crowding factor and the $gbest$ is randomly selected from the top-ranked solutions. CMDPSO employs an archive to store the non-dominated solutions. A binary tournament selection and a crowding factor are used together to select a $gebst$ for a particle. More details about NSPSO and CMDPSO can be seen in [103, 104].

Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D). MOEA/D is another popular EMO algorithm proposed by Zhang and Li [105] in 2007. MOEA/D is based on decomposition, which is a basic strategy in traditional multi-objective optimisation. In MOEA/D, a multi-objective optimisation problem is decomposed into a number of scalar optimisation sub-problems. Each sub-problem is optimised by only using information from its neighboring sub-problems. All the sub-problems are optimised simultaneously. This lowers the computational complexity over NSGAII, but still maintains a powerful search ability. However, MOEA/D involves a decomposition process, which is not particularly suitable for the feature selection task in this thesis. More details about MOEA/D can be seen in [105].

2.5 Information Theory

Information theory provides a way to measure the information of the random variables [106]. Information theory can be viewed as a branch of mathematics and it is also related to electrical engineering, bioinformatics, and computer science [107]. Since the fundamental premises of information theory were proposed by Shannon [106] in 1949, it has gained attention from almost every field of science and technology [108].

Information theory provides different ways to quantify uncertainty. We only review the concepts of entropy and mutual information briefly since they are used in this thesis. More details about information theory can be seen from [107, 108]

2.5.1 Entropy and Mutual Information

The entropy is a measure of the uncertainty of random variables. Let X be a random variable with discrete values, its uncertainty can be measured by entropy $H(X)$, which is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{2.11}$$

where $p(x) = Pr(X = x)$ is the probability density function of X. Note that entropy does not depend on actual values, just the probability distribution of the random variable.

For two discrete random variables X and Y with their probability density function $p(x, y)$, the joint entropy $H(X, Y)$ is defined as

$$H(X, Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log_2 p(x, y) \tag{2.12}$$

When a certain variable is known and others are unknown, the remaining uncertainty is measured by the conditional entropy. Assume that variable Y is given, the conditional entropy $H(X|Y)$ of X with respect to Y is shown by Equation 2.13.

$$H(X|Y) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log_2 p(x|y) \tag{2.13}$$

where $p(x|y)$ is the posterior probabilities of X given Y. From this definition, if X completely depends on Y, then $H(X|Y)$ is zero, which means that no more other information is required to describe X when Y is known. On the other hand, $H(X|Y) = H(X)$ denotes that knowing Y will do nothing to observe X, i.e. they are fully independent or unrelated.

The information shared between two random variables is defined as mutual information. Given variable X, how much information one can gain about variable Y, which is mutual information $I(X; Y)$.

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log_2 \frac{p(x, y)}{p(x)p(y)} \end{aligned} \tag{2.14}$$

According to Equation 2.14, the mutual information $I(X;Y)$ will be large if two variables X and Y are closely related. Otherwise, $I(X;Y) = 0$ if X and Y are totally unrelated.

Information theory, mainly mutual information, has been applied to filter feature selection to measure the relationship between the selected features and the class labels. Typical methods will be reviewed in the next section.

2.6 Traditional Methods for Feature Selection

This section reviews typical traditional feature selection methods some of which will be used in this thesis to compare with the newly developed algorithms.

2.6.1 Wrapper Feature Selection Approaches

Generally, wrapper feature selection algorithms are usually computationally more expensive than filters because each evaluation involves a training process and a testing process of the classification algorithm [53]. Meanwhile, since the search space of a feature selection problem with n features has 2^n possible points, it is usually impossible to search the whole search space exhaustively. Therefore, most of the existing wrappers employ greedy or stochastic search strategies [4].

Sequential forward selection (SFS) [109] and sequential backward selection (SBS) [110] are two commonly used wrapper feature selection algorithms. Both of them use a greedy hill-climbing search strategy to search for the optimal feature subset. SFS starts with an empty set of features and iteratively adds one feature at one time until no improvement in classification accuracy can be achieved. By contrast, SBS sequentially removes features from a full candidate feature subset until the further removal of any feature does not increase the classification accuracy. However, both

SFS and SBS suffer from the so-called nesting effect, which means that once a feature is selected (discarded) it cannot be discarded (selected) later. Therefore, both SFS and SBS are easily trapped in local optima [5]. In addition, both SFS and SBS require long computational time when the number of features is large [5].

In order to avoid nesting effect, Stearns [111] proposed a "plus-l-take away-r" method in which SFS was applied l times forward and then SBS was applied for r back tracking steps. However, determining the best values of (l, r) is a challenging task. In order to solve this problem, Pudil et al. [112] proposed two floating selection methods, sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS) to automatically determine the values of (l, r). In addition, the values of (l, r) in SBFS and SFFS that denotes the number of forward and backtracking steps are dynamically controlled instead of being fixed in the "plus-l-take away-r" method. Although the floating methods are claimed to be at least as good as the best sequential method, they are still likely to become trapped in a local optimal solution even the criterion function is monotonic and the scale of the problem is small [113].

Based on the best-first algorithm and SFFS, Gutlein et al. [114] proposed a linear forward selection (LFS) in which the number of features considered in each step was restricted. Because of the small number of features used for evaluations in each step, LFS improves the computational efficiency of sequential forward methods while maintaining comparable accuracy of the selected feature subset. However, LSF starts with ranking all the individual features without considering the presence or absence of some other features, which in turn limits the performance of the LSF algorithm in problems where there are interactions between features.

Recently, evolutionary computation techniques have been applied to wrapper feature selection models, such as PSO [115], GAs [8], GP [116], and ACO [117]. Typical methods will be reviewed in Section 2.7.

2.6.2 Filter Feature Selection Approaches

A filter feature selection algorithm searches for the optimal feature subset in the search space based on a certain evaluation criterion, which is independent of any learning/classification algorithm.

Different criteria, including distance measures [118], dependency measures [119], consistency measures [120], and information measures [121], have been applied to develop filter feature selection algorithms. Besides the evaluation criterion, how to search for the best feature subset is another important factor in feature selection methods. Among the existing feature selection algorithms, two classical filter based methods are FOCUS [122, 123] and Relief [124]. The FOCUS algorithm was originally defined for noise-free Boolean domains [123]. It starts with an empty feature subset and exhaustively examines all subsets of features and then selects the minimal subset of features that is sufficient to determine the class labels for all instances in the training set. However, the FOCUS algorithm performs an exhaustive search to find the best feature subset, which is computationally expensive.

The Relief algorithm is another popular filter feature selection method that assigns a relevance weight to each feature [124]. The weight is intended to denote the relevance of the feature to the target concept. Relief samples instances randomly from the training set and updates the relevance values based on the difference between the selected instance and the two nearest instances of the same and opposite class (the "near-hit" and "near-miss"). However, the Relief algorithm does not deal with redundant features, because it attempts to find all relevant features regardless of the redundancy between them [125], which is referred as feature interaction, a challenge in feature selection tasks.

Decision trees use only relevant features that are needed to completely classify the training set and remove all other features. Cardie [126] proposed a filter based feature selection algorithm that used a decision tree algorithm to select a subset of features for a nearest neighbourhood algo-

rithm. Experiments showed that the feature subset generated by a decision tree helped the nearest neighbour algorithm to reduce its classification error rate.

Yu and Liu [119] claimed that feature relevance alone was insufficient for efficient feature selection of high-dimensional data. They proposed a feature selection algorithm that took both relevance and redundancy into account. The algorithm, however, is limited to problems that only have discrete features.

Mutual Information for Filter Feature Selection

Since mutual information are capable to evaluate the relationship between variables, they have been applied to feature selection to measure the relationship between the selected features and the class labels.

Hall [127] proposed a correlation based filter feature selection method (Cfs), which uses mutual information to evaluate the correlation between the features and the class labels to evaluate the goodness of the selected features. Kwak and Choi [128] developed a greedy search based feature selection method, where mutual information was used to evaluate the goodness of the selected features. The algorithm stopped when a desired number of features was reached. There are also some other filter methods using mutual information, but most of them suffer from two problems [129]. The first one is that they need a predefined weighting parameter to balance the relative importance of the relevance (reflecting the classification performance) and the redundancy (reflecting the number of features) of the selected feature subset, which is usually difficult to determine. The second one is that the redundancy was shown by the mutual information between two features and the class labels was not considered. Because of feature interaction, two correlated features may become complementary to each other when considering the class labels [129].

To address these problems, Peng et al. [130] combined the use of mutual information (filter) with a wrapper method that considers the class la-

bels. A feature selection evaluation criterion named minimal-redundancy-maximal-relevance criterion (mRMR) was first developed, where mutual information was also used to measure the *relevance* and the *redundancy* of the selected features. Based on mRMR, a two-stage algorithm by combining the mRMR with other more sophisticated feature selectors (e.g., wrappers) was developed and successfully selected a small number of features and maintained or increased the classification performance. To avoid the determination of the weighting parameter, Foithong et al. [129] also combined the mutual information based criterion with a wrapper method, where Multilayer perceptron (MLP) [131] with a single hidden layer was trained by using the back-propagation algorithm to evaluate the goodness of the feature subsets. Later, Liu et al. [132] developed a feature selection method based on dynamic mutual information, where the mutual information of each candidate feature was re-calculated on unlabelled instances, rather than the whole sampling space. These existing works have shown that the concept of mutual information can be used for feature selection, but it has never been applied together with EC based algorithms for feature selection.

Different evolutionary computation techniques have been applied to develop filter feature selection algorithms and typical algorithms will be reviewed in Section 2.7.

2.7 EC Techniques for Feature Selection

Recently, different evolutionary computation techniques have been applied to develop feature selection algorithms, such as PSO [133], GAs [134], GP [135], and ACO [136, 137]. Since the focus of this thesis is to develop new PSO based approaches to feature selection, we review PSO related work first, then other EC related work for feature selection.

2.7.1 PSO for Feature Selection

There are two versions of PSO, which are the original continuous PSO and BPSO. Both of them have been applied to feature selection. Generally, when a continuous PSO algorithm is applied to feature selection problems, the dimensionality of the search space is n, where n is the total number of available features in the dataset. Each particle in the swarm is encoded using a vector of n real numbers. The position of particle i in the dth dimension, x_{id}, is usually in interval [0, 1]. In order to determine whether a feature will be selected or not, a threshold $0 < \theta < 1$ is needed to compare with the real numbers in the position vector. If $x_{id} > \theta$, then the corresponding feature d will be selected. Otherwise, feature d will be abandoned. When using BPSO to solve feature selection problems [13, 16], the representation of a particle is a n-bit binary string. The position of each particle is Boolean, where "1" represents that the feature will be selected and "0" otherwise.

PSO has recently gained more attention to solve feature selection problems. Many PSO based feature selection algorithms have been proposed, which include both wrapper approaches and filter approaches.

PSO Based Wrapper Feature Selection

Azevedo et al. [138] proposed a wrapper feature selection algorithm using PSO and SVM for personal identification in a keystroke dynamic system. Experiments showed that the proposed algorithm produces better performance than a GA with SVM model. However, the proposed algorithm obtained a relatively high false acceptance rate, which should be low in most identification systems. Marinakis et al. [139] proposed a wrapper algorithm based on BPSO and KNN for a real-world medical diagnosis problem called Pap-smear cell classification. The results showed that this method removed around half of the features and achieved good classification performance.

Based on PSO and a linear discriminant analysis algorithm (LDA), Lin and Chen [140] proposed a wrapper feature selection algorithm named PSOLDA. PSOLDA aimed to maximise the classification performance evaluated by LDA. Different parameters were tuned to obtain the best settings for PSOLDA. Experimental results showed that PSOLDA outperformed LDA using all features, LDA with principal components analysis (PCA), and LDA with either forward or backward selection in almost all cases. However, PSOLDA is sensitive to parameter settings and the datasets in the experiments have a small number of features.

Lin et al. [115] proposed a wrapper feature selection algorithm, which was based on PSO and SVM. The difference from the method in [138] was that this method could optimise the parameters in SVM and search for the best feature subset simultaneously. Huang and Dun [141] also proposed a wrapper algorithm for feature selection and parameter optimisation in a SVM using both continuous PSO and BPSO. In the proposed algorithm, each particle was encoded by two parts, where the first part represents the features in a dataset, which are optimised by BPSO, and the second part is the parameters in SVM, which are evolved by continuous PSO. Experiments showed that the proposed algorithm could determine the parameters and search for the optimal feature subset simultaneously, and also achieve high classification accuracy. However, only one dataset with a small number of features was used in the experiments, which is not enough to verify the performance of the proposed algorithm. Mohemmed et al. [15] proposed a hybrid method (PSOAdaBoost), which incorporated PSO with an AdaBoost framework [142] for face detection. The proposed PSOAdaBoost algorithm aimed to search for the best feature subset and determine the decision thresholds of AdaBoost simultaneously, which could also speed up the process of the training and increase the accuracy of weak classifiers in AdaBoost.

Inertia weight can improve the performance of PSO by properly balancing its local search and global search. Yang et al. [52] proposed two

strategies to determine the inertia weight of BPSO. Experiments on a wrapper feature selection model suggested that the two proposed BPSOs outperformed other methods, including SFS, "plus-l-take away-r" method, SFFS, sequential GA and different hybrid GAs.

In order to avoid the particles converging at local optima, Yang et al. [16] proposed a strategy to reset the $gbest$ during the search process to keep the diversity of the population in BPSO. In the proposed algorithm, when $gbest$ was identical after three iterations, a Boolean operator 'and(.)' would 'and' each bit of the $pbest$ of all particles in an attempt to create a new $gbest$. Experimental results illustrated that the proposed method usually achieved higher classification accuracy with fewer features than GA and standard BPSO. Chuang et al. [93] also developed a strategy for $gbest$ in BPSO for feature selection in which $gbest$ would be reset to zero if it maintained the same value after several iterations. Experiments with cancer-related human gene expression datasets showed that the proposed BPSO algorithm outperformed the algorithm proposed by Yang et al. [16] in most cases. However, the proposed algorithm was only compared with one traditional method in terms of the classification performance. No PSO or EC based algorithms have been used for comparisons. Chuang et al. [143] applied the so-called catfish effect to BPSO for feature selection, which was to introduce new particles into the swarm by re-initialising the worst particles when $gbest$ has not improved for a number of iterations. The authors claimed that the introduced catfish particles could help PSO avoid premature convergence and lead to better results than sequential GA, SFS, and SFFS.

The use of the $gbest$ resetting strategy in PSO for feature selection was also investigated by Vieira et al. [144], who proposed two improved BPSO for wrapper feature selection. To avoid premature convergence, the first algorithm employed mutation operators and a $gbest$ reseting strategy, which randomly reset $gbest$ using a feature subset including only one feature. The second algorithm combined the same $gbest$ reseting strategy with local

search and a maximum velocity control strategy. Results suggested that the proposed methods outperformed two other PSO based algorithms, and achieved similar classification performance to GA, but used less computational time and selected fewer features. The results also showed that the $gbest$ resetting strategy helped BPSO avoid the problem of premature convergence. However, the experiments were only conducted on one dataset and the number of features is relatively small.

Alba et al. [145] combined a geometric BPSO with a SVM algorithm for feature selection, 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. Experiments on high dimensional microarray problems showed that the proposed algorithm could achieve slightly higher accuracy than a GA with SVM in most cases. Meanwhile, experiments also showed that the initialisation of the BPSO had a great influence in the performance since it introduced an early subset of acceptable solutions in the evolution process. Talbi et al. [146] also proposed a geometric BPSO and compare it with a GA using SVM for feature selection in high dimensional microarray data. They concluded that the performance of the proposed BPSO was superior to a GA in terms of the classification accuracy.

Unler and Murat [13] modified the standard BPSO by extending social learning to update the velocity of the particles. Meanwhile, an adaptive feature subset selection strategy was developed, where the features were selected not only according to the likelihood calculated by BPSO, but also according to their contribution to the subset of features already selected. The improved BPSO was applied to a wrapper feature selection model for binary classification problems. Experimental results indicated that the proposed BPSO method outperformed tabu search and scatter search algorithms.

Liu et al. [14] proposed a multiple swarm BPSO (MSPSO) to search for the best feature subset and optimise the parameters of SVM. Experimental

results showed that the proposed feature selection methods could achieve higher classification accuracy with a smaller subset of features than grid search, standard BPSO and a GA. However, the proposed MSPSO was computationally more expensive than the other three methods because of the large population size and complicated communication rules between different sub-swarms. Fdhila et al. [98] also developed a distributed PSO algorithm with a number of sub-swarms searching for the optimal feature selection, where KNN with $K = 1$ was used to test the classification performance. The number of features and the classification error rate were combined into a single fitness function. The proposed algorithm achieved a set of solutions by keeping the good solutions in different sub-swarm. However, the computational cost of the proposed algorithm is also high because it involves parallel evolutionary processes and multiple sub-swarms with a relative large number of particles.

PSO Based Filter Feature Selection

Wang et al. [133] proposed an improved BPSO by defining the velocity as the number of elements that should be changed. The performance of the improved BPSO was compared with that of a GA in a filter feature selection model based on rough sets theory [147]. Experimental results showed that the improved BPSO was computationally less expensive than a GA in terms of both memory and running time. However, the classification performance of the feature subset is only tested on one learning algorithm, the LEM2 algorithm, which has some bias for rough set based algorithms.

Chakraborty [148] compared the performance of BPSO with that of a GA in a filter feature selection algorithm with a fuzzy sets [149] based fitness function. The results showed that BPSO performed better than a GA in terms of the classification accuracy.

Based on BPSO, Iswandy and Koenig [97] developed a filter based feature selection algorithm, where they used different weights to linearly combine three objectives, which were evaluated by three filter criteria, into

a single fitness function. Experimental results showed that the proposed algorithm outperformed other methods on several benchmark problems. However, although the proposed algorithm used multiple evaluation criteria, it aimed to select only one feature subset instead of a set of non-dominated solutions.

Esseghir et al. [150] proposed a filter-wrapper feature selection method based on PSO, which aimed to integrate the strengths of both filters and wrappers. The proposed filter-wrapper scheme encoded the position of each particle in PSO with filter scores of features, which reflected feature-class dependency levels, and then PSO was applied to adjust the scores to search for the best feature subset. The positive score meant the corresponding feature was selected, otherwise it was abandoned. The fitness of each particle was the classification accuracy achieved by a KNN classifier with the selected features. The results showed that the proposed method could achieve slightly better performance than a BPSO based filter algorithm. However, the performance of the proposed algorithm has not been compared with any wrapper algorithm, which can usually obtain higher classification performance than a filter algorithm.

Overall, existing work has shown that PSO has the potential to address feature selection problems. Almost all PSO based feature selection approaches are developed in recent years (after 2007). There are more PSO based wrapper approaches than filter approaches. Meanwhile, PSO has only been used for single objective feature selection and no work has been conducted for multi-objective feature selection. Therefore, it is needed to further investigate the potential of PSO for feature selection.

2.7.2 Other EC Techniques for Feature Selection

Besides PSO, many other EC algorithms have also been applied to feature selection problems such as GAs [134], GP [135], and ACO [151]. Typical methods will be briefed in this section.

Genetic Algorithms (GAs) for Feature Selection

Generally, in a GA based feature selection algorithm, each individual (chromosome) in the population represents a subset of features. Each chromosome is encoded by a n-bit binary string for a n-dimensional feature search space. The bit with value "1" indicates the feature is selected in the subset, and "0" otherwise. Crossover, mutation and reproduction operators are applied in the algorithm to search for the optimal subset of features [134]. GAs have been applied to both filter and wrapper models for feature selection as a single objective and also a multi-objective task.

Before developing the feature selection method based on fuzzy sets and BPSO [148], Chakraborty [134] proposed a GA with fuzzy sets based fitness function to build a filter algorithm for feature selection. This method had the same fitness function as the BPSO based method [148]. The GA based feature selection method was robust but the computational time was usually long. Meanwhile, the performance of proposed method was worse than that of PSO based feature selection method in [148] in terms of the classification accuracy, the number of selected features and the computation time.

Yuan et al. [120] proposed a two-phase feature selection algorithm using both filter and wrapper, which aimed to take advantages of both models. The proposed method started with a filter model to remove irrelevant features, and then a wrapper algorithm was applied to remove the redundant features. In the filter phase, a GA was employed for feature selection with inconsistency criterion to evaluate the fitness of solutions. The wrapper phase started with a feedforward neural network whose input nodes were features in the feature subset obtained in the first phase. The proposed algorithm intended to reduce the computational cost in the wrapper algorithm in the second phase by deleting irrelevant features in the first phase. However, because of feature interactions (epistasis), the proposed algorithm may remove the features in the first phase, which should be included in the best feature subset.

Zhu et al. [152] proposed a feature selection method using a memetic algorithm that is a combination of local search and a GA. In this algorithm, individual features were firstly ranked according to a filter measure. The GA employed the classification accuracy as the fitness function and deleted or added a feature according to the ranking information. The experiments showed that this algorithm outperformed the GA alone and other algorithms. The results also suggested that the performance and the efficiency of the proposed algorithm can be improved by finding a proper balance between the genetic search and the local search.

GAs have also been applied to the wrapper model using multi-objective methods in feature selection problems. For example, based on a multi-objective GA and neural networks (NN), Oliveira et al. [8] proposed a modified wrapper feature selection method. Instead of directly using the classification performance, the sensitivity of NN, which estimates the relationship between input features and classification performance of the NN, was used to evaluate the goodness of the selected features. Experiments on a handwritten digit recognition dataset showed the proposed algorithm reduced the number of features and improved the classification performance. However, only one dataset is not sufficient to verify the effectiveness of this method. Hamdani et al. [153] develop a multi-objective feature selection algorithm using non-dominated sorting based multi-objective genetic algorithm II (NSGAII), but the performance of the proposed algorithm has not been compared with any other feature selection algorithm. Waqas et al. [20] also developed a wrapper algorithm to feature selection using a multi-objective GA. In this method, a subset that was irrelevant with one class and might be relevant with another one was regarded as a non-dominated or Pareto-optimal solution. ID3 was employed to evaluate the fitness of each individual. Experiments showed that selected subsets of features could achieve high classification accuracy.

Overall, it can be seen that the research in a GA for feature selection covers both filter and wrapper, single objective and multi-objective ap-

proaches, which is wider than PSO for feature selection. Meanwhile, a GA for feature selection started at least more than 14 years ago ([120] published in 1999), which has a much longer history than PSO. This is probably because PSO was first proposed much later than GAs and the use of PSO for feature selection has not been fully investigated. However, although a GA has been applied to both single objective and multi-objective feature selection, there is no work conducted using the popular multi-objective GA, i.e. NSGAII, for multi-objective, filter feature selection.

Genetic Programming (GP) for Feature Selection

GP is an evolutionary computation technique inspired by biological evolution to find computer programs that perform a user-defined task. GP evolves computer programs, traditionally represented as tree structures [68]. Basically, in a GP based feature selection method, there are a function set F and a terminal set T including the original features and randomly generated constants. Each tree for each individual (classifier) is initialised with a subset of features using F and T. The population evolves to search for the optimal feature subset using genetic operators iteratively. Many GP based algorithms have been proposed in recent years, including both filter and wrapper models for feature selection.

Muni et al. [116] developed a multi-tree GP algorithm for feature selection (GPmtfs) to simultaneously select a feature subset and design a classifier using the selected features. For a problem with c classes, each classifier in GPmtfs has c trees and each tree was initialised with a random feature subset. Comparisons suggest that GPmtfs achieved better results than SFS, SBS and other methods. However, the number of features selected increases when there are (synthetically added) noisy features. Based on the two crossover operations introduced by [116], Purohit et al. [53] introduced another crossover operator to GP to reduce its randomness when used in a feature selection model. The crossover operator intended to select a subtree from the first parent and find its best place in the second

parent. GP with multi-trees was used to design classifiers with feature selection for a multi-class classification problem. Experiments showed that proposed GP performed better than GPmtfs [116].

Ramirez and Puiggros [154] applied multi-tree GP to solve a multi-class problem, which was to classify the instantaneous cognitive state of a person. The performance measure, the fitness function and initialisation of the classifiers were similar with those in [53]. Experiments showed that the proposed method could accurately classify different cognitive states.

Chien and Yang [155] proposed a feature selection algorithm based on GP and rough sets. Rough membership was used to transform nominal data into numerical values. After transformation, new features and training sets were produced, and then GP was applied to search for the optimal feature subset and learn classification functions. Experiments showed that the proposed method outperformed other different features selection algorithms in terms of the number of selected features and the classification accuracy.

Neshatian and Zhang [135] proposed a GP based filter model as a multi-objective algorithm for feature selection in binary classification problems. Unlike most filter methods that usually could only measure the relevance of single features to the class variables, the proposed algorithm could discover the hidden relationships between subsets of features and the target classes. In this method, an inexpensive binary relevance fitness function was defined to measure the relevance of a GP program tree (a subset of features) to the classification task. In order to explore large feature subsets and at the same time avoid overfitting and bloating, the standard GP was modified by adopting a run-time mechanism for depth control along with an overfit monitoring system. Moreover, a Pareto front archive was proposed as a multi-objective algorithm to maximising the relevance of subsets while minimising their sizes. So the result of the proposed method was a vector of Pareto front points serving as a trade-off matrix. Experiments showed that an inexpensive linear search over this

vector could improve the classification performance of classifiers while decrease their complexity. However, the proposed method might not be quite appropriate for the problems where the best feature subset was expected to have a very large number of features.

Based on [135], Neshatian and Zhang [57] further proposed a GP relevance measure (GPRM) to evaluate and rank feature subsets in binary classification tasks. GPRM extended the concept of a feature relevance measure function by proposing a virtual structure for GP program trees. Through case studies, it was found that the proposed method could detect relevant subsets of features in different situations including multimodal class distributions and mutually correlated features, where other methods had difficulties.

Overall, GP has been applied to both single objective and multi-objective feature selection. Since GP itself can be a classification algorithm, a large number of GP based feature selection algorithms select features and simultaneously train a GP classifier, which are wrapper algorithms. Meanwhile, since GP has a tree representation, where both features and mathematical operators can be evolved, GP are more suitable and more often used for feature construction rather than feature selection [156, 157, 158, 159, 60, 160].

ACO for Feature Selection

Ant colony optimisation (ACO) has also been applied to feature selection problems. Typically, in an ACO based feature selection model, features are represented as nodes in the graph. Edges between the nodes indicate the possible choices of the next feature. Ants traverse through this graph to add nodes (features) until the stopping criterion is satisfied. The feature selection problem is thus transformed to the problem of ant finding the best path on the graph [161, 162, 163, 164, 137].

Jensen and Shen [165] applied ACO to find a small reduct (i.e. feature subset) in rough set theory to address feature selection problems. Later,

Ming [151] also proposed a filter algorithm for feature selection based on ACO and rough set theory. In the proposed method, the CORE of the rough set is used as the start point and forward selection is adopted to search for the best subset of features. In addition, the algorithm stops when the positive region of the selected features in rough set reaches the original positive region. Experiments compares the performance of the proposed method and C4.5 as a feature selection method and the results show that the proposed method could achieve higher accuracy with fewer features than C4.5.

Jensen [136] proposed a filter feature selection model based on ACO and fuzzy-rough theory. The proposed algorithm was examined on classification of web content and complex systems monitoring. Experiments compared the proposed method with other five benchmark techniques. Results illustrated that hill-climbing feature selection algorithms often failed to find minimal subsets even in small and medium-sized datasets. The proposed algorithm and a simulated annealing (SA) based feature selection algorithm achieved similar results and both of them outperformed other three benchmark techniques. However, the proposed method has not been compared with variations of ACO or other evolutionary computation techniques for feature selection.

Gao et al. [161] proposed an ACO based wrapper feature selection algorithm to network intrusion detection. In this wrapper model, least square based SVM was applied as the classifier to evaluate the feature subset generated by ants. Fisher discrimination rate was adopted as the heuristic information for ACO. Experiments on three datasets showed that the proposed method could be an effective algorithm to intrusion feature selection and detection. Later, Kanan and Faez [166] also developed a wrapper feature selection algorithm based on ACO, where both the classification performance and the number of features are considered. The proposed algorithm outperformed GA and other ACO based algorithms on a face detection dataset, but its performance has not been tested on

other problems.

Ke et al. [21] developed a Pareto-based multi-objective ACO for feature selection based on rough set theory. It adopted elite strategy to speed up the convergence performance, used the non-dominated solutions to add pheromone so as to reinforce the exploitation, and applied crowding comparison operator to maintain the diversity of the solutions. In addition, it intended to avoid premature convergence by imposing limits on pheromone values. Compared with a modified non-dominated sorting GA, the proposed method obtained competitive solutions for rough feature selection. However, only three datasets were used in the experiments, which could not confirm the generalisation of the proposed algorithm.

Overall, ACO has been applied to both single objective and multi-objective, filter and wrapper feature selection. However, the datasets used in the papers (we can find) have a relatively small number of features. The use of ACO for feature selection with a relatively large number of features has not been done. However, feature selection is actually more important and necessary in datasets with a large number of features.

2.8 Summary

This chapter reviewed the main concepts of machine learning, classification, feature selection, evolutionary computation techniques, particularly PSO, multi-objective optimisation, entropy and mutual information. This chapter also reviewed the related work of using conventional methods and evolutionary computation algorithms for feature selection.

The limitations of the existing work that form the motivations of this research were also discussed. The overall motivation is that EC techniques have been successfully used to address feature selection problems. Compared with other EC techniques, PSO has the advantages of being computationally less expensive, easier to implement, having fewer parameters and converging more quickly. However, the investigation of using PSO

for feature selection has much less work and much shorter history than other EC algorithms. It is needed to further investigate and improve the performance of PSO for feature selection.

Specifically, the limitations of existing work and the motivations of this research can be summarised as follows.

- The performance of PSO can be improved by developing good initialisation strategies and $gbest$ and $pbest$ updating mechanisms. Feature selection problems are difficult tasks. However, there has been no work on proposing new initialisation strategies for feature selection. Although there are works on updating $gbest$, they are not applied to $pbest$. Therefore, it is needed to investigate new initialisation and updating mechanisms in PSO for feature selection with the expectations of reducing the number of features, increasing the classification performance and reducing the computational time.

- PSO has been successfully used to solve many multi-objective problems and shown promising performance. Feature selection is a multi-objective task. However, there is no existing work investigating the use of PSO for multi-objective (wrapper or filter) feature selection.

- Most of the existing PSO based feature selection algorithms are wrappers and there are very few works using PSO for filter feature selection. Information theory, including entropy and mutual information, can be used to evaluate the relationship between variables. It has been used to develop feature selection algorithms, but the use of information theory and PSO for *filter* feature selection has never been investigated.

- Wrapper feature selection algorithms are argued to be able to achieve better classification performance than filters, but filter algorithms are computationally less expensive and more general than wrappers. However, no thorough work has been conducted to investigate the

differences between the two approaches in terms of the classification performance and the computational cost, and no work has been conducted to investigate the generality of wrappers.

Following Chapters

This thesis aims to address the above-mentioned issues. The following chapters will investigate those issues by developing new algorithms. Chapter 3 will develop new initialisation and $gbest$ and $pbest$ updating mechanisms in PSO to propose a new wrapper based single objective feature selection algorithm. Chapter 4 will develop a PSO based multi-objective, wrapper feature selection approach. Chapter 5 will introduce entropy and mutual information to PSO for feature selection to develop a new filter feature selection approach. Chapter 6 will develop a PSO based multi-objective, filter feature selection approach. Chapter 7 will investigate the difference between filters and wrappers in terms of the classification performance and the computational time, and also examines the generality of wrappers.

Chapter 3

Wrapper Based Single Objective Feature Selection

3.1 Introduction

Feature selection aims to find the minimal feature subset that can achieve similar or even better classification performance than using all features. However, most of the existing feature selection approaches, including PSO based methods, are wrappers and aim to maximise the classification performance only. As a result, the selected features may still have redundancy and the same classification performance can be achieved by a smaller feature subset. Therefore, it is necessary to develop a PSO based feature selection method to optimise both the classification performance and the number of features.

3.1.1 Chapter Goals

The goal of this chapter is to develop a PSO based wrapper feature selection algorithm to maximise the classification performance and minimise the number of features. To achieve this goal, a new fitness function is proposed to combine the two objectives into a single function. Further, PSO

is investigated to optimise these two objectives by developing new initialisation and new $pbest$ and $gbest$ updating mechanisms. Specifically, this chapter will investigate:

- Whether the PSO based algorithm with the new fitness function can select a feature subset with a smaller number of features and better classification performance than using all features, and can achieve better performance than PSO with the fitness function considering only the classification performance;
- Whether the new initialisation strategies can improve the performance of PSO for feature selection over the traditional initialisation strategy;
- Whether the new updating mechanisms can improve the performance of PSO for feature selection over the traditional $pbest$ and $gbest$ updating mechanism;
- Whether combining the new initialisation and updating mechanisms can further increase the performance of PSO for feature selection and can outperform all methods mentioned above.

3.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. The second section describes the new PSO based algorithms. The third section presents the design of the experiments. The results and discussions are presented in the fourth section. The fifth section provides a summary of this chapter.

3.2 The Proposed Algorithms

In this section, new algorithms are proposed to investigate and improve the performance of PSO for feature selection. The overview of a PSO

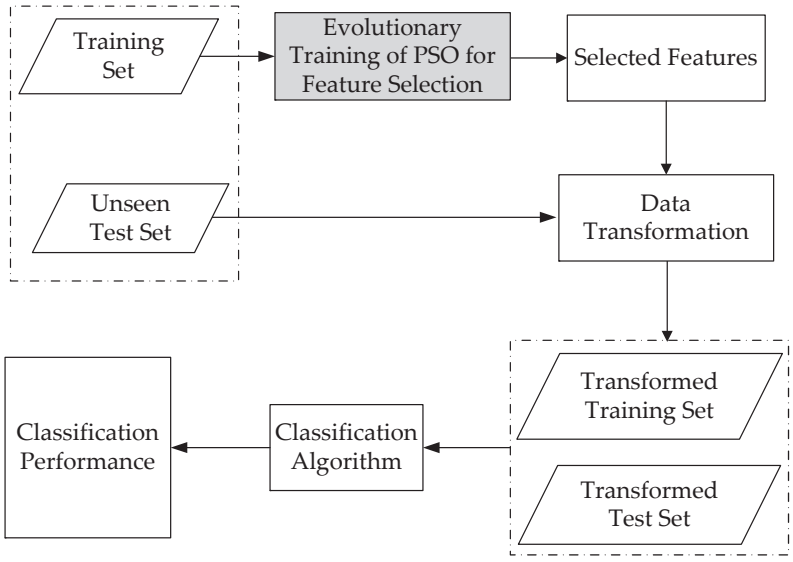

Figure 3.1: The overall structure of PSO based feature selection methods.

based feature selection algorithm is first given. The basic PSO based algorithm (PSOFS) is described as the baseline to test the performance of the newly proposed algorithms. A new fitness function, three new initialisation strategies and three new $pbest$ and $gbest$ updating mechanisms are then proposed to improve the performance of PSO for feature selection.

3.2.1 Overall Structure

The overall structure of the training and testing processes of a PSO based feature selection method is shown in Figure 3.1. The algorithm firstly runs on the training set of the dataset to select a subset of relevant features, which is the evolutionary training process. Then the training set and the test set are transformed to a new training set and a new test set by removing the features that are not selected. A classification algorithm is trained (learns) on the transformed training set. The learnt classifier is then applied to the transformed test set to obtain the final testing classification performance.

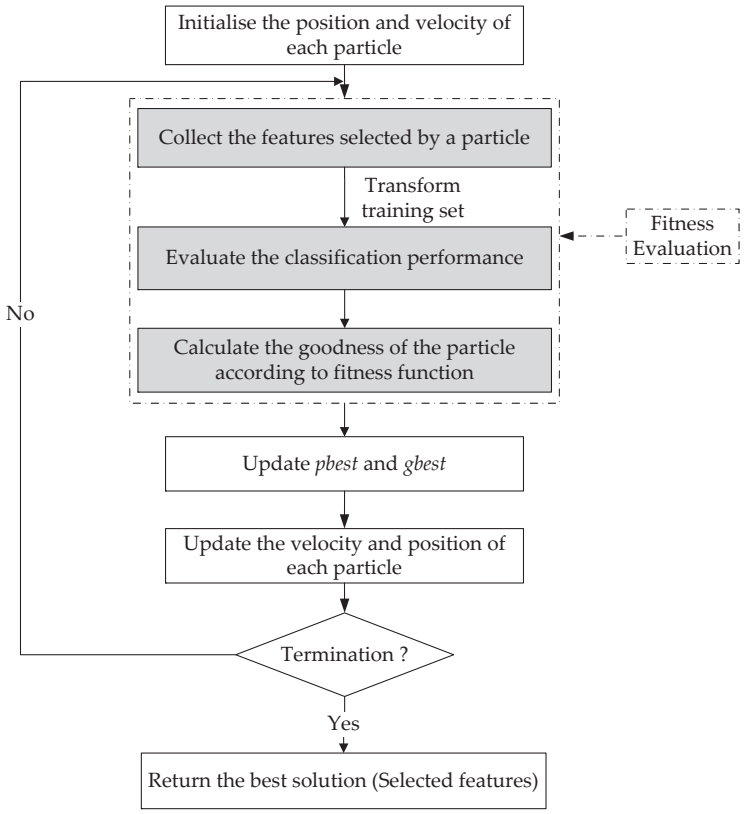

Figure 3.2: The evolutionary training process of a PSO based feature selection algorithm.

The evolutionary training process of a PSO based wrapper feature selection algorithm is shown in Figure 3.2. The key step is the goodness/fitness evaluation procedure. The position of a particle represents a selected feature subset. By removing the features that are not selected, the training set is transformed to a new training set. The classification performance of the selected features is evaluated on the transformed training set. Based on the classification performance, the fitness of the particle is then calculated according to the predefined fitness function. After evaluating the fitness of all particles, the algorithm updates the $pbest$ and $gbest$, and then updates the velocity and position of each particle. The algorithm stops when a pre-

defined stopping criterion, e.g. the maximum number of iterations or an optimal fitness value, has been met.

Representation

In PSO for feature selection, the representation of a particle is a n-bit string, where n is the total number of features in the dataset. The position value in the dth dimension (i.e. x_{id}) is in [0,1], which shows the probability of the dth feature being selected. A threshold θ is used to determine whether a feature is selected or not. If $x_{id} > \theta$, the dth feature is selected. Otherwise, the dth feature is not selected.

3.2.2 Basic PSO Based Feature Selection Method: PSOFS

During the evolutionary training process, Equation 3.1, which aims to minimise the classification error rate, is used as the fitness function to evaluate the goodness of particle i, where the position x_i represents a feature subset.

$$Fitness_1(x_i) = ErrorRate \tag{3.1}$$

where $ErrorRate$ is determined according to Equation 3.2:

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \tag{3.2}$$

where TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives, respectively.

3.2.3 New Fitness Function

The feature subset selected by PSOFS may still contain redundancy, because the basic fitness function (Equation 3.1) does not intend to minimise

the number of features. We hypothesise that the same classification performance could be achieved by a smaller feature subset. In order to further reduce the number of features without reducing the classification performance, a new two-stage feature selection approach (PSO2S) is proposed, where the whole evolutionary process is divided into two stages. In the first stage, the algorithm focuses on the optimisation of the classification performance. In the second stage, the number of features is included into the fitness function. The second stage starts with the solutions achieved in the first stage, which ensures that the minimisation of the number of features is based on feature subsets with high classification performance. The new two-stage fitness function used in PSO2S is shown in Equation 3.3.

$$Fitness_2(x_i) = \begin{cases} ErrorRate, & \text{Stage 1} \\ \alpha * \frac{\#Features}{\#All\ Features} + (1-\alpha) * \frac{ErrorRate}{Error_{all}}, & \text{Stage 2} \end{cases} \tag{3.3}$$

where $ErrorRate$ is the classification error rate obtained by the selected feature subset. α is a constant value and $\alpha \in [0, 1]$. $\#Features$ represents the number of features selected. $\#All\ Features$ stands for the number of all the available features. $Error_{all}$ is the error rate obtained by using all the available features for classification on the training set.

The fitness function in the second stage considers both the number of features and the classification error rate. In order to ensure these two components are in the same range, i.e. [0,1], the number of features is normalised and represented by $\frac{\#Features}{\#All\ Features}$. The classification performance is represented by $\frac{ErrorRate}{Error_{all}}$. $\frac{ErrorRate}{Error_{all}}$ rather than $ErrorRate$ is used here to avoid the situation when $ErrorRate$ is very small (e.g. less than 0.005), $\frac{\#Features}{\#All\ Features}$ plays the major role in the fitness function. In such a situation, the number of features is treated as more important than the classification performance, which is more likely to obtain a feature subset with higher error rate than using all features. Since $ErrorRate$ should

Input : A Training set and a Test set;
Output : $gbest$ (selected feature subset);
Training and test classification accuracies.

```
1  begin
2      randomly initialise the position and velocity of each particle;
3      while Maximumiterations is not reached do
4          evaluate fitness of each particle ;   /* according to Equation
           3.1 in PSOFS or Equation 3.3 in PSO2S */
5          for i=1 to PopulationSize do
6              update the pbest of particle i;
7              update the gbest of particle i;
8          for i=1 to PopulationSize do
9              for d=1 to Dimensionality do
10                 update the velocity of particle i according to Equation 2.2;
11                 update the position of particle i according to Equation 2.1;
12     calculate the classification accuracy of the selected feature subset on the
       test set;
13     return the position of gbest (the selected feature subset), the training and
       test classification accuracies;
```

Algorithm 2: Pseudo-code of PSOFS and PSO2S.

be smaller than $Error_{all}$ after the first stage, $\frac{ErrorRate}{Error_{all}}$ will be in the same range with $\frac{\#Features}{\#All\ Features}$, i.e. [0,1].

When combining them into a single fitness function, α is used to show the relative importance of the number of features and $(1-\alpha)$ shows the relative importance of the classification error rate. Since the classification performance is assumed to be more important than the number of features, α is set to be smaller than $(1-\alpha)$ (i.e. $\alpha < 0.5$).

The pseudo-code of PSOFS and PSO2S can be seen in Algorithm 2. The main difference between PSOFS and PSO2S relies on the fitness evaluation procedure, which is shown in Line 4.

3.2.4 New Initialisation Strategies

Traditionally, particles in PSO are randomly initialised. When using standard PSO for feature selection (i.e. PSOFS), each particle is randomly initialised in terms of both the number of features and the selection of individual features. To investigate the influence the initialisation strategy in PSO for feature selection, three new initialisation strategies are proposed to increase its performance.

The new initialisation strategies are motivated by two typical traditional feature selection methods, forward selection [109] and backward selection [110]. Forward selection starts with an empty set of features and it usually selects a small number of features, but it may miss the optimal feature subset with a larger number of features. Backward selection starts with the full set of features and it usually selects a large number of features, but the computational time is usually longer than forward selection. To simulate forward and backward selection and take their advantages, three new initialisation strategies are proposed in PSO for feature selection, which are **small initialisation** motivated by forward selection, **large initialisation** motivated by backward selection and **mixed initialisation** aiming to take the advantages of forward and backward selection and to avoid their disadvantages. Details of these three new initialisation strategies are described as follows.

(1) Small initialisation. This is motivated by forward selection to initialise each particle using a small number of features, but different combinations of randomly selected individual features.

(2) Large initialisation. This is motivated by backward selection to initialise each particle using a large number of features, but different combinations of randomly selected individual features.

(3) Mixed initialisation. This initialisation strategy combines both the small initialisation and large initialisation strategies. In this strategy, most

particles are initialised using a small number of features (simulating forward selection) and other particles are initialised using large feature subsets (simulating backward selection). Meanwhile, through social interaction (updating $pbest$ and $gbest$), PSO is expected to be able to reach and search the area of the solution space with medium size feature subsets if these feature subsets can achieve better classification performance.

Based on the three new initialisation strategies, three new PSO based feature selection algorithms are developed, which are PSOIni1, PSOIni2 and PSOIni3 using the small initialisation, the large initialisation and the mixed initialisation, respectively. These three new algorithms use the traditional $pbest$ and $gbest$ updating mechanism and use Equation 3.1 as the fitness function, which are the same as PSOFS. The performance of PSOIni1, PSOIni2 and PSOIni3 will be compared with that of PSOFS to investigate the influence of the initialisation strategy.

3.2.5 New $pbest$ and $gbest$ Updating Mechanisms

Sharing information through $pbest$ and $gbest$ is an important component in PSO, as it influences the behaviour of the swarm during the evolutionary process. However, the traditional $pbest$ and $gbest$ updating mechanism in PSO has potential limitation in feature selection tasks.

Traditionally, the $pbest$ and $gbest$ are updated solely based on the fitness value of the particles. In PSOFS, the $pbest$ and $gbest$ are updated based on the classification performance. The $pbest$ of a particle is updated only when the classification performance of the particle's new position is better than the current $pbest$. The $gbest$ is updated only if a $pbest$ achieves higher classification accuracy than the current $gbest$. During the evolutionary training process, if the classification performance of the particle's new position is the same as the current $pbest$, but the number of features is smaller, the particle's new position corresponds to a better feature subset. However, according to the traditional updating mechanism, the $pbest$ will not be updated because their classification performances are the same.

```
   // Fitness_1(x_i) measures the classification error rate of x_i
1  if Fitness_1(x_i) < Fitness_1(pbest) then
2      pbest = x_i ;                                   // Update the pbest
3  else if Fitness_1(x_i) = Fitness_1(pbest) and |x_i| < |pbest| then
4      pbest = x_i ;                                   // Update the pbest
5  if any Fitness_1(pbest) < Fitness_1(gbest) then
6      gbest = pbest ;                                 // Update the gbest
7  else if any Fitness_1(pbest) = Fitness_1(gbest) and |pbest| < |gbest| then
8      gbest = pbest ;                                 // Update the gbest
```

Pseudo-code 3.1: Classification performance as the first priority.

In order to overcome this limitation, three new $pbest$ and $gbest$ updating mechanisms are proposed, which now considers the number of features when updating $pbest$ and $gbest$. The detailed description of the three new mechanisms is shown as follows.

(1) Classification performance as the first priority. $pbest$ and $gbest$ are updated in two situations (See Pseudo-code 3.1). The first situation is that if the classification performance of the particle's new position is better than $pbest$, $pbest$ will be updated and replaced by the new position. In this case, the number of features will be ignored, which is the same as in the traditional updating mechanism. The second situation is that if the classification performance is the same as $pbest$ and the number of features is smaller, the current $pbest$ will be replaced by the particle's new position. After updating the $pbest$ of each particle, $gbest$ of each particle is updated in the same way by comparing $gbest$ with the $pbest$ of the particle and its neighbours.

(2) Improve both the number of features and the classification performance. $pbest$ and $gbest$ are updated in two situations (See Pseudo-code 3.2). The first situation is that if the classification performance of the particle's new position is better than that of $pbest$ and the number of features is not larger than $pbest$, $pbest$ is updated. The second situation is that if the

```
   // Fitness_1(x_i) measures the classification error rate of x_i
1  if Fitness_1(x_i) < Fitness_1(pbest) and |x_i| ≤ |pbest| then
2      pbest = x_i ;                                   // Update the pbest
3  else if Fitness_1(x_i) = Fitness_1(pbest) and |x_i| < |pbest| then
4      pbest = x_i ;                                   // Update the pbest
5  if any Fitness_1(pbest) < Fitness_1(gbest) and |pbest| ≤ |gbest| then
6      gbest = pbest ;                                 // Update the gbest
7  else if any Fitness_1(pbest) = Fitness_1(gbest) and |pbest| < |gbest| then
8      gbest = pbest ;                                 // Update the gbest
```

Pseudo-code 3.2: Improving both the number of features and the classification performance.

number of features is smaller than $pbest$ and the classification performance of the new position is not worse (the same or better) than the current $pbest$, $pbest$ is updated. $gbest$ is updated in the same way.

(3) Compromise between the classification performance and the number of features. $pbest$ and $gbest$ are also updated in two situations (See Pseudo-code 3.3). The first situation is that if the classification performance is better than $pbest$ and the number of features is not larger than $pbest$, $pbest$ is updated. The second situation is that if the classification performance decreases by less than 5% and the number of features is smaller, $pbest$ is updated. $gbest$ is updated in the same way.

All these three new updating mechanisms include the traditional updating mechanism and add other situations in updating $pbest$ and $gbest$. Where available, the first two mechanisms will always select a better feature subset to be the $pbest$ or $gbest$, which either has better classification performance or the same classification performance with a smaller number of features. This can help the algorithm filter out redundant features and make the feature subset with good classification performance and a small number of features to be the *leader* ($pbest$ or $gbest$) of each particle and the whole swarm.

```
// Fitness_1(x_i) measures the classification error rate of x_i
if Fitness_1(x_i) < Fitness_1(pbest) and |x_i| ≤ |pbest| then
    pbest = x_i ;                                   // Update the pbest
else if Fitness_1(x_i) < 0.95 * Fitness_1(pbest) and |x_i| < |pbest| then
    pbest = x_i ;                                   // Update the pbest
if any Fitness_1(pbest) < Fitness_1(gbest) and |pbest| ≤ |gbest| then
    gbest = pbest ;                                 // Update the gbest
else if Fitness_1(pbest) < 0.95 * Fitness_1(gbest) and |pbest| < |gbest| then
    gbest = pbest ;                                 // Update the gbest
```

Pseudo-code 3.3: Compromise between the classification performance and the number of features.

Note that the proposed $pbest$ and $gbest$ updating mechanisms are similar to parsimony pressure used in genetic programming (GP) [167]. In GP, each individual is typically represented as a tree. The size of the trees can be considered in the selection process, where the selection operator prefers smaller trees only when their fitnesses are equal, known as parsimony pressure [167]. However, the proposed updating mechanisms are different from parsimony pressure in two aspects. Firstly, the parsimony pressure in GP changes the size of the trees while the proposed $pbest$ and $gbest$ updating mechanisms do not change the size of the particles that is always the total number of features in the dataset. Secondly, the parsimony pressure is to control the size of the trees in GP, which was not designed for any problem domain, but the number of features considered in the proposed $pbest$ and $gbest$ updating mechanisms are particularly designed for feature selection problems to optimise one of the two main objectives, i.e. minimising the number of features.

Based on the three new $pbest$ and $gbest$ updating mechanisms, three new PSO based feature selection algorithms are developed, which are PSOPG1, PSOPG2 and PSOPG3. These three new algorithms use the random initialisation strategy and use Equation 3.1 as the fitness function, which are the

same as PSOFS. The three new algorithms will be compared with PSOFS to test the influence of the $pbest$ or $gbest$ updating mechanism in PSO for feature selection.

3.2.6 Combination of New Initialisation and Updating Mechanisms

To further investigate and improve the performance of PSO for feature selection, it is also necessary to test the performance of PSO using a new initialisation strategy and a new $pbest$ and $gbest$ updating mechanism. Therefore, a new algorithm named PSOIniPG is formed by combining the mixed initialisation and the $pbest$ and $gbest$ updating mechanism, which treats the classification performance as the first priority. The reason is that the mixed initialisation is proposed to utilise the advantages and avoid the disadvantages of both forward selection and backward selection. Considering the classification performance as the first priority will reduce the number of features without reducing the classification performance, which may even increase the classification performance on unseen test set because of the removal of redundancy. Therefore, PSOIniPG is expected to simultaneously increase the classification performance and reduce the number of features.

The pseudo-code of PSOIniPG can be seen in Algorithm 3. The pseudo-code of the other new algorithms (i.e. PSOIni1, PSOIni2, PSOIni3, PSOPG1, PSOPG2 and PSOPG3) are similar to that of PSOIniPG except for the procedures in the initialisation and the $pbest$ and $gbest$ updating mechanism.

3.3 Design of Experiments

3.3.1 Benchmark Techniques

Two conventional wrapper feature selection methods, linear forward selection (LFS) [114] and greedy stepwise backward selection (GSBS) [168],

Input : A Training set and a Test set;
Output : $gbest$ (selected feature subset);
Training and test classification accuracies.

```
1  begin
2      initialise most of the particles using small feature subsets and the others
       particles using relatively large feature subsets;
3      initialise the velocity of each particle;
4      while Maximum Iterations is not reached do
5          evaluate the fitness (classification performance, i.e. error rate) of each
           particle on the Training set;
6          for i=1 to Population Size do
               // Fitness_1(x_i) measures the error rate of x_i
7              if Fitness_1(x_i) < Fitness_1(pbest) then
8                  pbest = x_i ;          // Update the pbest of particle i
9              else if Fitness_1(x_i) = Fitness_1(pbest) and |x_i| < |pbest| then
10                 pbest = x_i ;          // Update the pbest of particle i
11             if any Fitness_1(pbest) < Fitness_1(gbest) then
12                 gbest = pbest ;        // Update the gbest of particle i
13             else if any Fitness_1(pbest) = Fitness_1(gbest) and |pbest| < |gbest|
               then
14                 gbest = pbest ;        // Update the gbest of particle i
15         for i=1 to Population Size do
16             update the velocity and the position of particle i;
17     calculate the classification accuracy of the selected feature subset on the Test
       set;
18     return the position of gbest (the selected feature subset), the training and test
       classification accuracies;
```

Algorithm 3: The pseudo-code of PSOIniPG.

are used as benchmark techniques in the experiments to examine the performance of the proposed feature selection algorithms.

LFS and GSBS were derived from SFS and SBS, respectively. LFS [114] restricts the number of features that are considered in each step of the for-

ward selection, which can reduce the number of evaluations. Therefore, LFS is computationally less expensive than SFS and can obtain good results. More details can be seen in the literature [114].

The greedy stepwise based feature selection algorithm can move either forward or backward in the search space [168]. Given that LFS performs a forward selection, a backward search is chosen in the greedy stepwise search to form a greedy stepwise backward selection (GSBS). GSBS starts with all available features and stops when the deletion of any remaining feature results in a decrease in evaluation, i.e. the accuracy of classification.

3.3.2 Datasets and Parameter Settings

In order to examine the performance of the proposed feature selection algorithms, a set of experiments have been conducted on 14 datasets, where the details of the datasets can be seen in Table 1.1 on Page 16.

In the experiments, all the instances in each dataset are divided into two sets: a training set and a test set. A common splitting strategy is that 2/3 (around 66%) of instances in the datasets are in the training set and 1/3 (around 33%) of the instances are in the test set [169]. To make it easy, we split 70% of the instances in each dataset as the training set and the other 30% as the test set. The instances are selected so that the proportion of instances from different classes remains the same in both the training set and the test set. Note that n-fold cross-validation is not used here. The main reason is that a feature selection process is different from classification. n-fold cross-validation for classification produces n accuracies and their average value is the desired result. However, a n-fold cross-validation feature selection process produces n feature subsets, but the n feature subsets can not be averaged and the averaged feature subset is not a meaningful/valid solution for users. Another reason is that the majority of datasets have a good number of instances and the 70/30 splitting can

cope well. It is not entirely necessary to use n-fold cross-validation.

As wrapper approaches, the proposed algorithms require a learning/ classification algorithm to evaluate the fitness of the selected feature subsets. Any classification algorithm can be used here. A simple and commonly used classification algorithm [93], KNN is used in the experiments and K=5 (5NN). During the evolutionary training process, the classification performance of a selected feature subset is evaluated by 10-fold cross-validation on the training set. Note that 10-fold cross-validation is performed as an inner loop on the training set to evaluate the classification performance of a single feature subset and it does not generate 10 feature subsets. After the evolutionary training process, the selected feature subset is evaluated on the test set to obtain the testing classification performance. A detailed discussion of why and how 10-fold cross-validation is applied in this way is given by [5].

The experiments of LFS and GSBS are conducted using Waikato Environment for Knowledge Analysis (Weka) [170]. All the settings in LFS and GSBS are kept to the defaults because they can achieve good performance. 5NN is also used in LFS and GSBS. Both LFS and GSBS are deterministic methods, which produce a unique solution (feature subset) for each dataset.

The parameters in all the PSO based feature selection algorithms are selected according to common settings proposed by Clerc and Kennedy [89]. The common settings are used here because using them can clearly test whether the improvement of the performance is caused by the newly proposed mechanisms rather than other factors. The detailed settings are shown as follows: $w = 0.7298$, $c_1 = c_2 = 1.49618$, population size is 30, and the maximum iteration is 100. The fully connected topology is used. According to our preliminary experiments, the threshold θ is set as 0.6 to determine whether a feature is selected or not. In the two-stage approach, i.e. PSO2S, as the maximum number of iteration is 100, the first 50 iterations are set as the first stage and the last 50 iterations are set as the second

stage. $\alpha = 0.2$ and $1 - \alpha = 0.8$, which means the classification performance is more important than the number of features. In PSOIni1, all the particles are initialised using a small number of features, which is around 10% of the total number of features in the dataset, but the combination of individual features for each particle are randomly selected. In PSOIni2, all the particles are initialised using a large number of features (more than half of the total number of features), where for one particle, a random number (e.g. m, where m is between half and the total number of features) is firstly generated and m features are randomly selected to initialise this particle. In PSOIni3 and PSOIniPG, a major part of the swarm (2/3) is initialised using small feature subsets like PSOIni1, and the other minor part of the swarm (1/3) is initialised using more than half of the total number of features like PSOIni2.

For each dataset, the experiments of each algorithm has been conducted for 40 independent runs. Two types of statistical significance tests, pairwise Student's T-test [171] and the non-parametric statistical significance test, Wilcoxon test [172], are performed between the testing classification performance of different algorithms. The significance level is selected as 0.05 (or confidence interval is 95%). The results of the Wilcoxon test are similar (almost identical) to that of T-test. Therefore, only the results of the T-test are listed in the next section.

3.4 Results and Discussions

The performance of the two-stage algorithm is firstly compared with LFS, GSBS, and PSOFS. The results are shown in Table 3.1. Table 3.2 compares the performance of PSO for feature selection using different initialisation strategies, i.e. PSOFS, PSOIni1, PSOIni2 and PSOIni3. Table 3.3 compares the performance of PSO for feature selection using different $pbest$ and $gbest$ updating mechanisms, i.e. PSOFS, PSOPG1, PSOPG2, and PSOPG3. Table 3.4 compares the results of the standard algorithm (PSOFS), PSO2S with

Table 3.1: Results of PSO2S.

*T1: Significance tests against PSOFS. The more "-", the better PSOFS.

*T2: Significance tests against PSO2S. The more "-", the better PSO2S.

Dataset	Method	Size	Best	Mean	StdDev	T1	T2	Dataset	Method	Size	Best	Mean	StdDev	T1	T2
	All	13	76.54			-	-		All	14	70.05			-	-
	LFS	7	74.07			-	-		LFS	4	70.05			-	-
Wine	GSBS	8	85.19			-	-	Australian	GSBS	12	69.57			-	-
	PSOFS	8	100	95.96	1.83		=		PSOFS	3.88	87.44	85.48	3.6		=
	PSO2S	8	100	95.96	1.83	=			PSO2S	3.42	87.44	84.24	4.56	=	
	All	17	80.95			-	-		All	18	83.86			-	-
	LFS	8	79.05			-	-		LFS	9	83.07			-	-
Zoo	GSBS	7	80.0			-	-	Vehicle	GSBS	16	75.79			-	-
	PSOFS	9.18	97.14	95.5	90.1E-2		=		PSOFS	9.52	87.01	84.99	79E-2		=
	PSO2S	9.18	97.14	95.5	90.1E-2	=			PSO2S	8.65	87.01	84.95	77.8E-2	=	
	All	24	68.0			-	-		All	30	92.98			-	-
	LFS	3	68.67			-	-		LFS	10	88.89			-	-
German	GSBS	18	64.33			-	-	WBCD	GSBS	25	83.63			-	-
	PSOFS	13.48	72	69.41	1.33		=		PSOFS	13.42	94.74	93.39	55.8E-2		=
	PSO2S	11.92	72	69.15	1.18	=			PSO2S	5	94.74	93.54	75.3E-2	=	
	All	34	83.81			-	-		All	56	70.0			-	-
	LFS	4	86.67			-	-		LFS	6	90.0			+	+
Ionosp	GSBS	30	78.1			-	-	Lung	GSBS	33	90.0			+	+
	PSOFS	12.58	93.33	88.4	2.14		=		PSOFS	27.35	80	72	6		=
	PSO2S	12.05	91.43	88.14	1.89	=			PSO2S	27.38	90	72.25	6.89	=	
	All	60	76.19			-	-		All	90	94.81			+	+
	LFS	3	77.78			=	=		LFS	14	95.06			+	+
Sonar	GSBS	48	68.25			-	-	MoveLib	GSBS	80	93.46			-	-
	PSOFS	25.82	85.71	77.98	3.97		=		PSOFS	42.6	95.06	94.49	29.2E-2		=
	PSO2S	23.7	85.71	78.02	3.93	=			PSO2S	42.18	95.06	94.51	23.7E-2	=	
	All	100	56.59			-	-		All	166	83.92			=	-
	LFS	8	57.69			=	=		LFS	10	85.31			+	+
Hillvalley	GSBS	90	49.45			-	-	Musk1	GSBS	122	76.22			-	-
	PSOFS	47.32	61.81	57.54	1.52		=		PSOFS	86.48	88.81	84.58	2.04		=
	PSO2S	47.05	61.81	57.57	1.55	=			PSO2S	85.58	88.81	84.58	1.99	=	
	All	500	70.9			-	-		All	617	98.45			-	-
	LFS	7	64.62			-	-		LFS	24	98.34			-	-
Madelon	GSBS	489	51.28			-	-	Isolet5	GSBS	560	97.16			-	-
	PSOFS	258.1	79.49	76.55	1.22		=		PSOFS	318.7	98.77	98.57	9.98E-2		=
	PSO2S	256.48	79.36	76.52	1.26	=			PSO2S	315.62	98.75	98.57	9.65E-2	=	

the new fitness function and PSOIniPG with new initialisation and updating mechanisms.

3.4.1 Results of the New Fitness Function (PSO2S)

In Table 3.1, "All" shows the total number of features in the dataset. "Size" represents the number of features selected by LFS and GSBS or the average number of features selected by PSOFS and PSO2S in the 40 independent runs. "Best" , "Mean", and "StdDev" indicate the best, the average, and the standard deviation of the 40 classification accuracies obtained from the 40 runs on the **test set**. In order to examine the performance of PSOFS and PSO2S, the significance tests have been conducted to compare the classification performance achieved by PSOFS and PSO2S with that of all other methods. "T1" shows the results of T-test between the classification performance achieved by PSOFS and that of "All", LFS, GSBS and PSO2S. "T2" shows the results of T-test between the classification performance achieved by PSO2S and that of "All", LFS, GSBS and PSOFS. In both "T1" and "T2", "+" ("-") means that the classification performance of one algorithm is significantly better (worse) than that of PSOFS or PSO2S with the confidence interval of 0.95. "=" indicates their classification performances are similar to each other (not significantly different).

Results of LFS and GSBS. According to Table 3.1, in most cases, LFS selected a smaller number of features and achieved a similar or even higher classification accuracy than using all available features. GSBS could reduce the number of features, but only increased the classification performance on a few datasets. In most cases, LFS outperformed GSBS in terms of both the number of features and the classification performance. The results suggest that LFS as a forward selection algorithm is more likely to obtain some optimality of the small feature subsets than GSBS (backward selection) because of the different starting points. However, on two of the fourteen datasets (Wine and Zoo), GSBS achieved better classification performance than LFS.

Results of PSOFS. According to Table 3.1, on almost all datasets, the basic PSO based algorithm, PSOFS, evolved a feature subset, which only contained around half (less than half on seven datasets) of the available features and achieved higher classification accuracy than using all features.

Comparing PSOFS with LFS and GSBS, PSOFS achieved significantly better classification performance than LFS on 11 out of the 14 datasets and than GSBS on 12 out of the 14 datasets. On the other two datasets, the best classification accuracy of PSOFS is the same or better than LFS and GSBS. Only on the Lung dataset, LFS and GSBS achieved better classification performance than PSOFS, but this dataset only has 32 instances and it is easy to get poor classification performance. As the objective functions of LFS, GSBS and PSOFS are to minimise the classification error rate (i.e. maximise the classification accuracy) only, the results also suggest that PSO as an evolutionary search technique can obtain a better feature subset than LFS and GSBS.

Results of PSO2S. According to Table 3.1, PSO2S evolved a feature subset with less than half of the available features on nine of the 14 datasets and achieved higher classification accuracy than using all features on 13 of the 14 datasets. Only on the MoveLib dataset, the average classification accuracy of PSO2S is lower than using all features, but the best accuracy of PSO2S is better than using all features. One possible reason for this is that the classification performance of using all the original features in this dataset is already very high (94.81%). Another possible reason is that this dataset has 90 features, but only 306 instances, which makes it easy for the features selected from the small training set perform poorly on the unseen test set. Due to the same reasons, other methods described later show a similar pattern to PSO2S.

Comparing with LFS, PSO2S achieved higher classification accuracy than LFS in almost all cases and a smaller number of features in some cases. PSO2S outperformed GSBS in terms of both the number of features

and the classification performance on almost all datasets. The results further confirm that PSO as an EC technique can better search the solution space to achieve better performance than LFS and GSBS.

Comparing with PSOFS, the number of features selected by PSO2S is the same as PSOFS on two datasets and smaller than PSOFS on all the other 12 out of the 14 datasets. There is no significant difference between the classification performance of PSOFS and PSO2S. This is consistent with our hypothesis on PSO2S, which is to further reduce the number of features without decreasing the classification performance.

The results in Table 3.1 suggest that PSO with different fitness functions in PSOFS and PSO2S obtain different feature subsets. PSO2S further improves the feature subset obtained by PSOFS in almost all cases due mainly to the new two-stage fitness function. In the first stage, the fitness function (Equation 3.3) could guide PSO to search for the feature subset with the minimum classification error rate, then in the second stage, it guides PSO to search for the smallest feature subset with the already achieved high classification performance. Therefore, PSO2S can successfully remove redundant features without reducing the classification performance.

3.4.2 Results of New Initialisation Strategies

Table 3.2 is mainly used to show the influence of the initialisation strategy in PSO for feature selection. The results in Table 3.1 has already shown that the basic PSO based feature selection algorithm (PSOFS) can achieve better performance than LFS and GSBS. Therefore, the results of LFS and GSBS are not included in this section. In Table 3.2, "T1" shows the results of the significance tests between the classification performance of "All" and that of a PSO based feature selection algorithm. "T2" shows the results of the significance tests between PSOFS and another algorithm.

Results of PSOIni1. PSOIni1 uses the proposed small initialisation strategy simulating forward selection. According to Table 3.2, PSOIni1 further reduced the number of features selected by PSOFS, which is less than 20% of the available features. PSOIni1 achieved similar or better classification performance than using all features in most cases. The classification performance of PSOIni1 is similar or better than PSOFS on seven datasets, but worse than PSOFS on the other seven datasets. The results suggest that PSOIni1 has the advantage of the forward selection to select a small number of features. However, for the datasets in which the best classification performance was achieved by a large feature subset, PSOIni1 achieved worse classification performance than the other three algorithms.

In PSOIni1, the number of features was not considered or evolved during the evolutionary training process. Accordingly, if two feature subsets have the same (high) classification performance, but have different numbers of features, then PSOIni1 is more likely to reach the smaller feature subset first and keep it as $pbset$ or $gbest$. Even if PSOIni1 reaches the larger feature subsets, $pbset$ or $gbest$ will not be replaced. Therefore, the number of features in PSOIni1 is usually small, but this may also limit PSOIni1 to search for the area containing solutions with a larger number of features, which results in slightly worse classification performance than PSOFS in many cases. Increasing the number of iterations may address this limitation in some cases, but a larger number of iterations will increase the computational cost.

Results of PSOIni2. PSOIni2 uses the proposed large initialisation strategy simulating backward selection. According to Table 3.2, PSOIni2 selected a larger number of features than PSOFS and PSOIni1. PSOIni2 achieved similar or better classification performance than using all features on 13 of the 14 datasets and similar classification performance to PSOFS in most cases. The results suggest that PSOIni2 suffers from the

Table 3.2: Results of New Initialisation Strategies.

*T1: Significance tests against "All". The more "+", the better the PSO based algorithm.

*T2: Significance tests against PSOFS. The more "+", the better PSOIni1 or PSOIni2 or PSOIni3.

Dataset	Method	Size	Best	Mean	StdDev	T1	T2	Dataset	Method	Size	Best	Mean	StdDev	T1	T2
	All	13	76.54						All	14	70.05				
	PSOFS	8	100	95.96	1.83E0	+			PSOFS	3.88	87.44	85.48	3.6E0	+	
Wine	PSOIni1	3.55	100	95.86	2.34E0	+	=	Australian	PSOIni1	2.58	86.96	78.03	7.7E0	+	-
	PSOIni2	9.42	98.77	96.45	2.53E0	+	=		PSOIni2	4.6	87.92	84.67	3.08E0	+	=
	PSOIni3	8.9	100	96.64	1.49E0	+	=		PSOIni3	3.2	87.44	81.4	8.82E0	+	-
	All	17	80.95						All	18	83.86				
	PSOFS	9.18	97.14	95.5	90.1E-2	+			PSOFS	9.52	87.01	84.99	79E-2	+	
Zoo	PSOIni1	3.22	97.14	94.24	1.33E0	+	-	Vehicle	PSOIni1	3.82	85.24	81.82	1.81E0	-	-
	PSOIni2	9.92	97.14	95.62	92.1E-2	+	=		PSOIni2	10.7	87.99	85.37	1.01E0	+	=
	PSOIni3	7.58	97.14	95.26	86.3E-2	+	=		PSOIni3	9.3	87.8	85.22	77.5E-2	+	=
	All	24	68.0						All	30	92.98				
	PSOFS	13.48	72	69.41	1.33E0	+			PSOFS	13.42	94.74	93.39	55.8E-2	+	
German	PSOIni1	2.55	72	67.39	5.57E0	=	-	WBCD	PSOIni1	3.1	94.74	93.26	2.3E0	=	=
	PSOIni2	16.68	71.67	69.18	1.16E0	+	=		PSOIni2	19.22	94.15	93.01	36.9E-2	=	-
	PSOIni3	10.9	72.33	68.34	1.99E0	=	-		PSOIni3	7.28	94.74	93.83	77.5E-2	+	+
	All	34	83.81						All	56	70.0				
	PSOFS	12.58	93.33	88.4	2.14E0	+			PSOFS	27.35	80	72	6E0	+	
Ionosp	PSOIni1	3.45	92.38	87.83	2.16E0	+	=	Lung	PSOIni1	2.88	90	78.75	9E0	+	+
	PSOIni2	18.25	94.29	86.57	2.02E0	+	-		PSOIni2	37.25	90	73.75	5.33E0	+	=
	PSOIni3	3.18	93.33	87.05	2.23E0	+	-		PSOIni3	13.15	90	75.75	7.71E0	+	+
	All	60	76.19						All	90	94.81				
	PSOFS	25.82	85.71	77.98	3.97E0	+			PSOFS	42.6	95.06	94.49	29.2E-2	-	
Sonar	PSOIni1	6.08	85.71	76.95	5.26E0	=	=	MoveLib	PSOIni1	11.8	95.19	94.25	47.5E-2	-	-
	PSOIni2	32.1	85.71	78.41	3.4E0	+	=		PSOIni2	50.68	95.06	94.42	25.9E-2	-	=
	PSOIni3	12.6	84.13	78.53	3.55E0	+	=		PSOIni3	25.95	94.94	94.37	26.3E-2	-	=
	All	100	56.59						All	166	83.92				
	PSOFS	47.32	61.81	57.54	1.52E0	+			PSOFS	86.48	88.81	84.58	2.04E0	=	
Hillvalley	PSOIni1	6.9	60.99	57.05	1.79E0	=	=	Musk1	PSOIni1	16.58	89.51	81.17	3.25E0	-	-
	PSOIni2	60.12	60.71	57.71	1.2E0	+	=		PSOIni2	106.18	90.21	85.51	2.33E0	+	=
	PSOIni3	9.82	62.09	57.38	2.33E0	+	=		PSOIni3	68.9	91.61	85.54	3.1E0	+	=
	All	500	70.9						All	617	98.45				
	PSOFS	258.1	79.49	76.55	1.22E0	+			PSOFS	318.7	98.77	98.57	9.98E-2	+	
Madelon	PSOIni1	16.18	88.97	76.9	9.43E0	+	=	Isolet5	PSOIni1	115.22	98.69	98.44	14E-2	=	-
	PSOIni2	332.3	78.85	74.45	1.47E0	+	-		PSOIni2	395.6	98.77	98.54	10E-2	+	=
	PSOIni3	225.22	85.77	77.8	3.19E0	+	+		PSOIni3	318.88	98.83	98.57	11.1E-2	+	=

problem of selecting a relatively large number of features. The main reason is that $pbest$ and $gbest$ in PSOIni2 are firstly assigned by feature subsets with a large number of features. Even if PSOIni2 reaches a smaller feature subset with the same classification performance, $pbest$ and $gbest$ will not be

updated. Therefore, this may limit PSOIni2 to search for the space containing solutions with a smaller number of features, and also result in slightly worse classification performance than PSOFS in some cases.

Results of PSOIni3. PSOIni3 uses the proposed mixed initialisation strategy. According to Table 3.2, PSOIni3 achieved better classification performance than using all features on 13 of 14 datasets. In most cases, PSOIni3 achieved similar or better classification performance than PSOFS, but selected a smaller number of features. In most cases, the number of features selected by PSOIni3 is larger than PSOIni1 but smaller than PSOFS and PSOIni2. This might be because PSOIni3 was proposed to simulate both forward and backward selection to utilise their advantages and avoid their disadvantages. As the number of features selected is smaller, the computational time of PSOIni3 is usually shorter than PSOFS and PSOIni2.

Generally, all these four methods using different initialisation strategies selected a smaller number of features and achieved better classification performance than using all features. Although the classification performance is slightly different, the main difference between the above four algorithms are the number of features. The main reason is that all these four algorithms do not consider (or evolve) the number of features during the evolutionary training process. Therefore, the initialisation of the number of features can significantly influence the final solutions. PSOIni3 simulates both forward selection and backward selection in the initialisation procedure, which results in at least as good classification performance as PSOFS, but it selected a smaller number of features in most cases and also reduced the computational time. The results and comparisons suggest that the initialisation strategy is important in PSO for feature selection, and it should not be ignored.

3.4.3 Results of New Updating Mechanisms

Table 3.3 shows the experimental results of PSO using the traditional initialisation strategy (random initialisation) and different $pbset$ and $gbest$ updating mechanisms for feature selection. 'T1" shows the results of the significance tests between the classification performance of "All" and that of a PSO based feature selection algorithm. "T2" shows the results of the significance tests between PSOFS and another algorithm.

Results of PSOPG1. PSOPG1 treats the classification performance as the first priority when updating $pbest$ and $gbest$. As can be seen from Table 3.3, PSOPG1 selected less than half or even less than one third of the available features and achieved similar or significantly better classification than using all features on 13 of the 14 datasets. The only exception is the MoveLib dataset, where PSOPG1 selected around one third of the available features, achieved slightly worse average classification performance (94.57%) than using all features (94.81%), but the best classification performance (95.19%) of PSOPG1 is better than using all features.

PSOPG1 outperformed PSOFS in terms of the classification performance and the number of features on almost all datasets. The main reason is that PSOPG1 takes the classification performance as the first priority, which can firstly guarantee PSOPG1 achieves at least as good classification performance as PSOFS. PSOPG1 also considers the number of features if the classification performance is the same, which can further remove redundant or irrelevant features to reduce the number of features and may further improve the classification performance on the unseen test set.

Results of PSOPG2. PSOPG2 aims to improve both the number of features and the classification performance when updating $pbest$ and $gbest$. According to Table 3.3, PSOPG2 selected around a quarter (or less) of the available features and achieved significantly better or similar classification performance than using all features on almost all datasets. The number

Table 3.3: Results of New Updating Mechanisms.

*T1: Significance tests against "All". The more "+", the better the PSO based algorithm.

*T2: Significance tests against PSOFS. The more "+", the better PSOPG1 or PSOPG2 or PSOPG3.

Dataset	Method	Size	Best	Mean	StdDev	T1	T2	Dataset	Method	Size	Best	Mean	StdDev	T1	T2
	All	13	76.54						All	14	70.05				
	PSOFS	8	100	95.96	1.83E0	+			PSOFS	3.88	87.44	85.48	3.6E0	+	
Wine	PSOPG1	5.95	98.77	95.37	2.01E0	+	=	Australian	PSOPG1	3.82	87.44	85.79	3.71E0	+	=
	PSOPG2	4.7	100	96.79	2.77E0	+	=		PSOPG2	2.85	87.44	78.31	10.2E0	+	-
	PSOPG3	4.65	100	96.79	2.72E0	+	=		PSOPG3	2.58	87.44	77.15	9.9E0	+	-
	All	17	80.95						All	18	83.86				
	PSOFS	9.18	97.14	95.5	90.1E-2	+			PSOFS	9.52	87.01	84.99	79E-2	+	
Zoo	PSOPG1	5.02	97.14	95.36	57E-2	+	=	Vehicle	PSOPG1	9.35	87.01	85.17	84.6E-2	+	=
	PSOPG2	4.1	96.19	95.19	42.2E-2	+	=		PSOPG2	5.48	86.02	84.13	1.27E0	=	-
	PSOPG3	4.35	96.19	95.24	47.5E-2	+	=		PSOPG3	4.45	86.02	83.66	1.26E0	=	-
	All	24	68.0						All	30	92.98				
	PSOFS	13.48	72	69.41	1.33E0	+			PSOFS	13.42	94.74	93.39	55.8E-2	+	
German	PSOPG1	11.48	72.33	68.66	2.17E0	=	=	WBCD	PSOPG1	4.12	94.74	93.74	86.2E-2	+	+
	PSOPG2	6.4	72	68.87	2.06E0	+	=		PSOPG2	3.08	94.74	93.96	1.34E0	+	+
	PSOPG3	4.5	72	68.73	2.05E0	+	=		PSOPG3	2.58	94.74	93.73	1.78E0	+	=
	All	34	83.81						All	56	70.0				
	PSOFS	12.58	93.33	88.4	2.14E0	+			PSOFS	27.35	80	72	6E0	+	
Ionosp	PSOPG1	8.38	92.38	88.74	2.17E0	+	=	Lung	PSOPG1	12.55	90	75.75	7.71E0	+	+
	PSOPG2	3.35	95.24	88.24	2.78E0	+	=		PSOPG2	5.32	90	78.75	5.99E0	+	+
	PSOPG3	3.28	91.43	87.95	2.16E0	+	=		PSOPG3	5.8	90	78.5	6.14E0	+	+
	All	60	76.19						All	90	94.81				
	PSOFS	25.82	85.71	77.98	3.97E0	+			PSOFS	42.6	95.06	94.49	29.2E-2	-	
Sonar	PSOPG1	17.85	85.71	77.42	3.28E0	+	=	MoveLib	PSOPG1	36.65	95.19	94.57	30.7E-2	-	=
	PSOPG2	8.85	87.3	77.82	3.85E0	+	=		PSOPG2	18.4	95.19	94.47	38.7E-2	-	=
	PSOPG3	7.52	87.3	77.03	3.35E0	=	=		PSOPG3	12.35	95.19	94.47	34E-2	-	=
	All	100	56.59						All	166	83.92				
	PSOFS	47.32	61.81	57.54	1.52E0	+			PSOFS	86.48	88.81	84.58	2.04E0	=	
Hillvalley	PSOPG1	40.5	59.89	58	1.23E0	+	=	Musk1	PSOPG1	72.58	88.81	84.37	2.06E0	=	=
	PSOPG2	18.52	60.16	57.94	1.49E0	+	=		PSOPG2	38.78	88.11	83.1	2.78E0	=	-
	PSOPG3	4.52	60.16	56.21	1.85E0	=	-		PSOPG3	30.45	88.11	83.09	2.86E0	=	-
	All	500	70.9						All	617	98.45				
	PSOFS	258.1	79.49	76.55	1.22E0	+			PSOFS	318.7	98.77	98.57	9.98E-2	+	
Madelon	PSOPG1	234.5	80.64	77.1	1.79E0	+	=	Isolet5	PSOPG1	281.85	98.85	98.6	9.01E-2	+	=
	PSOPG2	102.3	85.77	80.15	2.48E0	+	+		PSOPG2	150.88	98.95	98.55	12.8E-2	+	=
	PSOPG3	74.15	86.41	81.71	2.05E0	+	+		PSOPG3	98.98	98.8	98.59	11.6E-2	+	=

of features selected by PSOPG2 is much smaller than that of PSOFS and PSOPG1. The classification performance of PSOPG2 is similar to that of PSOFS on eight of the 14 datasets, slightly worse on three datasets and slightly better on three datasets. This might be because in PSOFS and

PSOPG1, if the classification performance was increased, the number of features was ignored. PSOPG2 aims to ensure that neither the classification error rate nor the number of features is increased when updating $pbest$ or $gbest$. This can further reduce the number of features without decreasing the classification performance in most cases, but it might also cause the algorithm missing the feature subsets with high classification performance and a large number of features.

Results of PSOPG3. In PSOPG3, the classification performance compromises the number of features. According to Table 3.3, on most datasets, PSOPG3 selected around one fifth (or less) of the available features and achieved similar or even better classification performance than using all features. PSOPG3 further reduced the number of features selected by PSOFS, PSOPG1, and PSOPG2. PSOPG3 achieved similar classification performance to PSOFS in most cases, but worse in three cases. The reason is that when updating $pbest$ or $gbest$, the number of features in PSOPG3 was treated as more important than in PSOFS, PSOPG1 and PSOPG2, which guides PSOPG3 to search for the feature subsets with a small number of features. Meanwhile, the classification performance in PSOPG3 compromises the number of features to a very small extent. Therefore, the classification performance of PSOPG3 was slightly worse than PSOFS, PSOPG1 and PSOPG2 in some cases.

Generally, all these four methods using different $pbset$ and $gbest$ updating mechanisms can select a smaller number of features and achieve better classification performance than using all features. PSOPG1 achieved at least as good classification performance as PSOFS, but selected a smaller number of features. The results and comparisons show that the $pbset$ and $gbest$ updating mechanism can *significantly* influence the performance of PSO for feature selection in terms of both the classification performance and the number of features. The results also show that the updating mechanism is more important than the initialisation strategy in PSO for feature selection. Therefore, to improve the performance of PSO for feature selec-

tion, the $pbset$ and $gbest$ updating mechanism should be naturally considered first. Meanwhile, since the initialisation strategy is simple and easy to implement, we should combine them together to further improve the feature selection performance and reduce the computational cost.

3.4.4 Results of PSOIniPG

Table 3.4 shows the results of PSOFS, PSO2S, and PSOIniPG, where "T" shows the result of the significance tests between PSOIniPG and others.

According to Table 3.4, PSOIniPG evolved feature subsets that selected less than half (or even close to 10% on four datasets) of the available features, but achieved significantly better classification performance than using all features on 13 of the 14 datasets. Only on the MoveLib dataset, the average classification performance obtained by PSOIniPG (94.62%) was less, by 0.2%, than that of using all features (94.81%), but the best accuracy (95.19%) is higher than using all features.

Comparing PSOIniPG with PSOFS. According to Table 3.4, PSOIniPG selected feature subsets with a smaller number of features and achieved similar or significantly better classification performance than PSOFS on almost all datasets. This suggests that although PSOFS and PSOIniPG shared the same fitness function (Equation 3.1), the proposed initialisation strategy and $pbest$ and $gbest$ updating mechanism can help PSOIniPG to effectively eliminate/reduce redundant and irrelevant features to obtain a smaller feature subset with similar or significantly better classification performance than PSOFS.

Comparing PSOIniPG with PSO2S. According to Table 3.4, PSOIniPG selected a smaller number of features than PSO2S on 12 of the 14 datasets and achieved similar or better classification performance on 11 datasets. On the seven datasets that have a relatively large number of features, PSOIniPG selected a smaller number of features than PSO2S and achieved

Table 3.4: Results of PSOIniPG.

*T: Significance tests against PSOIniPG. The more "-", the better PSOIniPG.

Dataset	Method	Size	Best	Mean	StdDev	T	Dataset	Method	Size	Best	Mean	StdDev	T
Wine	All	13	76.54			-	Australian	All	14	70.05			-
	PSOFS	8	100	95.96	1.83E0	=		PSOFS	3.88	87.44	85.48	3.6E0	+
	PSO2S	8	100	95.96	1.83E0	=		PSO2S	3.42	87.44	84.24	4.56E0	+
	PSOIniPG	6.78	98.77	95.12	1.87E0			PSOIniPG	3.28	87.44	80.46	9.05E0	
Zoo	All	17	80.95			-	Vehicle	All	18	83.86			-
	PSOFS	9.18	97.14	95.5	90.1E-2	=		PSOFS	9.52	87.01	84.99	79E-2	=
	PSO2S	9.18	97.14	95.5	90.1E-2	=		PSO2S	8.65	87.01	84.95	77.8E-2	=
	PSOIniPG	6.58	97.14	95.52	71.1E-2			PSOIniPG	10.28	87.01	85.31	95.5E-2	
German	All	24	68.0			-	WBCD	All	30	92.98			-
	PSOFS	13.48	72	69.41	1.33E0	+		PSOFS	13.42	94.74	93.39	55.8E-2	-
	PSO2S	11.92	72	69.15	1.18E0	+		PSO2S	5	94.74	93.54	75.3E-2	-
	PSOIniPG	12.88	70.67	68.53	1.39E0			PSOIniPG	3.45	94.74	94.09	82.8E-2	
Ionosp	All	34	83.81			-	Lung	All	56	70.0			-
	PSOFS	12.58	93.33	88.4	2.14E0	+		PSOFS	27.35	80	72	6E0	-
	PSO2S	12.05	91.43	88.14	1.89E0	+		PSO2S	27.38	90	72.25	6.89E0	-
	PSOIniPG	3.2	91.43	87.14	1.88E0			PSOIniPG	6.22	90	78.75	6.4E0	
Sonar	All	60	76.19			-	MoveLib	All	90	94.81			+
	PSOFS	25.82	85.71	77.98	3.97E0	=		PSOFS	42.6	95.06	94.49	29.2E-2	=
	PSO2S	23.7	85.71	78.02	3.93E0	=		PSO2S	42.18	95.06	94.51	23.7E-2	=
	PSOIniPG	10.98	84.13	77.82	2.96E0			PSOIniPG	27.75	95.19	94.62	39.7E-2	
Hillvalley	All	100	56.59			-	Musk1	All	166	83.92			-
	PSOFS	47.32	61.81	57.54	1.52E0	=		PSOFS	86.48	88.81	84.58	2.04E0	=
	PSO2S	47.05	61.81	57.57	1.55E0	=		PSO2S	85.58	88.81	84.58	1.99E0	=
	PSOIniPG	12.72	60.71	57.95	1.48E0			PSOIniPG	79.35	89.51	84.98	2.55E0	
Madelon	All	500	70.9			-	Isolet5	All	617	98.45			-
	PSOFS	258.1	79.49	76.55	1.22E0	-		PSOFS	318.7	98.77	98.57	9.98E-2	-
	PSO2S	256.48	79.36	76.52	1.26E0	-		PSO2S	315.62	98.75	98.57	9.65E-2	-
	PSOIniPG	216.4	84.23	78.49	3.23E0			PSOIniPG	281.05	98.87	98.63	11.9E-2	

similar or better classification performance than PSO2S in all cases. Although the datasets are commonly used benchmark problems in the literature, the performance of a feature selection algorithm on datasets with a larger number of features is clearly more important than on datasets with a smaller number of features. Therefore, PSOIniPG is clearly better than PSO2S. The main reason is that the fitness function in the second stage in PSO2S aims to find a balance between the classification performance and the number of features. Therefore, further reduction of the number of features may also decrease the classification performance. In PSOIniPG, the

fitness function only includes the classification performance during the whole evolutionary process. This ensures that the reduction of the number of features in PSOIniPG will not reduce the classification performance. Meanwhile, the proposed initialisation strategy and $pbest$ and $gbest$ updating mechanism can help PSOIniPG further remove irrelevant or redundant features to reduce the number of features, which in turn could increase the classification performance on unseen test set. In addition, compared with PSO2S, another advantage of PSOIniPG is that it does not need a predefined parameter to balance the relative importance of the classification performance and the number of features.

Note that simply increasing the number of iterations cannot help PSOFS and PSO2S achieve the same performance as PSOIniPG. The main reason is that PSOFS does not consider the number of features and PSO2S needs to manually balance the trade-off (using a predefined parameter) between the classification performance and the number of features. PSOIniPG simulates both forward and backward selection to duplicate their advantages, which helps PSOIniPG pay more attention to small feature subsets, but does not miss the large feature subsets with high classification performance. Meanwhile, because of the proposed $pbest$ and $gbest$ updating mechanism, for two feature subsets with the same classification performance, PSOIniPG will select the smaller one as the new $pbest$ or $gbest$. PSOFS using the traditional updating mechanism will not always be able to do this during the evolutionary training process. Therefore, PSOFS and PSO2S can not achieve as good performance as PSOIniPG.

3.4.5 Analysis on Computational Time

Table 3.5 shows the average computational time used by different methods in the 40 independent runs. All the algorithms used in the experiments are wrapper based feature selection approaches. Therefore, most of their computational time was spent on the fitness evaluation procedure, i.e cal-

Table 3.5: Computational Time (minutes)

Method	Wine	Australian	Zoo	Vehicle	German	WBCD	Ionosp	Lung	Sonar	MoveLib	Hillva	Musk1	Madelon	Isolet5
PSOFS	0.25	4.2	0.11	8.13	12.76	3.97	1.36	0.02	0.75	3.31	42.55	10.09	866.47	363.46
PSO2S	0.22	2.96	0.09	5.87	9.36	2.67	1.08	0.02	0.58	2.7	32.04	7.66	842.01	239.97
PSOIniPG	0.21	2.57	0.07	6.02	9.37	2.07	0.71	0.02	0.37	1.88	14.6	7.22	651.88	247.12
PSOIni1	0.22	3.42	0.06	5.5	5.67	2.36	0.96	0.01	0.3	1.51	14.24	2.6	99.3	142.36
PSOIni2	0.34	3.74	0.1	8.63	14.95	5.18	1.92	0.03	0.99	3.66	47.42	13.78	1220.3	580.04
PSOIni3	0.33	3.97	0.11	8.12	11.89	3.23	0.98	0.02	0.54	2.6	15.66	9.28	818.33	402.78
PSOPG1	0.29	3.42	0.1	6.69	12.52	2.96	1.35	0.02	0.71	2.81	34.53	10.54	947.91	424.48
PSOPG2	0.25	3.29	0.07	5.37	8.08	2.16	0.88	0.01	0.42	2.22	26.43	5.29	437.7	214.47
PSOPG3	0.25	3.82	0.07	6.28	7.11	2.09	1.04	0.01	0.47	1.82	16.18	5.41	372	167.28

culating the classification performance of the selected features.

For the algorithms using a new initialisation strategy, from Table 3.5, it can be seen that PSOIni1 with the small initialisation strategy used the shortest time. PSOIni2 with the large initialisation strategy used the longest time. PSOIni3 used slightly less time than PSOFS. The main reason for the time differences is that a larger number of features needs longer time for classification in each evaluation during the evolutionary training process. PSOIni1 (or PSOIni2), which usually selected the smallest (or the largest) number of features, used the shortest (or the longest) time.

For the algorithms using a new $pbest$ and $gbest$ updating mechanism, from Table 3.5, it can be seen that PSOIni2 and PSOIni3 used less computational time than PSOFS and PSOIni1. The reason is that they selected a smaller number of features. The time used by all these four algorithms also follow the same observations mentioned above, which is the algorithms selecting more features used a longer time.

According to Table 3.5, it can also be seen that PSOIniPG took less time than PSO2S on most datasets. Comparing PSOIniPG with all other PSO based algorithms, PSOIniPG used less computational time than all others except for PSOIni1 because PSOIni1 usually selected a small number of features, but achieved worse classification performance than PSOIniPG.

For the benchmark techniques, LFS usually used less time than the other methods because the forward selection strategy starts with a small

number of features. GSBS costs less time than the other PSO based algorithms on datasets with a small number of features, but used more time on datasets with a large number of features, such as the Madelon and Isolet5 datasets. The reason is that GSBS starts with the full set of features, which needs much longer time for each evaluation. The number of evaluations in GSBS substantially increased on such large datasets while the number of evaluations in the PSO based algorithms still remains the same.

3.5 Chapter Summary

The goal of this chapter was to develop a new PSO based feature selection approach to investigate and improve the performance of PSO for feature selection, which is expected to select a smaller number of features and achieve better classification performance than using all features. To achieve this goal, we investigated the influence of the fitness function, the initialisation strategy and the $pbest$ and $gbest$ updating mechanism in PSO for feature selection. The new algorithms were then developed to successfully improve the performance of PSO for feature selection, and to outperform the classification performance achieved by using all features and two traditional feature selection algorithms.

This chapter shows that the fitness function can significantly influence the performance of PSO for feature selection. A well-designed fitness function can reduce the number of features and/or improve the classification performance. The newly developed two-stage fitness function aims to maximise the classification performance in the first stage and considers both the number of features and the classification performance in the second stage. It can further remove redundancy in the feature subsets evolved by PSO using the fitness fitness considering the classification performance only. As a result, the number of features is further reduced without significantly decreasing or even improving the classification performance.

This chapter shows that the initialisation strategy in PSO for feature selection can not be ignored. When the number of features is not included in the fitness function, the initialisation of solutions (starting points) can influence the size of final feature subsets, which then influences the classification performance. The results show that the small initialisation usually selected a small number of features, but the classification performance was not as good as the large initialisation, which usually selected a large number of features. The mixed initialisation successfully avoided the limitations to select a small number of features, but maintained the classification performance achieved by the large initialisation and traditional random initialisation.

This chapter also shows that the $pbest$ and $gbest$ updating mechanism can significantly influence the performance of PSO for feature selection. By considering the number of features when updating $pbest$ and $gbest$, which are the leaders of particles, the number of features was significantly reduced and the classification performance was maintained or even increased.

By combining the best initialisation strategy and $pbest$ and $gbest$ updating mechanism, the PSO based algorithm simultaneously improved the classification performance and reduced the number of features, especially on datasets with a large number of features. By reducing the number of features, the computational time can also be reduced. The reason is that the computational time was mainly spent on the classification process in the fitness evaluation procedure and a smaller number of features cost less time for each classification process.

The proposed PSO based algorithms in this chapter focus on the single fittest solution found during the evolutionary search process. Although both the number of features and the classification performance are considered, it is unknown whether the obtained solution still have redundancy. Meanwhile, in real-world applications, it is needed to provide users the trade-off between the two objectives. To achieve this, it is thought to use

evolutionary multi-objective algorithms to evolve a set of Pareto front solutions (feature subsets), which allows decision-makers to choose a preferred solution according to their own requirements. Therefore, the next chapter will develop a multi-objective feature selection approach based on multi-objective PSO, where the two objectives are to maximise the classification performance and to minimise the number of features.

Chapter 4

Wrapper Based Multi-Objective Feature Selection

4.1 Introduction

Feature selection in nature is a multi-objective problem, which is to maximise the classification accuracy (minimise the classification error rate) and minimise the number of features. These two objectives are usually conflicting to each other and the optimal decision needs to be made in the presence of a trade-off between them. Treating feature selection as a multi-objective problem can obtain a set of non-dominated feature subsets to meet different requirements in real-world applications. Although PSO, multi-objective optimisation, and feature selection have been individually investigated frequently, the use of PSO for multi-objective feature selection has not been investigated.

4.1.1 Chapter Goals

The overall goal of this chapter is to develop a PSO based multi-objective feature selection approach to classification with the expectation of achieving a Pareto front of non-dominated solutions, which hopefully include

a smaller number of features and achieve a lower classification error rate than using all available features. In order to achieve this goal, we investigate two Pareto front feature selection algorithms based on multi-objective PSO, which are *NSPSOFS* using the idea of non-dominated sorting and *CMDPSOFS* using the ideas of crowding, mutation and ε-dominance. The two feature selection algorithms will be examined and compared with the best single objective algorithm (PSOIniPG) developed in the previous chapter and three well-known evolutionary multi-objective algorithms (Details can be seen in Section 4.3). Specifically, we will investigate:

- whether NSPSOFS can evolve a Pareto front of non-dominated solutions, which include a smaller number of features and achieve better classification performance than using all features, and outperform PSOIniPG,

- whether CMDPSOFS can evolve a Pareto front of non-dominated feature subsets and outperform PSOIniPG, and

- whether NSPSOFS and CMDPSOFS can achieve better performance than three well-known multi-objective algorithms, NSGAII, SPEA2 and PAES.

4.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. The second section describes the new multi-objective feature selection algorithms. The third section describes the design of the experiments. The results and discussions are presented in the fourth section. The fifth section provides a summary of this chapter.

4.2 Multi-Objective Feature Selection Algorithms

In this section, we propose two new algorithms for feature selection using multi-objective PSO. The two objectives in the multi-objective algorithms are to minimise both the number of features and the classification error rate (maximise the classification accuracy).

Representations. The representation used in this chapter is the same as described in Chapter 3, which is a n-bit string, where n is the total number of features in the dataset. A parameter θ is used to determine whether a feature is selected or not.

4.2.1 Multi-Objective Feature Selection Algorithm 1: NSP-SOFS

In this section, we develop a new approach to feature selection using multi-objective PSO with the two main objectives to explore the Pareto front of feature subsets. However, standard PSO was originally proposed for single objective optimisation and could not directly be used to address multi-objective problems. In order to investigate a PSO based multi-objective (Pareto front) feature selection algorithm, one of the most important tasks is to determine a good leader ($gbest$) for each particle from a set of potential non-dominated solutions. NSGAII is one of the most popular evolutionary multi-objective techniques [23]. Li [103] introduces the idea of NSGAII into PSO to develop a multi-objective PSO algorithm and achieves promising results on benchmark functions optimisation. However, this algorithm has never been applied to feature selection problems. In this study, we develop a PSO based multi-objective feature selection algorithm (NSP-SOFS) based on the idea of non-dominated sorting in NSGAII to investigate whether a relatively simple multi-objective PSO can achieve good performance for feature selection problems.

Algorithm 4 shows the pseudo-code of NSPSOFS. The two most important ideas in NSPSOFS are to select a good $gbest$ for each particle and to update the swarm during the evolutionary process. To select a $gbest$, in each iteration, the fitness values (i.e. the number of features and the classification error rate of the selected feature subset) of each particle are calculated. Then the algorithm identifies the non-dominated solutions in the swarm according to the fitness values. The crowding distance of each non-dominated solution is calculated and all the non-dominated solutions are sorted according to the crowding distance. When updating the swarm to a new iteration, a $gbest$ for each particle is randomly selected from the highest ranked non-dominated solutions, which are the least crowded solutions.

To update the swarm, $pbest$ of each particle also needs to update. In NSPSOFS, $pbest$ of a particle is replaced with the new position only if the new position dominates the current $pbest$. After determining the $gbest$ and $pbest$, the new velocity and the new position of each particle are calculated according Equations 2.1 and 2.2. The old positions (solutions) and the new positions of all particles are firstly combined into one union (the size of the union is then twice as the swarm size). The non-dominated solutions in the union are called the first non-dominated front, which are excluded from the union. Then the non-dominated solutions in the new union are called the second non-dominated front. The following levels of non-dominated fronts are identified by repeating this procedure. For the next iteration, solutions (particles) are selected from the top levels of the non-dominated fronts to form a new/updated swarm, starting from the first front (from Line 16 to Line 23). If the number of solutions needed is larger than the number of solutions in the current non-dominated front, all the solutions are added into the swarm for the next iteration. Otherwise, the solutions in the current non-dominated front are ranked according to the crowding distance and the highest ranked solutions are added into the next iteration.

Input : A Training set and a Test set;
Output : A set of non-dominated solutions, training and test accuracies.

begin
  initialise the position and velocity of each particle in the $Swarm$;
  while $Maximum\ Iterations$ *is not met* **do**
    evaluate two objective values of each particle ; `/* number of features and the classification error rate on the Training set */`
    identify the particles ($nonDomS$) that are non-dominated solutions;
    calculate crowding distance of each particle in $nonDomS$;
    Sort particles in $nonDomS$ based on the crowding distance;
    copy all the particles in $Swarm$ to a union ($Union$);
    for *i=1* **to** *Population Size (P)* **do**
      update the $pbest$ of particle i;
      randomly selecting a $gbest$ for particle i from the highest ranked (least crowded) solutions in $nonDomS$;
      update the velocity and postion of particle i
      evaluate two objective values of particle i;
      add the updated particle i to $Union$;
    identify different levels of non-dominated fronts $F = (F_1, F_2, F_3, \ ...)$ in $Union$;
    empty the current $Swarm$ for the next iteration;
    $i = 1$;
    while $|Swarm| < P$ **do**
      if *(*$|Swarm| + |F_i| \leq P$*)* **then**
        add F_i to $Swarm$; $i = i + 1$;
      else if *(*$|Swarm| + |F_i| > P$*)* **then**
        calculate crowding distance in F_i and sort particles in F_i;
        add the $(P - |Swarm|)$ least crowded particles to $Swarm$;
  calculate the classification error rate of the solutions (feature subsets) in the F_1 on the test set ; `/* F1 is the achieved Pareto front */`
  return the positions of particles in F_1, the training and test classification error rates of the solutions in F_1;

Algorithm 4: Pseudo-Code of NSPSOFS

4.2.2 Multi-Objective Feature Selection Algorithm 2: CMDPSOFS

NSPSOFS can extend PSO to tackle multi-objective feature selection problems. However, NSPSOFS has a potential limitation of losing the diversity of the swarm quickly during the evolutionary process. Specifically, when using the idea of NSGAII to update the population, many of the particles in the new iteration may be identical. Because new particles are selected from the combination of current particles and the updated particles, all non-dominated particles that share the same position will be added into the next iteration. Therefore, the diversity of the swarm might be lost fast during the evolutionary process. In order to better address feature selection problems, we use another multi-objective PSO to develop a multi-objective feature selection algorithm, CMDPSOFS, which is based on the ideas of crowding, mutation and ε-dominance [104]. CMDPSO has never been applied to feature selection problems to date.

Algorithm 5 shows the pseudo-code of CMDPSOFS. In order to address the main issue of determining a good leader ($gbest$), CMDPSOFS employs a leader set to store the non-dominated solutions as the potential leaders for each particle. A $gbest$ is selected from the leader set according to their crowding distances and a binary tournament selection. Specifically, a crowding factor is employed to decide which non-dominated solutions should be added into the leader set and kept during the evolutionary process. The binary tournament selection is used to randomly sample two solutions from the leader set and the less crowded solution is chosen as the $gbest$. The maximum size of the leader set is usually set as the number of particles in the swarm. Mutation operators are adopted to keep the diversity of the swarm and improve the search ability of the algorithm. An ε-dominance factor is adopted to determine the size of an archive, which is the number of non-dominated solutions that the algorithm reports. The solutions (feature subsets) in the final archive are used for classification on

Input : A Training set and a Test set;
Output : A set of non-dominated solutions, training and test classification accuracies.

```
1  begin
2      initialise the swarm;
3      initialise the set of leaders LeaderSet and Archive
4      calculate the crowding distance of each member in LeaderSet;
5      while Maximum Iterations is not met do
6          for each particle do
7              select a leader (gbest) from LeaderSet for each particle by
               using a binary tournament selection based on the crowding
               distance;
8              update the velocity and position of particle i
9              apply mutation operators;
10             evaluate two objective values for each particle;  /* number
               of features and the classification error rate
               on the Training set */
11             update the pbest of each particle;
12         identify the non-dominated solutions (particles) to update
           LeaderSet;
13         send leaders to Archive;
14         calculate the crowding distance of each member in LeaderSet;
15     calculate the classification error rate of the solutions in Archive on the
       test set;
16     return the solutions in Archive and their training and test
       classification error rates;
```

Algorithm 5: Pseudo-Code of CMDPSOFS

the test set in each dataset.

Note that CMDPSOFS employs two different mutation operators, uniform mutation in which the variability range allowed for each decision variable is kept constant over generations and non-uniform mutation in which the variability range allowed for each decision variable decreases over time. The two mutation operators are used to maintain the diversity

of the swarm and improve the search ability of the algorithm. In order to achieve this, CMDPSOFS randomly divides the whole swarm into three different groups in the initialisation procedure. The first group does not have any mutation. The second group employs uniform mutation to keep the *global* search ability and the third group employs non-uniform mutation to keep the *local* search ability. Furthermore, the three groups have the same leader set, which allows them to share their success to take advantages of different behaviors to improve the abilities to search for the Pareto non-dominated solutions. Meanwhile, in CMDPSOFS, w is a random value in [0.1,0.5], c_1 and c_2 are random values in [1.5, 2.0], which are different from most of other PSO based algorithms in which these values are constants. This is a convenient way to address the problem of tuning these parameters for difficult tasks.

Both NSPSOFS and CMDPSOFS follow the basic update strategies of standard PSO. As multi-objective algorithms, both NSPSOFS and CMDPSOFS employ a crowding distance to the non-dominated solutions (potential $gbest$) to keep the diversity of the selected $gbest$ for particles. The main differences between NSPSOFS and CMDPSOFS are:

1. How to store the non-dominated solutions (potential $gbest$). In NSPSOFS, there is no external set to store non-dominated solutions and all the non-dominated solutions are kept and updated within the swarm. CMDPSOFS includes an external leader set, which is used to store the non-dominated solutions. The leader set is updated from iteration to iteration.

2. How to choose a $gbest$ for each particle. NSPSOFS ranks all the non-dominated solutions according to their crowding distance, then a $gbest$ is randomly selected from the highest ranked (least crowded) non-dominated solutions. In CMDPSOFS, a binary tournament selection is applied to randomly sample two non-dominated solutions from the leader set and the less crowded solution is selected as $gbest$.

3. How to update the swarm. When updating the swarm to a new iteration, NSPSOFS combines the new solutions (after applying the velocity and position updating equations) and the old solutions into a union. Different levels of non-dominated solutions are identified from the union to form the new swarm in the next iteration. The non-dominated solutions in the last iteration are reported by NSPSOFS. In CMDPSOFS, two different mutation operators are applied together with the velocity and position updating equations to form the new swarm in the next iteration. The solutions in the archive in the last iteration are reported as final solutions.

4. How to set the parameters. The parameters, such as w, c_1 and c_2, in NSPSOFS are constants and in CMDPSOFS are random values within known ranges.

4.3 Design of Experiments

4.3.1 Benchmark Techniques

In order to examine the performance of the two PSO based multi-objective feature selection algorithms, the best single objective algorithm (PSOIniPG) developed in the previous chapter and three well-known evolutionary multi-objective algorithms are used as benchmark techniques in the experiments. The three multi-objective algorithms are NSGAII, strength Pareto evolutionary algorithm 2 (SPEA2) and Pareto archived evolutionary strategy (PAES).

NSGAII is one of the most popular evolutionary multi-objective algorithms proposed by Deb et al. [23]. The main principle of NSGAII is the use of fast non-dominated sorting technique and the diversity preservation strategy. The fast non-dominated sorting technique is used to rank the parent and offspring populations to different levels of non-dominated solution fronts. A density estimation based on the crowding distance is

adopted to keep the diversity of the population. More details can be seen in the literature [23].

SPEA2 is a popular evolutionary multi-objective algorithm proposed by Zitzler et al. [24]. The main principle is a fine-gained fitness assignment strategy and the use of an archive truncation method. In SPEA2, the fitness of each individual is the sum of its strength raw fitness and a density estimation. A new population is constructed by the non-dominated solutions in both the original population and the archive. When the number of non-dominated solutions is larger than the population size, the archive truncation method is applied to determine whether a non-dominated solution should be selected or not according to the distance to its kth nearest neighbour. More details can be seen in the literature [24].

PAES is an evolutionary multi-objective algorithm proposed by Knowles and Corne [102], which has never been applied to feature selection problems. The authours claimed that PAES may represent the simplest possible non-trival algorithm capable of generating diverse solutions in the Pareto front. The main idea of PAES is the use of a local search and the use of an archive of previously found non-dominated solutions. Authours stated that PAES was proposed as a baseline approach for Pareto multi-objective algorithms. More details about PAES can be seen in the literature [102].

NSGAII, SPEA2 and PAES are used here for feature selection with the two objectives of minimising both the number of features and the classification error rate.

4.3.2 Datasets and Parameter Settings

Twelve datasets of varying difficulty are used in the experiments, which can be seen from Table 1.1 in Page 16. The twelve datasets were chosen from the UCI machine learning repository [25]. Note that fourteen datasets were used in Chapter 3, but the MoveLib and Sonar datasets are not used in this chapter. The main reason is that Chapter 3 indicates they are not

quite appropriate for feature selection.

In the experiments, all the instances in each dataset are randomly divided into two sets: 70% as the training set and 30% as the test set, which is the same as in Chapter 3. During the training process, each particle (individual) represents **one** feature subset. The classification performance of a selected feature subset is evaluated by 10-fold cross-validation on the training set. Note that 10-fold cross-validation is performed as an inner loop in the training process to evaluate the classification performance of a single feature subset on the training set and it does not generate 10 feature subsets. After the evolutionary training process, the selected features are evaluated on the test set to obtain the testing classification error rate. A detailed discussion of why and how 10-fold cross-validation is applied in this way is given by [5]. These settings are the same as in Chapter 3 for consistency and comparison purposes.

All the feature selection algorithms here are wrapper approaches, which need a learning/classification algorithm in the evolutionary training process to evaluate the classification performance of the selected feature subset. Any learning algorithm can be used here. One of the simplest and commonly used learning algorithms [143, 52], KNN, was chosen in the experiments. We use K=5 in KNN (5NN) to simplify the evaluation process, which is also the same as in Chapter 3.

In all the PSO based algorithms, the fully connected topology is used, the maximum velocity $v_{max} = 0.6$, the population size $P = 30$ and the maximum iteration $T = 100$. In PSOIniPG and NSPSOFS, the inertia weight $w = 0.7298$, the acceleration constants $c_1 = c_2 = 1.49618$. These values are chosen based on the common settings in the literature [89, 173]. In the CMDPSO, w is a random value in [0.1,0.5], c_1 and c_2 are random values in [1.5, 2.0], and the mutation rate is $1/n$, where n is the number of available features (dimensionality) [104]. The threshold θ in NSPSOFS and CMDPSOFS is set as 0.6, which is the same as in PSOIniPG.

In NSGAII, SPEA2 and PAES, the representation of each individual is

the same as the commonly used representation in GA for feature selection [134, 153], where each individual is encoded by a n-bit binary string, and n is the number of available features. The bit with value "1" indicates the feature is selected in the subset, and "0" otherwise. A bit-flip mutation operator is applied to all these three methods and a single point crossover operator is used in NSGAII and SPEA2. The mutation rate is $1/n$, where n is the number of available features (dimensionality) and the crossover probability is 0.9. For each dataset, all the algorithms have been conducted for 40 independent runs for comparison purposes.

4.4 Results and Discussions

For each dataset, PSOIniPG obtains a single solution in each of the 40 independent runs. The multi-objective algorithms, NSPSOFS, CMDPSOFS, NSGAII, SPEA2 and PAES, obtain a set of non-dominated solutions in each run. In order to compare these two kinds of results, all the 40 solutions resulted from PSOIniPG in the 40 independent runs are shown in the figures. On the other hand, the 40 sets of feature subsets achieved by each multi-objective algorithm are firstly combined into one union set. In the union set, for the feature subsets including the same number of features (e.g. m), their classification error rates are averaged. The average classification error rate is assigned as the average classification performance of the subsets with m features. Therefore, a set of average solutions is obtained by using the average classification error rates and the corresponding numbers (e.g. m). The set of average solutions is called the *average* front and presented here. Besides the average front, the non-dominated solutions in the union set are also presented to compare with the solutions achieved by the single objective algorithm, PSOIniPG. Since PAES achieved similar results to SPEA2, only the results of NSGAII and SPEA2 are shown in this section.

4.4.1 Results of NSPSOFS and CMDPSOFS

As PSOIniPG achieved better performance than the other single objective methods, the results of PSOIniPG are included in this section as the baseline to test the performance of NSPSOFS and CMDPSOFS. Figure 4.1 shows the experimental results of NSPSOFS and PSOIniPG. Figure 4.2 shows the experimental results of CMDPSOFS and PSOIniPG. In Figures 4.1 and 4.2, each chart corresponds to one of the datasets used in the experiments. On the top of each chart, the numbers in the brackets show the total number of available features and the classification error rate using all features. In each chart, the horizontal axis shows the number of features selected and the vertical axis shows the classification error rate.

In Figures 4.1 and 4.2, "-A" stands for the average front resulted from NSPSOFS or CMDPSOFS in the 40 independent runs. "-B" represents the non-dominated solutions resulted from NSPSOFS or CMDPSOFS in the 40 independent runs. "PSOIniPG" shows the 40 solutions of PSOIniPG. On some datasets, PSOIniPG may evolve the same feature subset in different runs and they are shown in the same point in the chart. Therefore, although 40 results are presented, there may be less than 40 distinct points shown in a chart.

(1) Results of NSPSOFS

As can be seen in Figure 4.1, in most cases, the average front of NSPSOFS (NSPSOFS-A) includes two or more solutions, which selected a smaller number of features and achieved a lower classification error rate than using all features. For the same number of features, there are a variety of combinations of features with different classification performances. The feature subsets obtained in different runs may include the same number of features but different classification error rates. Therefore, although the solutions obtained in each run are non-dominated, some solutions in the average front (NSPSOFS-A) may dominate others.

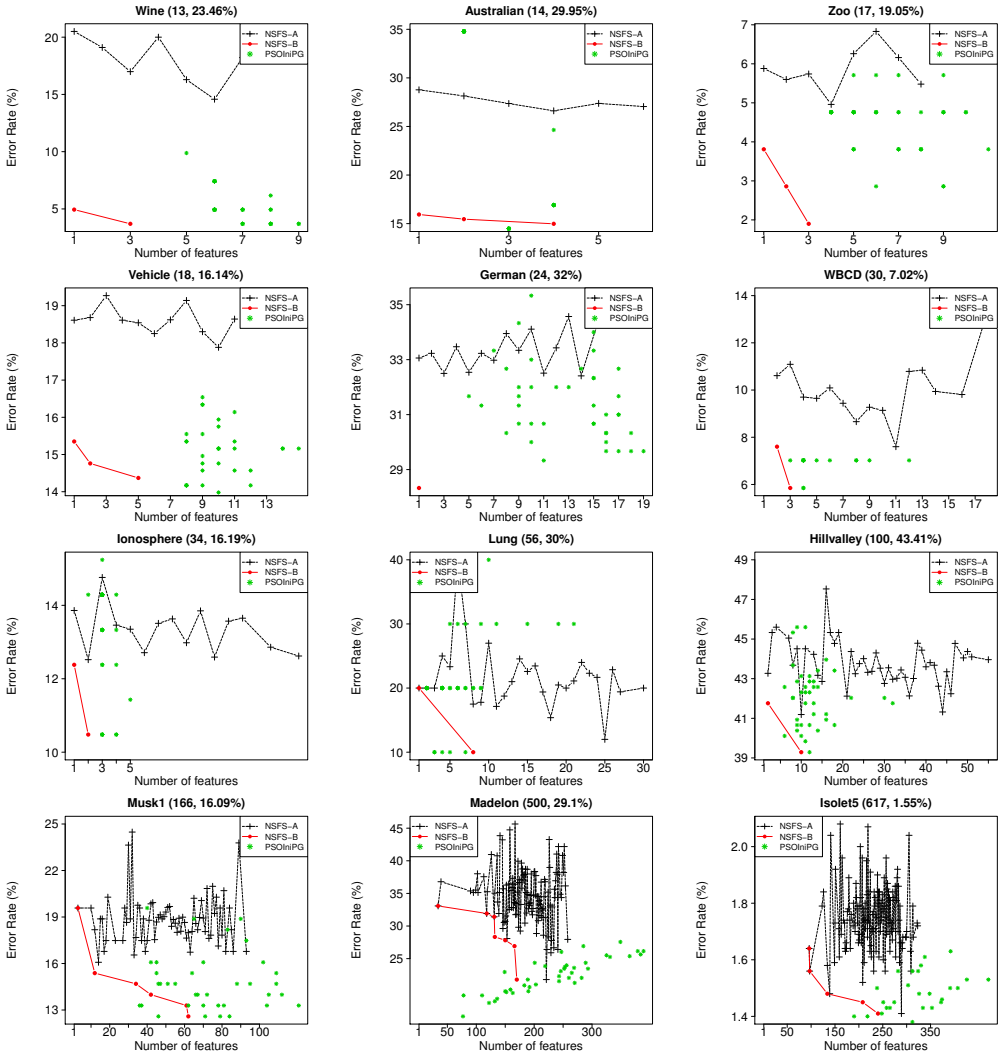

Figure 4.1: Experimental Results of NSPSOFS and PSOIniPG.

According to Figure 4.1, the non-dominated solutions (NSPSOFS-B) achieved by NSPSOFS includes one or more feature subsets, which achieved better classification performance than using all features on *all* datasets. On most datasets, NSPSOFS evolved a feature subset that only selected 1 or 2 features, but achieved a lower classification error rate than using all features. For example, NSPSOFS selected only around 1.8% of the available features (1 from 56) on the Lung dataset and selected 2% of the available

features (2 from 100) on the Hillvalley dataset, but achieved higher classification performance than using all features. In almost all cases, the number of features was reduced to 10% or less, except for around 17.64% in Zoo, 26.4% in Madelon and 21.72% in Isolet5.

Comparing NSPSOFS with PSOIniPG, it can be seen that classification error rates of PSOIniPG and the average front (NSPSOFS-A) are similar on many datasets, but NSPSOFS-B outperformed PSOIniPG in most cases. This shows that NSPSOFS has the potential to achieve better performance than the single objective algorithm PSOIniPG.

The results suggest that although NSPSOFS shares the same parameter settings with PSOIniPG, NSPSOFS as a multi-objective technique can effectively explore the Pareto front to obtain a set of solutions (feature subsets) rather a single best solution in each run. The obtained solutions have the potential to outperform the single best solution resulted from PSOIniPG in terms of both the classification performance and the number of features.

(2) Results of CMDPSOFS

According to Figure 4.2, in *all* cases, CMDPSOFS-A includes two or more feature subsets, which successfully selected a smaller number of relevant features and achieved a lower classification error rate than using all features.

As can be seen in Figure 4.2, CMDPSOFS-B includes one or more small feature subset solutions, which achieved a lower classification error rate than using all features on *all* datasets. On most datasets, CMDPSOFS evolved a feature subset, which only selected 1 or 2 features, but achieved a better classification performance than using all features. For example, CMDPSOFS selected only around 1.8% of the available features (1 from 56) on the Lung dataset and selected only 2% of the available features (2 from 100) on the Hillvalley dataset, but achieved a lower classification error rate than using all features. On almost all datasets, the number of fea-

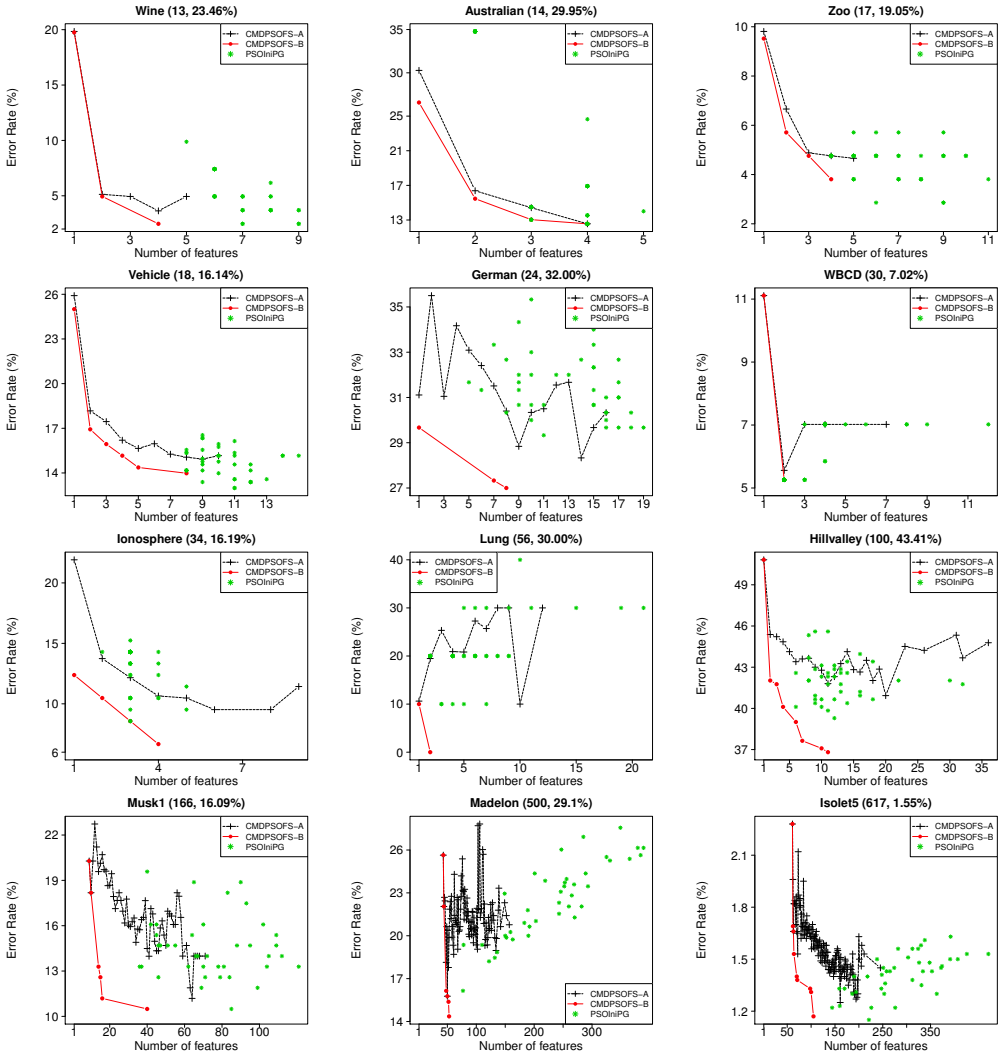

Figure 4.2: Experimental Results of CMDPSOFS and PSOIniPG.

tures selected is less than 10% of the total number of features, except for around 16.67% on the Vehicle dataset and 10.37% on the Isolet5 dataset.

Comparing CMDPSOFS with PSOIniPG, on all datasets, the classification performance of CMDPSOFS-A is similar to that of PSOIniPG, but the number of features in CMDPSOFS-A is smaller than in PSOIniPG in some cases. Moreover, CMDPSOFS-B outperformed PSOIniPG in terms of both the number of features and the classification performance on *all* datasets.

The results further show that CMDPSOFS as a multi-objective technique guided by the two objectives can effectively explore the Pareto front and achieve a set of small feature subsets and improve the classification performance over using all features. By using random parameters (i.e. w, c_1 and c_2), CMDPSOFS can further improve the feature selection performance to outperform the single objective algorithm, PSOIniPG, in *all* cases.

4.4.2 Comparisons Between NSPSOFS, CMDPSOFS, NSGAII and SPEA2

In order to further test the performance of NSPSOFS and CMDPSOFS, they are compared with three popular evolutionary multi-objective algorithms, NSGAII, SPEA2 and PAES. PAES achieved similar results to SPEA2 in terms of the number of features and the classification performance. Therefore, the results of PAES are not presented in this section. Comparisons between NSPSOFS, NSGAII and SPEA2 are shown in Figures 4.3. Comparisons between CMDPSOFS, NSGAII and SPEA2 are shown in Figure 4.4.

(1) Comparisons Between NSPSOFS, NSGAII and SPEA2

According to Figures 4.3, in most cases, the average results, NSPSOFS-A, NSGAII-A and SPEA2-A, include two or more feature subsets, which selected a smaller number of features and achieved better classification performance than using all features. There are also some dominated solutions in the three average fronts and the reason is the same as discussed in Section 4.4.1.

Comparing NSPSOFS-A with NSGAII-A and SPEA2-A, in most cases, the classification performance of three methods are similar. Although in a few cases the classification error rates of NSPSOFS-A are slightly higher than that of NSGAII-A and SPEA2-A, the number of features is usually

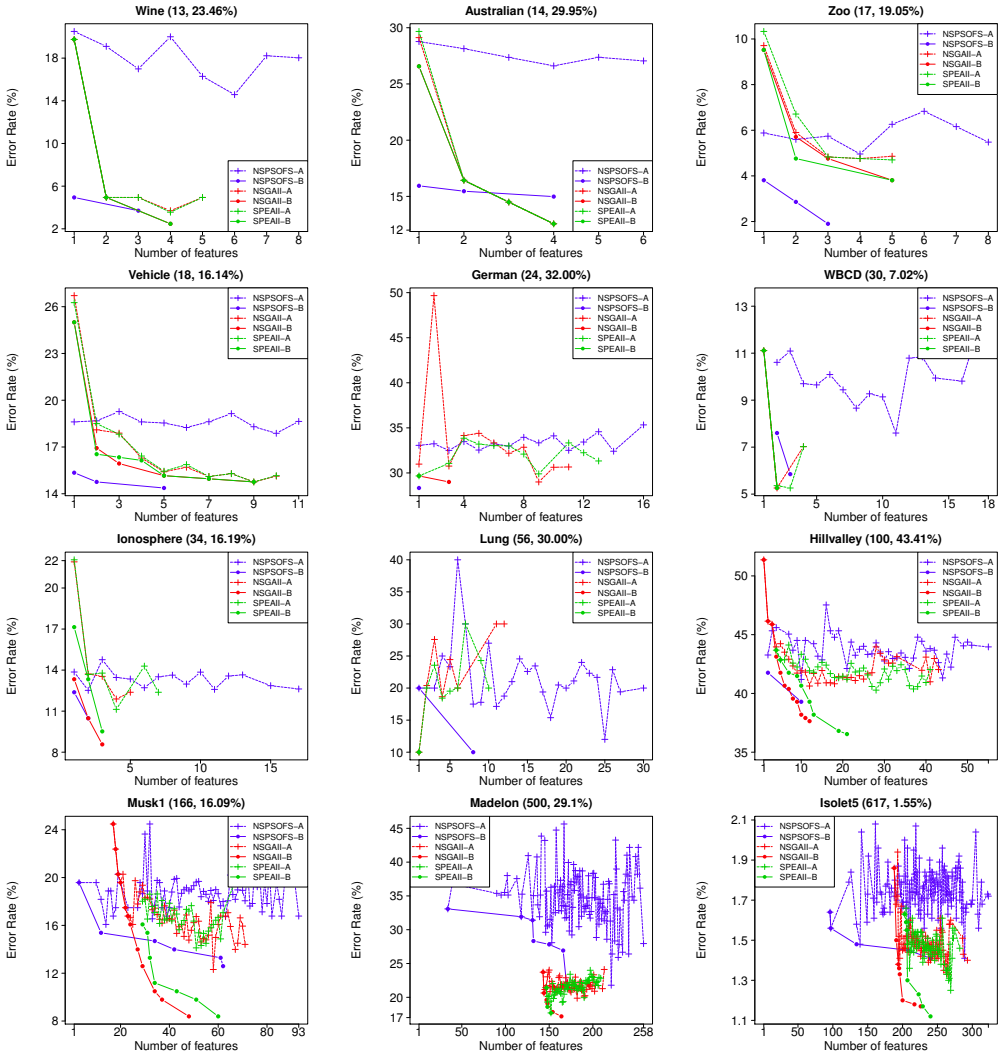

Figure 4.3: Comparisons between NSPSOFS, NSGAII and SPEA2.

smaller in NSPSOFS-A than in NSGAII-A and SPEA2-A.

In terms of the non-dominated solutions (NSPSOFS-B, NSGAII-B and SPEA2-B), the results on different datasets show different patterns. Specifically, on three datasets (Zoo, Vehicle and German), NSPSOFS-B dominated NSGAII-B and SPEA2-B while on two datasets (WBCD and Lung), NSPSOFS-B was dominated by NSGAII-B and SPEA2-B. On the other datasets, there are always solutions in NSPSOFS-B, which dominate solutions in

NSGAII-B and SPEA2-B although there are also solutions in NSPSOFS-B dominated by solutions in NSGAII-B and SPEA2-B. In most cases, NSGAII-B outperformed SPEA2-B in terms of both the classification performance and the number of features.

The results in Figure 4.3 suggest that NSPSOFS, NSGAII and SPEA2 are generally competitive to each other. However, as discussed in Section 4.2.2, NSPSOFS has a potential limitation of quickly losing the diversity of the swarm because of the updating mechanism. The performance of a PSO based multi-objective algorithm could be improved if this limitation can be addressed.

(2) Comparisons Between CMDPSOFS, NSGAII and SPEA2

According to Figure 4.4, the average results, CMDPSOFS-A, NSGAII-A and SPEA2-A, achieved similar classification performance on all datasets. However, the number of features in CMDPSOFS-A is usually smaller than that of NSGAII-A and SPEA2-A, especially on the Madelon and Isolet5 datasets with a large number of features.

Comparing the non-dominated solutions CMDPSOFS-B with NSGAII-B and SPEA2-B, it can be seen that on almost all datasets, CMDPSOFS-B achieved better results than NSGAII-B and SPEA2-B in terms of both the number of features and the classification performance. On datasets with a large number of features, the number of features selected by CMDPSOFS is much smaller than that of NSGAII-B and SPEA2-B. For example, on the Madelon dataset, the number of features in NSGAII-B and SPEA2-B is around 150 while the number in CMDPSOFS-B is only around 50, which means CMDPSOFS further reduced by two thirds of the number of features selected.

The results show that CMDPSOFS can address the limitation in NSPSOFS and achieve better performance than NSPSOFS, NSGAII and SPEA2 in terms of both the number of features and the classification performance.

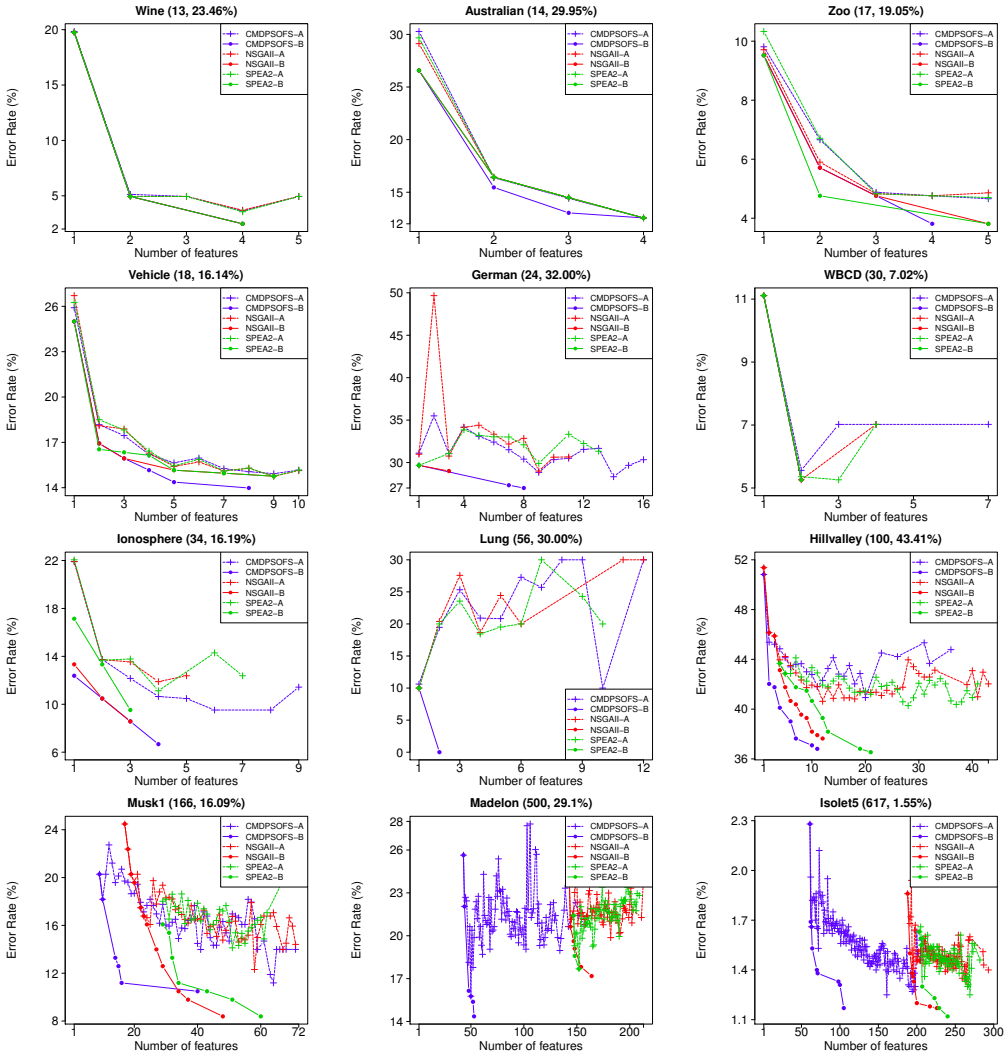

Figure 4.4: Comparisons between CMDPSOFS, NSGAII and SPEA2.

4.4.3 Results of Hyper Volume Indicator

In order to further compare the results of the multi-objective algorithms, NSPSOFS, CMDPSOFS, NSGAII, SPEA2 and PAES, the hyper volume indicator [174] is used in the experiments. In each run, each method obtained two Pareto fronts, which are a *training Pareto front* according to the training classification performance and the number of features, and a *test-*

ing Pareto front according to the testing classification performance and the number of features. Therefore, for each method, we calculated two sets of hyper volume values based on the Pareto-fronts on the training process and the testing process, respectively. Therefore, for each method, 40 hyper volume values on the training process and 40 hyper volume values on the testing process were calculated. As the calculation of hyper volume needs the true Pareto front, which is not available in the datasets, we firstly combine the training (or testing) Pareto front of these five methods into a union, then identify the Pareto front in the union as the "true Pareto front" to calculate the hyper volume values. The hyper volume values of each Pareto front are normalised to hyper volume ratios, which is the division of the hyper volume value of a Pareto front and the hyper volume value of the "true Pareto front". In order to compare NSPSOFS and CMDPSOFS with the other three algorithms, NSGAII, SPEA2 and PAES, the Student's T-test was performed on their hyper volume ratios, where the significance level is set as 0.05 (or confidence interval is 95%).

Results of Hyper Volume on Testing Process

Table 4.1 shows the results of the T-test between NSPSOFS, CMDPSOFS, NSGAII, SPEA2 and PAES on the hyper volume ratios in the *testing* process, where "NS" and "CMD" represent NSPSOFS and CMDPSOFS. In Table 4.1, "+" ("-") indicates that NSPSOFS or CMDPSOFS is significantly better (worse) than another corresponding algorithm. "=" means they are similar. On the WBCD and Lung datasets, the "?" means the hyper volume ratio could not be obtained because the extracted "true Pareto front" only contains two points and its hyper volume value is zero.

Table 4.1 shows that compared with CMDPSOFS, NSGAII, SPEA2 and PAES, NSPSOFS achieved similar results in most cases, although NSPSOFS achieved better results on the Australian dataset and worse results on the Hillvalley and Musk1 datasets. Table 4.1 also shows that CMDPSOFS achieved similar results with other methods in most cases. On the

Table 4.1: T-test on Hyper Volume Ratios on Testing Accuracy

Dataset	Wine		Australian		Zoo		Vehicle		German		WBCD	
	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD
NSPSOFS		=		-		=		=		=		?
CMDPSOFS	=		+		=		=		=		?	
NSGAII	=	=	+	=	=	=	=	=	=	=	?	?
SPEA2	=	=	+	=	=	=	=	=	=	=	?	?
PAES	=	=	=	=	=	=	=	=	=	=	?	?

Dataset	Lung		Ionosphere		Hillvalley		Musk1		Madelon		Isolet5	
	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD
NSPSOFS		?		+		+		+		=		+
CMDPSOFS	?		-		-		-		=		-	
NSGAII	?	?	-	=	-	+	-	+	=	=	+	+
SPEA2	?	?	=	+	-	+	-	+	=	=	+	+
PAES	?	?	-	=	-	+	-	+	=	=	-	+

datasets with a large number of features, such as Hillvalley, Musk1 and Isolet5, CMDPSOFS achieved significantly better results than NSPSOFS, NSGAII, SPEA2, and PAES.

Results of Hyper Volume on Training Process

Table 4.2 shows the results of the T-test between NSPSOFS, CMDPSOFS, NSGAII, SPEA2 and PAES on the hyper volume ratios in the *training* process. It can be seen that NSPSOFS achieved slightly worse results than other methods in most cases, but NSPSOFS achieved better results than NSGAII and SPEA2 on the Isolet5 dataset, where the number of features is large. Table 4.2 also shows that on the datasets with a relatively small number of features, CMDPSOFS usually achieved similar results to NSGAII, SPEA2 and PAES. On the datasets with large numbers of features, such as Hillvalley, Musk1, Madelon and Isolet5, CMDPSOFS achieved significantly better results than NSPSOFS, NSGAII and SPEA2. Although CMDPSOFS achieved slightly worse results than PAES on the training set, CMDPSOFS achieved similar or better results than PAES on the test set (shown in Table 4.1), which is considered due to the overfitting problem in PAES.

Table 4.2: T-test on Hyper Volume Ratios on Training Accuracy

Dataset	Wine		Australian		Zoo		Vehicle		German		WBCD	
	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD
NSPSOFS		+		+		+		+		+		+
CMDPSOFS	-		-		-		-		-		-	
NSGAII	-	=	-	=	-	-	-	-	-	=	-	=
SPEA2	-	=	-	=	-	=	-	=	-	=	-	=
PAES	-	=	-	=	-	-	-	-	-	-	-	=

Dataset	Lung		Ionosphere		Hillvalley		Musk1		Madelon		Isolet5	
	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD	NS	CMD
NSPSOFS		+		+		+		+		+		+
CMDPSOFS	-		-		-		-		-		-	
NSGAII	-	-	-	-	-	+	-	+	=	+	+	+
SPEA2	-	-	-	-	=	+	-	+	=	+	+	+
PAES	-	-	-	-	-	-	-	-	-	-	-	-

From the results of the hyper volume ratios, it can be seen that the hyper volume indicator does not seem a good measure for feature selection problems. Although the results are still consistent, the pattern shown by the hyper volume indicator is less clear than the previous direct comparisons using figures. A possible reason is that the hyper volume indicator is mainly used for continuous multi-objective algorithms, not for discrete multi-objective algorithms. This is also the case for other multi-objective performance indicators/measures. Therefore, in Chapter 6 which presents the work of multi-objective filter feature selection, only direct comparisons will be used and the hyper volume indicator will not be used.

4.4.4 Further Discussions

Experimental results show that both NSPSOFS and CMDPSOFS can be successfully used for feature selection, but NSPSOFS could not achieve as good results as CMDPSOFS. The main reasons are that feature selection tasks are difficult problems with many local optima. CMDPSOFS employs different mechanisms to maintain the diversity of both the leader set and the swarm. Specifically, it selects and filters out crowded leaders and uses different mutation operators to maintain the diversity of the swarm

to avoid stagnation in local optima.

As discussed in Section 4.2, CMDPSOFS has an external leader set to store the non-dominated solutions, which are used as potential leaders for each particle. Different from other multi-objective evolutionary techniques, CMDPSOFS employs a crowding factor to maintain and update the leader set from generation to generation, which helps to filter out some crowded potential leaders. This mechanism will be more helpful for the datasets with a large number of features, where most non-dominated solutions in the leader set may have similar numbers of features and slightly different classification error rates. These crowded non-dominated solutions have a chance to be selected as a leader, which will limit the exploration ability of the algorithm. Eliminating such solutions helps the algorithm explore the solution space more effectively to search for better results.

When selecting a leader for a particle, CMDPSOFS employs a binary tournament to select two non-dominated solutions from the leader set and the less crowded one will be selected as the leader. This mechanism attempts to keep the diversity of the swarm in future iterations and further avoids particles from converging to local optima. Moreover, CMDPSOFS employs different mutation operators for different groups of particles to maintain the diversity of the swarm and balance its global and local search abilities.

By contrast, NSPSOFS is less effective than CMDPSOFS in terms of avoiding stagnation in local optima. NSPSOFS employs different levels of Pareto fronts to store the already found non-dominated solutions. Therefore, all the non-dominated solutions will be kept in the swarm from generation to generation. Such non-dominated solutions may be duplicated and the swarm may lose diversity quickly, which will lead to the problem of premature convergence. Although NSGAII employs the same mechanism to store the non-dominated solutions, NSGAII also employs mutation and crossover operators to keep the diversity. Therefore, NSPSOFS

Table 4.3: Comparisons on Computational Time (in minutes)

	Wine	Australian	Zoo	Vehicle	German	WBCD
NSPSOFS	0.29	4.83	0.11	7.77	12.61	4.34
CMDPSOFS	0.25	4	0.09	6.59	9.3	2.71
NSGAII	0.24	3.99	0.09	6.59	9.32	2.67
SPEA2	0.2	3.24	0.07	5.53	7.64	2.13
PAES	0.21	3.95	0.07	6.35	8.85	2.01
	Lung	Ionosphere	Hillvalley	Musk1	Madelon	Isolet5
NSPSOFS	0.02	1.79	46.04	10.66	868.75	374.63
CMDPSOFS	0.01	1.54	23.49	6.02	394.45	200.71
NSGAII	0.01	1.06	28.88	7.46	721.71	338.23
SPEA2	0.01	0.87	30.23	6.69	694.91	331.91
PAES	0.01	1.04	24.88	5.73	560.83	276.97

usually could not achieve as good results as CMDPSOFS and NSGAII.

4.4.5 Analysis on Computational Time

Table 4.3 shows the average computational time (in minutes) used by NSPSOFS, CMDPSOFS, NSGAII, SPEA2 and PAES over the 40 independent runs.

From Table 4.3, it can be seen that for datasets that have a small number of features and a small number of instances, SPEA2 and PAES generally use less time than the other three methods. However, all algorithms can perform one run in a relatively short time, a few minutes or even less than one minute, such as the Wine, Zoo and Lung datasets. For datasets with a large number of features and instances, CMDPSOFS and PAES used a shorter time than the other three methods, especially for the Madelon and Isolet5 datasets, where CMDPSOFS used much less time than NSPSOFS, NSGAII and SPEA2. On such large datasets, computational time is more important than on small datasets. CMDPSOFS can finish the evolutionary training process in much shorter time and achieve better results, which suggests that this method is a better choice than the other four methods in real-world applications in which a large number of features and instances are involved.

All the methods have the same number of evaluations as they have the same number of individuals and iterations during the evolutionary training process. NSGAII and NSPSOFS generally consumed more time, which is probably caused by the different levels of non-dominated ranking mechanism and the calculation of crowding distances. CMDPSOFS also involves ranking, but it only happens in the small leader set. Therefore, ranking in CMDPSOFS does not cost as much time as NSGAII and NSPSOFS. More importantly, during the evolutionary training process, CMDPSOFS selected smaller numbers of features than the other four algorithms, which cost much less time for the 10-fold cross-validation to calculate the training classification performance in each evaluation, especially for the datasets with a large number of features.

4.4.6 How to Choose a Single Solution

In multi-objective problems, a set of Pareto front (non-dominated) solutions are obtained, which are trade-offs between different objectives. However, selecting a single solution from these solutions is an important issue. In feature selection problems, the two main objectives are minimising the number of features and maximising the classification performance, and the decision is a trade-off between these two objectives. If the Pareto front was "smooth" in that adding each additional feature would reduce the classification error rate by a small, but significant margin, users could weight the trade-off criteria to determine their preferred solutions. However, the results produced show that this is usually not the case. Adding features beyond a certain number does not increase the classification performance. For example, the Musk1 dataset in Figure 4.2, the subset with the lowest classification error rate in CMDPSOFS-B has 40 features. Adding more features does not further increase the classification performance because it does not increase the relevance, but increases the redundancy and the dimensionality. Meanwhile, removing features may not lead to a de-

crease in classification error rate as relevant features may be removed. On the Musk1 dataset, the solution that stands in the "elbow" of CMDPSOFS-B would be a good choice. Therefore, visually seeing these possible solutions in the Pareto front assists users in determining their preferred compromises. This is actually the main reason why solving feature selection problems as multi-objective tasks is important.

4.5 Chapter Summary

This chapter conducted the first study on multi-objective PSO for feature selection. Specifically, we investigated two PSO based multi-objective feature selection algorithms, NSPSOFS and CMDPSOFS. Experimental results show that both NSPSOFS and CMDPSOFS can achieve more and better feature subsets than PSOIniPG, which is the best single objective algorithm developed in the previous chapter. NSPSOFS achieved similar (or slightly worse in some cases) results to other three well-known evolutionary multi-objective algorithms based approaches, i.e. NSGAII, SPEA2 and PAES in most cases. CMDPSOFS outperformed all other methods mentioned above in terms of both the classification performance and the number of features. In particularly, for the datasets with a large number of features, CMDPSOFS achieved better classification performance using fewer features and shorter computational time than the other four multi-objective algorithms.

This chapter finds that as multi-objective algorithms, NSPSOFS and CMDPSOFS can search the solution space more effectively to obtain a set of non-dominated solutions instead of a single best solution. Examining the Pareto front achieved by the multi-objective algorithms can assist users in choosing their preferred solutions to meet their own requirements. Meanwhile, this chapter also discovers that the potential limitation of losing the diversity of the swarm quickly in NSPSOFS limits its performance for feature selection. More importantly, this chapter highlights the bene-

fits of the strategies of maintaining the diversity of the swarm in CMDPSOFS. A crowding factor together with a binary tournament selection can effectively filter out some crowded non-dominated solutions in the leader set. Different mutation operators in different groups of particles can effectively keep the diversity of the swarm and balance its global and local search abilities. These strategies accounts for the superior performance of CMDPSOFS over NSPSOFS, NSGAII, SPEA2 and PAES, especially on the datasets with a large number of features.

This chapter and the previous chapter (Chapter 3) have shown that PSO can be successfully used for feature selection in classification. However, these two chapters mainly focus on wrapper approaches and no filter approaches are involved. Therefore, in order to further investigate and improve the performance of PSO for feature selection, the next two chapters will focus on using PSO to develop new filter feature selection approaches in classification.

Chapter 5

Filter Based Single Objective Feature Selection

5.1 Introduction

Most of the existing PSO based feature selection algorithms are wrapper approaches, which are argued to be computationally more expensive and less general than filter approaches. However, there are very few studies on using PSO for filter feature selection. In filters, the evaluation measure, which determines the goodness of the selected features, is a key factor influencing the classification performance. Information theory is one of the most important theories that are capable of measuring the relevance between features and class labels [1]. However, no work has been conducted to investigate the use of information theory in PSO based feature selection.

5.1.1 Chapter Goals

The overall goal of this chapter is to investigate the use of information theory in PSO for feature selection. To achieve this goal, we develop two new filter feature selection algorithms based on PSO and two information measures with the expectation of selecting a small number of features and

maintaining or even improving the classification performance over using all features. Specifically, we will investigate:

- whether PSO using a mutual information based fitness function can reduce the number of features and achieve similar or even better classification performance than using all features, and can outperform traditional feature selection algorithms;
- whether PSO using an entropy based fitness function can select a smaller number of features and obtain similar or even better classification performance than using all features, and achieve better performance than the above mutual information based algorithm, and
- whether the feature subsets selected by the two new algorithms are general in that they enable high classification performance in different classification algorithms.

5.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. The second section describes the two new filter feature selection algorithms. The third section describes the design of the experiments. The results and discussions are presented in the fourth section. The fifth section provides a summary of this chapter.

5.2 Proposed Algorithms

In this section, two new PSO based filter feature selection methods are proposed, which are PSOMI using mutual information and PSOE using entropy to evaluate the relevance and the redundancy of the selected feature subset.

The overall structure of the training and testing processes in the two proposed filter methods, PSOMI and PSOE, is similar to that of wrapper

methods, which was shown by Figure 3.1 in Chapter 3 (Page 73). The evolutionary training process of PSOMI and PSOE is shown by Figure 5.1, which is different from the evolutionary training process of wrappers (shown by Figure 3.2 on Page 74) in the fitness evaluation procedure. In wrappers, a learning/classification algorithm is used to evaluate the classification performance of the selected features in the fitness function. In PSOMI and PSOE (filters), mutual information and entropy are used to form the fitness functions to evaluate the goodness of the selected feature subset and the classification performance is not involved in their fitness functions.

Representations. The representation used in this chapter is a n-bit binary string, where n is the total number of available features in the dataset. n is also the dimensionality of the search space. In the binary string, "1" represents that the corresponding feature is selected and "0" otherwise.

5.2.1 PSO with Mutual Information for Feature Selection

Mutual information is defined as the information shared between two random variables, which can be used to measure the relevance between a feature x and the class labels c [130]. For a feature subset X with m features, $X = (x_1, x_2, x_3, ..., x_m)$, the relevance between X and c can be shown by Equation 5.1. Filter feature selection aims to maximise the relevance between X and c, which is expected to maximise the classification accuracy (minimise the classification error rate).

$$Rel_{mi}(X, c) = max(\sum_{x \in X} I(x; c)) \tag{5.1}$$

where $I(x; c)$ means the mutual information between x and c, which was defined by Equation 2.14 in Chapter 2 (Page 49).

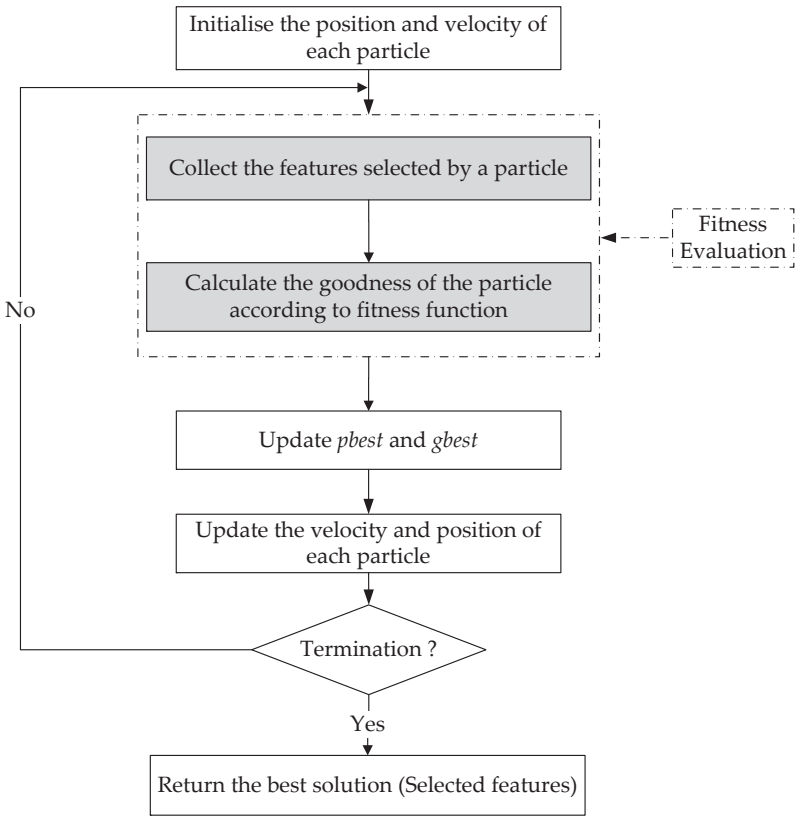

Figure 5.1: The evolutionary training process of a PSO based filter feature selection algorithm.

To maximise the relevance between X and c, a feature selection algorithm with Equation 5.1 as the fitness function will always select the m features that have the largest $I(x, c)$, i.e. the top m features if all the individual features are ranked in a descent order according to $I(x, c)$. However, due to the interactions between features, the combination of m individually good features may not be the best combination of m features. They may have rich redundancy (i.e. including features that contain similar information). The removal of some features may not reduce their classification performance, but might even increase the performance due to the reduc-

tion of the dimensionality. Therefore, it is necessary to reduce the redundancy among features. Based on mutual information, the redundancy in X can be shown by Equation 5.2.

$$Red_{mi}(X) = min(\sum_{x_i,x_j \in X} I(x_i, x_j)) \tag{5.2}$$

where $I(x_i, x_j)$ means the mutual information between feature x_i and feature x_j.

To maximise the relevance and minimise the redundancy, both Rel_{mi} and Red_{mi} should be considered when evaluating the goodness of the selected features. Therefore, a new filter feature selection algorithm, *PSOMI*, is proposed by using Rel_{mi} and Red_{mi} to form the fitness function, which is shown by Equation 5.3. Equation 5.3 is used in PSOMI to guide the evolutionary process of PSO to search for the optimal feature subsets.

$$Fit_{mi} = Rel_{mi} - Red_{mi} \tag{5.3}$$

Fit_{mi} is a maximisation function aiming to maximise the relevance Rel_{mi} and simultaneously minimise the redundancy Red_{mi} in the selected feature subset. The maximisation of Rel_{mi} is to maximise the classification performance while the minimisation of Red_{mi} is to remove redundant features to reduce the number of features. The detailed calculation of $I(x, c)$ and $I(x_i, x_j)$ is shown by Equation 2.14 in Section on Page 49.

5.2.2 PSO with Entropy for Feature Selection

Feature interaction is one of the reasons that make feature selection a challenging problem. Feature interaction can be two-way or multi-way. Therefore, the relevance and redundancy among features can also be two-way or multi-way. Fit_{mi} evaluates the two-way relevance and redundancy by evaluating mutual information between each pair of features. The multi-way relevance and redundancy should be evaluated in groups of features.

Therefore, we propose a new relevance and redundancy measure by evaluating the information gain between features based on entropy, which is an important measure in information theory. The relevance of the selected features is shown by Equation 5.4 while the redundancy is shown by Equation 5.5.

$$Rel_e = max((IG(c|X)) \tag{5.4}$$

$$Red_e = min(\frac{1}{|X|}\sum_{x \in X} IG(x|\{X/x\})) \tag{5.5}$$

Rel_e evaluates the information gain of the class label c given information of the features in X, which shows the relevance between X and c. Red_e evaluates the redundancy contained in X by summing up the information gained for each $x \in X$ by giving X/x, where X/x means all the features in X except for feature x. Both Rel_e and Red_e involve the calculation of a single feature given information about a set of features. Taking $IG(c|X)$ in Rel_e as the example,

$$\begin{aligned} IG(c|X) &= H(c) - H(c|X) \\ &= H(c) - (H(c \cup X) - H(X)) \\ &= H(c) + H(X) - H(c \cup X) \end{aligned}$$

where $H(X)$ is the joint entropy of all the features in X. If $X = \{W, Y, Z\}$ (W, Y, Z are the selected features), then

$$H(W, Y, Z) = -\sum_{w \in W}\sum_{y \in Y}\sum_{z \in Z} p(wyz)\log_2 p(wyz).$$

Based on Equations 5.4 and 5.5, a new fitness function is developed and shown in Equation 5.6.

$$Fit_e = Rel_e - Red_e \tag{5.6}$$

Fit_e is also a maximisation function aiming to maximise the relevance Rel_e (classification performance) and simultaneously minimise the redundancy Red_e (the number of features selected). Equation 5.6 evaluates the selected features as a whole rather than evaluating each pair of features like in Equation 5.3. The aim here is to select a subset of features, all of which working together can perform well. A new filter feature selection method is then proposed based on PSO and Equation 5.6. For presentation convenience, we call the new method *PSOE*.

5.2.3 Different Weights for Relevance and Redundancy in PSOMI and PSOE

The relevance and redundancy are equally important in the two fitness functions (Equations 5.3 and 5.6) in PSOMI and PSOE. In order to investigate how the weightings in the fitness functions (i.e. the relative importances for the relevance and redundancy) influence the feature selection performance, a parameter is introduced into Equation 5.3 in PSOMI (shown as α_{mi}) and Equation 5.6 in PSOE (shown as α_e). The two fitness functions are then re-defined as Equations 5.7 and 5.8.

$$Fit_{mi} = \alpha_{mi} * Rel_{mi} - (1 - \alpha_{mi}) * Red_{mi} \tag{5.7}$$

$$Fit_e = \alpha_e * Rel_e - (1 - \alpha_e) * Red_e \tag{5.8}$$

where α_{mi} and α_e are constant values in $[0, 1]$, which represent/reflect the relative importance of the relevance in two fitness functions. $(1 - \alpha_{mi})$ and $(1 - \alpha_e)$ show the relative importance of the reduction of the redundancy. Since the classification performance is usually more important than the number of features, we assume the relevance is more important than the redundancy (i.e. the number of features). Therefore, α_{mi} or α_e is set to be larger than $(1 - \alpha_{mi})$ or $(1 - \alpha_e)$, i.e. greater than 0.5. When $\alpha_{mi} = (1 - \alpha_{mi}) = 0.5$ and $\alpha_e = (1 - \alpha_e) = 0.5$, Equations 5.7 and 5.8 are the same

as Equations 5.3 and 5.6, where the relevance and redundancy are equally important.

5.2.4 Pseudo-code of PSOMI and PSOE.

Algorithm 6 shows the pseudo-code of PSOMI and PSOE, where the main difference between PSOMI and PSOE is the fitness evaluation procedure shown in Line 4. Since the performance of continuous PSO for feature selection has been investigated in Chapter 3 and Chapter 4, which show that continuous PSO can be successfully used to address feature selection problems. In this Chapter, binary PSO is used in PSOMI and PSOE to investigate its performance for feature selection, but continuous PSO can be easily implemented by changing the position updating equations in Line 11 of Algorithm 6.

5.3 Design of Experiments

5.3.1 Benchmark Techniques

In order to examine the performance of the proposed algorithms, two conventional filter feature selection methods (CfsF and CfsB) and a traditional wrapper method (GSBS) are used for comparison purposes in the experiments.

Hall [127] proposed a correlation based filter feature selection measure named (Cfs). Cfs evaluates the correlation between each feature and the class labels and between each pair of features using mutual information. Cfs is implemented in Weka [175] and it needs a search technique. Greedy stepwise search [168] in Weka is selected as the search technique to perform feature selection using Cfs to evaluate the goodness of the selected feature subsets, which is used as a representative sample of a traditional information theory based algorithm to test the performance of the pro-

```
Input  : A Training set and a Test set;
Output : gbest (selected feature subset);
         Training and test classification accuracies.
1  begin
2      initialise the position and velocity of each particle;
3      while Maximumiterations is not reached do
4          evaluate fitness of each particle ; /* according to Equation
           5.7 in PSOMI or Equation 5.8 in PSOE */
5          for i=1 to PopulationSize do
6              update the pbest of particle i;
7              update the gbest of particle i;
8          for i=1 to PopulationSize do
9              for d=1 to Dimensionality do
10                 update the velocity of particle i according to Equation 2.2;
11                 update the position of particle i according to Equations
                   2.4 and 2.5 ;
12     calculate the classification accuracy of the selected feature subset on
       the test set;
13     return the position of gbest (the selected feature subset), the training
       and test classification accuracies;
```

Algorithm 6: Pseudo-code of PSOMI and PSOE

posed PSO and information theory based methods. Cfs with greedy stepwise forward selection is named as CfsF and with backward selection is named as CfsB.

The Greedy stepwise search can also be used in wrapper feature selection. GSBS, which has already been used in Chapter 3 to test the performance of the wrapper algorithms, is also used to in this chapter. GSBS starts with all available features and stops when the deletion of any remaining feature results in a decrease in evaluation, i.e. the classification accuracy. GSBS is used here to test whether the new PSO based *filter* algorithms can achieve better performance than a traditional *wrapper* algorithm, which is argued to be better than filter approaches in terms of the

classification performance. Note that this chapter is not intended to thoroughly compare the new filter methods with all wrapper methods developed in the previous chapters. More comparisons and discussions will be provided in Chapter 7.

All the three traditional methods are deterministic methods, which produce a unique feature subset for each dataset. So each of them has a single result for each test set.

5.3.2 Datasets and Parameter Settings

Ten datasets of varying difficulty are used in the experiments, which can be seen from Table 1.1 on Page 16. The ten datasets were chosen from the UCI machine learning repository [25] and they are used as representative samples of the problems that the proposed algorithms can address. *Note* that all the data in the selected datasets are categorical values because mutual information and entropy are mainly used for discrete variables.

In the experiments, all the instances in each dataset are randomly divided into two sets: 70% as the training set and 30% as the test set, which is the same as in Chapters 3 and 4. The filter feature selection algorithms (except for GSBS) are firstly run on the training set to select feature subsets and then the classification performance of the selected features will be evaluated on the test set by a classification algorithm. In filter approaches, the training process is independent of any classification algorithm. In the testing process, any algorithm can be used here. To evaluate the claim that filter approaches are general to different classification algorithms, three commonly used algorithms, DT, NB, and KNN with K=5, are used to evaluate the classification performance of the selected features.

GSBS is a wrapper method, where a learning/classification algorithm is used during the training process to evaluate the classification performance (the goodness) of the selected features. The same classification algorithm is then used during the testing process to evaluate the testing clas-

sification performance of the selected features. All the three classification algorithms, i.e. DT, NB, and KNN, are used in GSBS.

The parameters of binary PSO in PSOMI and PSOE are set as follows: inertia weight $w = 0.7298$, acceleration constants $c_{mi} = c_e = 1.49618$, maximum velocity $v_{max} = 6.0$, population size $P = 30$, the maximum number of iterations $T = 500$. The fully connected topology is used in both PSOMI and PSOE. These values are chosen based on the common settings in the literature [84, 89]. Five different values for α_{mi} in PSOMI and α_e in PSOE are used in the experiments to investigate the influence of different weights for relevance and redundancy. They are 0.9, 0.8, 0.7, 0.6 and 0.5, where the classification performance is treated most important when the value of α_{mi} and α_e is 0.9. When the value of α_{mi} and α_e is 0.5, the relevance and redundancy are treated as equally important. Since the results of $\alpha_{mi} = \alpha_e = 0.6$ or 0.8 have a similar pattern to other values, their results are not presented in the next section.

The settings for CfsF, CfsB and GSBS follow the default settings in Weka because they produce good results. For each dataset, PSOMI and PSOE have been conducted for 40 independent runs. In order to examine the classification performance of PSOMI (PSOE), a statistical significant test, Student's T-test, is performed with a significance level of 0.05 (95% confidence interval) between the 40 classification accuracies achieved by PSOMI (PSOE) and the classification accuracy obtained by using all features.

5.4 Results and Discussions

This section firstly discussed the results of PSOMI and PSOE, which are shown in Tables 5.1 and 5.2. The performance of PSOMI and PSOE are then compared with that of the three traditional feature selection methods (shown in Table 5.5).

5.4.1 Results of PSOMI

Table 5.1 shows the experimental results of PSOMI with three different α_{mi} in PSOMI. In table 5.1, "All" means that all of the available features are used for classification. "Size" represents the average size of the feature subsets evolved by each algorithm in the 40 independent runs. "Best" and "Mean" show the best and the average of the 40 classification accuracies. "StdDev" shows the standard deviation of the 40 test accuracies. Since the standard deviation values for all the three classification algorithms are small, only the values for DT are given and that of KNN and NB are not listed in the table to save space. "T" shows the result of the T-tests, where "+" ("-") indicates that the classification performance of PSOMI is significantly better (worse) than that of using all features. "=" means they are similar.

Table 5.1 shows that PSOMI with *mutual information* as the evaluation criterion can usually reduce the number of features and maintain or even increase the classification performance over using all features. In most cases, the number of features selected by PSOMI is less than 40% of the total number of features. On nine of the ten datasets, PSOMI evolved a smaller number of features and maintained or even increased the classification performance overing using all features on at least one of the three classification algorithms, DT, KNN and NB. For example, on the Spect dataset, PSOMI with the three α_{mi} values selected 3, 4 or 6 features from the original 22 features and increased the classification performance of all the three classification algorithms over using all features. Although in some cases the overall classification performance of PSOMI is worse than that of using all features, its best result is better than using all features.

On all the ten datasets, PSOMI with a large α_{mi} evolved a subset with more features than with a small α_{mi}. In most cases, a large α_{mi} led to better classification performance than a small α_{mi}. This is because when α_{mi} is large, the relevance is treated as more important than when α_{mi} is small. By contrast, the redundancy, which indirectly influences the number of

Table 5.1: Results of PSOMI with Different α_{mi}.

Dataset	α_{mi}	Size	DT				KNN			NB		
			Best	Mean	Std	T	Best	Mean	T	Best	Mean	T
Lymph	All	18	82.22				82.22			80		
	0.9	14	82.22	82.22	5.68E-14	-	86.67	86.67	+	82.22	82.22	+
	0.7	7	77.78	77.78	5.68E-14	-	82.22	82.22	-	77.78	77.78	-
	0.5	4	77.78	77.78	5.68E-14	-	51.11	51.11	-	73.33	73.33	-
Mushroom	All	22	100				100			95.98		
	0.9	9.1	99.59	99.54	8.04E-2	-	99.59	99.54	-	98.11	97.94	+
	0.7	4.48	97.87	97.87	8.53E-14	-	97.87	97.87	-	97.76	97.76	+
	0.5	2	97.87	97.87	8.53E-14	-	97.87	97.87	-	97.76	97.76	+
Spect	All	22	66.25				63.75			70		
	0.9	6	71.25	71.19	39E-2	+	66.25	66.19	+	73.75	73.72	+
	0.7	4	71.25	71	1.56E0	+	73.75	73.75	+	72.5	72.5	+
	0.5	3	71.25	71.19	27.2E-2	+	70	68.44	+	67.5	59.12	-
Leddisplay	All	24	100				81			100		
	0.9	23.98	100	100	0E0	=	82.67	81.04	=	100	100	=
	0.7	17	100	100	0E0	=	92.33	91.09	+	100	100	=
	0.5	11.92	100	100	0E0	=	100	99.08	+	100	100	=
Dermatology	All	34	90				96.36			97.27		
	0.9	30.12	90	90	0E0	=	98.18	95.77	-	98.18	97.07	=
	0.7	11.62	93.64	88.16	1.33E0	-	96.36	93.98	-	96.36	93.7	-
	0.5	6.38	91.82	86.34	6.11E0	-	92.73	86.27	-	93.64	87.43	-
Soybeanlarge	All	35	90.73				88.29			88.29		
	0.9	22.78	90.73	89.8	97.4E-2	-	90.73	89.26	+	90.24	89.21	+
	0.7	9.68	88.78	84.11	2.49E0	-	84.39	80.26	-	88.78	86.49	-
	0.5	5.72	84.39	76.89	3.97E0	-	78.54	66.57	-	83.9	77.06	-
Chess	All	36	98.44				95.62			89.78		
	0.9	13.95	95.2	95.19	6.4E-2	-	95.1	94.76	-	93.22	92.12	+
	0.7	8.18	95.1	94.49	68.2E-2	-	95.2	94.22	-	94.99	93.89	+
	0.5	6.1	94.99	93.32	1.66E0	-	94.99	93.19	-	94.99	93.36	+
Connect4	All	42	74.62				73.48			72.23		
	0.9	9.68	70.36	69.4	60.7E-2	-	65.09	60.41	-	69.6	68.87	-
	0.7	7	68.73	67.6	54.2E-2	-	64.93	57.81	-	68.61	67.49	-
	0.5	5.12	67.52	66.55	50.6E-2	-	66.79	57.27	-	67.53	66.55	-
Lung	All	56	90				70			90		
	0.9	12.22	90	89.75	1.56E0	=	90	76	+	90	80.75	-
	0.7	7.7	90	81.25	12.9E0	-	90	82.5	+	90	82	-
	0.5	6.68	90	80.75	13.9E0	-	90	79.25	+	90	77.5	-
Musk1	All	166	71.33				83.92			42.66		
	0.9	42.3	79.02	71.68	3.68E0	=	83.92	79.14	-	75.52	70.23	+
	0.7	42.12	83.92	72.12	3.77E0	=	86.01	79.69	-	75.52	70.1	+
	0.5	42.02	83.92	71.99	3.63E0	=	86.01	79.63	-	79.02	70.33	+

features, is treated as less important with a large α_{mi} than with a small one. Note that in some cases, e.g. the Lymph, Mushroom and Leddisplay datasets, PSOMI using different α_{mi} values selected a different numbers of features, but achieved the same classification performance, which means the larger feature subsets still have redundancy. This is consistent with our hypothesis that redundant features can be further removed without reducing the classification performance.

Table 5.1 also shows that the results of PSOMI are basically general to the three classification algorithms. For example, on the Connect4 dataset, all the three algorithms did not perform well using the selected features. However, on the Spect and Leddisplay datasets, all the three algorithms achieved similar or better classification performance than using all features. Only on the Mushroom and Chess datasets, the results of the significant tests for NB are "+" in all cases, and for DT and KNN are "-" in all cases. One of the possible reasons is that the classification performance of DT or KNN using *all* features is already very high. This makes it difficult to improve the classification performance by using a very small feature subset, although the classification accuracy of DT and KNN using the small number of *selected* features is also high (around 95% or higher).

The results show that PSOMI using PSO as the search technique and mutual information as the evaluation criterion can reduce the number of features without decreasing or even increasing the classification performance. Meanwhile, the selected feature subsets are sufficiently general to the three classification algorithms.

5.4.2 Results of PSOE

According to Table 5.2, it can be seen that PSOE with entropy as the evaluation criterion can usually select a small number of features and achieve similar or even better classification performance than using all features. The number of features selected by PSOE is often less than half of the total

Table 5.2: Results of PSOE with Different α_e.

Dataset	α_e	Size	DT				KNN			NB		
			Best	Mean	Std	T	Best	Mean	T	Best	Mean	T
Lymph	All	18	82.22				82.22			80		
	0.9	9.58	80	78.72	1.1E0	-	80	77.45	-	77.78	77.28	-
	0.7	8.08	80	79.89	69.3E-2	-	77.78	76.61	-	77.78	77.5	-
	0.5	5.35	82.22	81.05	2.53E0	-	64.44	63.66	-	77.78	77	-
Mushroom	All	22	100				100			95.98		
	0.9	5.92	100	99.73	7.46E-2	-	99.88	99.72	-	95.63	91.35	-
	0.7	2.52	99.7	97.88	44.5E-2	-	99.7	97.88	-	97.76	97.39	+
	0.5	2.12	97.76	97.66	45.8E-2	-	97.76	97.66	-	97.76	97.44	+
Spect	All	22	66.25				63.75			70		
	0.9	17.02	71.25	70.88	1.32E0	+	72.5	70.16	+	73.75	72.56	+
	0.7	13	71.25	68	2.75E0	+	71.25	67.75	+	75	72.75	+
	0.5	7.42	71.25	68.56	3.53E0	+	72.5	64.22	=	73.75	72.41	+
Leddisplay	All	24	100				81			100		
	0.9	9	100	100	0E0	=	100	100	+	100	100	=
	0.7	9	100	100	0E0	=	100	100	+	100	100	=
	0.5	9	100	100	0E0	=	100	100	+	100	100	=
Dermatology	All	34	90				96.36			97.27		
	0.9	9.42	95.45	90.84	1.98E0	+	96.36	89.02	-	95.45	92.16	-
	0.7	7.68	93.64	90.27	1.31E0	=	94.55	86.34	-	93.64	90.57	-
	0.5	6.28	92.73	89.16	2.94E0	=	93.64	83.3	-	93.64	89.73	-
Soybeanlarge	All	35	90.73				88.29			88.29		
	0.9	20.72	88.29	82.94	2.88E0	-	82.44	76.55	-	87.32	81.71	-
	0.7	17.28	87.8	80.51	3.65E0	-	86.83	77.89	-	86.83	81.84	-
	0.5	13.68	89.27	83.74	3.28E0	-	86.34	78.83	-	88.29	82.11	-
Chess	All	36	98.44				95.62			89.78		
	0.9	25.82	99.06	98.91	27E-2	+	98.12	96.61	+	94.58	93.22	+
	0.7	21.38	99.37	98.81	31.1E-2	+	98.33	97.48	+	95.41	93.88	+
	0.5	16.82	98.54	98.01	63.2E-2	-	98.23	97.33	+	95.93	94.34	+
Connect4	All	42	74.62				73.48			72.23		
	0.9	37.92	75.9	74.66	73.9E-2	=	74.5	73.17	=	72.75	71.99	-
	0.7	38.12	76.89	74.63	1.05E0	=	74.2	72.9	-	72.55	71.73	-
	0.5	36.75	78.38	74.48	1.28E0	=	74.61	72.78	-	72.4	71.64	-
Lung	All	56	90				70			90		
	0.9	13.05	90	83.25	13.1E0	-	90	84.25	+	90	86	-
	0.7	13.05	90	83.5	12.8E0	-	90	83.75	+	90	86.25	-
	0.5	12.85	90	83.5	13.9E0	-	90	83.75	+	90	86.5	-
Musk1	All	166	71.33				83.92			42.66		
	0.9	60.55	76.58	69.11	4.42E0	-	81.98	76.4	-	68.17	64.04	+
	0.7	60.55	76.58	69.11	4.42E0	-	81.98	76.4	-	68.17	64.04	+
	0.5	60.55	76.92	71.98	3.03E0	=	85.31	78.85	-	77.62	71.92	+

number of features. In almost all cases, with the selected features, at least one of the three classification algorithms achieved similar or significantly higher accuracy than using all features. In some cases, the overall classification performance is lower than using all features, but the best accuracy is the same or higher than using all features.

In general, the number of features selected by PSOE decreased when α_e became smaller. The reason is that a smaller α_e means the redundancy measure, which indicates the number of features, was treated more important than a larger α_e. A smaller α_e also means that the relevance measure is treated as less important and accordingly, the classification performance decreased when α_e became smaller. There are many cases, where the number of features is reduced, but the classification performance is increased. This is because the larger feature subsets still have redundancy. The removal of redundant features does not decrease the classification performance, but may even increase the classification performance due to the reduction of the dimensionality and complexity.

In terms of the generality, the performance of the three classification algorithms is usually consistent. For some datasets, such as Spect and Led-display, all of the three algorithms using the selected features achieved similar or better performance than using all features. On the Dermatology dataset, DT using the selected features increased the classification performance, but KNN and NB did not. One possible reason is that KNN and NB using all features achieve very high classification accuracy and it is difficult to increase the performance with a very small number of features, which is the same case for the Mushroom and Lung datasets.

The results suggest that by using entropy as the evaluation criterion, PSOE can effectively search the solution space to obtain small feature subsets and maintain or even increase the classification performance over using all features. As a filter approach, the feature subsets selected by PSOE are sufficiently general to DT, KNN and NB.

5.4.3 Comparisons Between PSOMI and PSOE

Both PSOMI and PSOE are based on information theory. The fitness function in PSOMI is based on mutual information of each *pair* of features (or a feature and the class labels). The fitness function in PSOE is based on the information gain of a *group* of features (including the class labels). The fitness function is the main difference between PSOMI and PSOE, which leads to different performances in terms of the number of features and the classification performance.

Comparing Table 5.1 with Table 5.2, it can be seen that the (smallest) number of features is smaller in PSOMI on eight of the ten datasets and smaller in PSOE on the other two datasets. A possible reason is that the redundancy measure Red_e in PSOE is based on a group of features, which is less sensitive to the number of features than the pair-wise evaluation in the redundancy measure Red_{mi} of PSOMI. Adding more features will always increase (worse) Red_{mi} because Red_{mi} sums the mutual information between each pair of features. According to the calculation of Red_e, adding more features may not increase (worse) Red_e. However, since more features may increase the value of the relevance measure Rel_e, PSOE has a higher probability to obtain large feature subsets than PSOMI.

According to the results of the significance tests in Table 5.1, with the feature subsets selected by PSOMI, the total number of "+" and "=" is 11 for DT, 11 for KNN and 17 for NB, which indicate the number of cases, where the classification algorithm using the selected features achieved similar or better performance than using all features. According to Table 5.2, the numbers are 15, 13 and 14 for DT, KNN and NB, respectively. The total number in PSOE (42) is slightly larger than PSOMI (39). Meanwhile, it can be noticed that the numbers are more even in PSOE than in PSOMI, which means the performance of the three classification algorithms is more consistent in PSOE than in PSOMI. The reason may be that the relevance measure (Rel_e) in PSOE treats the feature subset as a whole, which might consider the interaction between them to select a subset of complementary

Table 5.3: Computational Time used by PSOMI and PSOE (In seconds).

Dataset	Lymph	Mushroom	Spect	Leddisplay	Dermatology	Soybeanlarge	Chess	Connect4	Lung	Musk1
PSOMI										
$\alpha_{mi} = 0.9$	0.13	0.67	0.27	0.23	0.24	0.38	0.57	102.57	0.34	15.17
$\alpha_{mi} = 0.7$	0.13	0.66	0.26	0.22	0.24	0.36	0.56	102.59	0.25	14.85
$\alpha_{mi} = 0.5$	0.12	0.65	0.25	0.22	0.22	0.35	0.56	102.6	0.26	14.86
PSOE										
$\alpha_e = 0.9$	3.8	149.88	12.21	35.13	17.51	55.31	358.04	19243.5	3.14	508.01
$\alpha_e = 0.7$	3.62	113.62	10.21	33.11	16.77	51.78	297.21	19793.3	3.08	497.01
$\alpha_e = 0.5$	3.19	121.51	8	34.31	15.42	48.66	256.17	18443.4	3.09	280.97

features, and does not bias to any classification algorithm. In contrast, the relevance measure (Rel_{mi}) in PSOMI evaluates the feature subsets by summing the mutual information between a single feature and the class labels, which treats features individually. So PSOMI is more likely to select features that individually work well, which better fits NB that assumes features are conditionally independent to each other. Therefore, NB performs much better than KNN and DT in PSOMI while there are no significant differences among the three algorithms in PSOE.

5.4.4 Comparisons on Computational Time

Table 5.3 shows the average computational time used by PSOMI and PSOE over the 40 independent runs for the evolutionary training process, where the time is expressed in seconds.

According to Table 5.3, it can be seen that on average, PSOMI can finish the evolutionary training process within 20 seconds except for the Connect4 dataset. PSOE can finish the evolutionary training process within 10 minutes (i.e. 600 seconds) except for the Connect4 dataset. Both PSOMI and PSOE used a much longer time on the Connect4 dataset than on all other nine datasets. The reason is that the Connect4 dataset has a much larger number of instances (44473), at least ten times or even a few hundreds more than the other datasets, which can be seen from Table 1.1 on Page 16.

PSOE clearly used a longer time than PSOMI. There are two main rea-

Table 5.4: Number of Appearance of Each Feature for the Chess Dataset.

Feature	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
PSOMI																																				
α_{mi}=0.9	9	8	22	16	6	24	5	27	6	40	5	10	5	18	30	31	17	14	19	5	40	16	14	15	21	0	17	17	25	14	8	28	40	3	9	1
α_{mi}=0.7	2	2	11	3	6	19	3	6	5	40	3	8	8	14	7	24	9	5	7	4	40	7	2	9	18	1	7	19	10	7	4	17	39	2	2	1
α_{mi}=0.5	3	7	10	9	1	6	1	3	4	39	3	7	8	7	1	13	5	6	7	5	40	0	3	8	11	1	2	15	6	3	2	11	31	0	1	0
PSOE																																				
α_e=0.9	40	29	40	40	1	40	40	1	6	40	0	20	8	38	40	40	40	9	34	32	40	21	40	10	38	9	40	33	33	40	15	40	40	40	40	16
α_e=0.7	40	22	31	26	2	40	35	0	1	40	0	10	1	36	40	38	40	4	35	2	40	6	40	2	33	1	32	37	31	29	0	40	40	36	40	5
α_e=0.5	35	8	15	6	1	40	0	0	1	40	0	3	0	39	39	23	40	18	35	0	40	5	40	0	37	0	8	39	29	20	0	29	40	3	40	0

sons why PSOE took a much longer running time than PSOMI. The first reason is that each calculation of the fitness function Fit_{mi} (according to Equation 5.7) in PSOMI needs a much shorter time than that of Fit_e (according to Equation 5.8) in PSOE. The second reason is that the number of calculations in terms of both the relevance measure and the redundant measure in PSOE is much larger than that of PSOMI. In PSOMI, the calculation of the possible mutual information between each feature and the class labels, i.e. $I(x;c)$ for Red_{mi}, and the possible mutual information between each pair of features, i.e. $I(x_i,x_j)$ for Red_{mi}, only needs to be performed once for each dataset before the evolutionary process. During the evolutionary training process, the calculation of Rel_{mi} and Red_{mi} only needs to refer to the values of $I(x;c)$ and $I(x_i,x_j)$, and then sums them. However, for $Rel_e = IG(c|X)$ and $Red_e = min(\frac{1}{|S|}\sum_{x\in X} IG(x|\{X/x\}))$ in PSOE, during the evolutionary training process, each particle has a different X. Therefore, each calculation of Rel_e or Rel_d needs to perform Equation 5.6, which takes a longer time than just calculating the sum in Rel_{mi} and Red_{mi}. Although PSOE took a longer time than PSOMI, both of them are relatively fast compared with the algorithms in Chapters 3 and 4 (wrappers).

5.4.5 Selected Features

Experimental results show that both PSOMI and PSOE are quite stable across different independent runs, where the most important feature is

always selected by all the algorithms in different runs. In order to show the stability of the proposed algorithms, we take the Chess dataset as an example as the other datasets show a similar pattern.

Both PSOMI and PSOE evolved one single feature subset in each run and thus 40 feature subsets were obtained from the 40 independent runs. Table 5.4 shows the number of appearance of each feature in the 40 feature subsets (40 runs) evolved by PSOMI and PSOE with α_{mi} and α_e as 0.9, 0.7 and 0.5. Note that PSOMI with $\alpha_{mi} = 0.5$ usually selected a small number of features (around 6 features, see Table 5.1), so the corresponding numbers in Table 5.4 are usually small. PSOE with $\alpha_e = 0.9$ usually selected a relatively large number of features (see Table 5.2) and the corresponding numbers in Table 5.4 are usually large.

According to Table 5.4, it can be seen that for the same relevance measure, in PSOMI with the three different α_{mi} values, Features 10 and 21 are the most frequently selected features. The results show that although different α_{mi} values lead to different results, Features 10 and 21 always have the largest chances to be selected by PSOMI. The most frequently selected features in PSOE with different α_e values are also the same, i.e. Features 6, 10, 17, 21, 23, 33 and 35. This indicates that both PSOMI and PSOE are reasonably stable algorithms.

Although the number of features selected by PSOE is much larger (almost twice) than that of PSOMI, not all of the features selected by PSOMI are included in PSOE. Due to different relevance and redundancy measures, some features that are selected frequently by PSOMI are not selected by PSOE. For example, Feature 8 is frequently selected by PSOMI with $\alpha_{mi} = 0.9$, but it is selected only once by PSOE with $\alpha_e = 0.9$ and never selected by PSOE with $\alpha_e = 0.7$ and 0.5. One possible reason is that Feature 8 is an individually relevant feature, but when combined with other features it brings redundancy to the feature subset. Therefore, Feature 8 is frequently selected by PSOMI using the relevance measure based on each individual feature, but not selected by PSOE using the relevance measure

based on a group of features.

5.4.6 Comparisons with Traditional Methods

Experiments have been conducted on the three traditional methods, CfsF, CfsB and GSBS. All the three classification algorithms have been used in the experiments and their results show a similar pattern. Therefore, the results of using KNN and NB as the classification algorithms are not listed here and only the results of DT are listed in Table 5.5.

Comparing Table 5.5 with Tables 5.1 and 5.2, it can be seen that both PSOMI and PSOE outperformed CfsF in most cases and outperformed CfsB in almost all cases in terms of both the classification performance and the number of features. Although on the Connect4 and Lung datasets, the average classification performance of PSOMI and PSOE is slightly worse than CfsF and CfsB, the best accuracies of them are better than CfsF and CfsB.

Note that it is not entirely fair to directly compare filter methods with wrapper methods since the wrapper methods use a classification/learning algorithm within the evaluation process. However, it does provide evidence to show that the filter methods are successful if they can outperform a wrapper method. Table 5.5 shows that GSBS as a wrapper method usually achieved better classification performance than the two filter algorithms (CfsF and CfsB). Comparing GSBS with PSOMI and PSOE, it can be seen that the number of features selected by GSBS is smaller than that of PSOMI on eight out of the 10 datasets and smaller than that of PSOE on five datasets. In terms of the classification performance, the best accuracy of PSOMI is better than GSBS on seven datasets and PSOE is better than GSBS on eight datasets, which shows that PSOMI and PSOE as filter methods have the potential to achieve better classification performance than a traditional wrapper method.

In terms of the computational cost, CfsF, CfsB and PSOMI are faster

Table 5.5: Results of CfsF, CfsB and GSBS.

	Lymph				Mushroom				Spect				Leddisplay				Dermatology			
Method	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS
Size	18	8	8	2	22	3	3	5	22	4	4	6	24	13	13	5	33	17	17	7
Accuracy	82.2	73.3	73.3	77.8	100	97.7	97.7	100	66.3	70	70	67.5	100	100	100	100	90	87.3	87.3	90

	Soybean Large				Chess				Connect4				Lung				Musk1			
Method	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS	All	CfsF	CfsB	GSBS
Size	35	12	14	12	36	5	5	17	42	6	6	28	56	6	11	33	166	36	41	122
Accuracy	90.7	80.5	85.4	90.2	98.4	78.1	78.1	99.1	74.6	70.3	70.3	78.8	90	90	90	90	71.3	71.3	70.6	74.8

than PSOE because PSOE involves a relative complex fitness function, but all of them can finish the feature selection process very fast. Since each evaluation in GSBS involves a training and a testing processes, GSBS is usually slower than all the other four algorithms, especially on the large datasets.

5.5 Chapter Summary

The goal of this chapter was to investigate the use of information theory in PSO for filter feature selection to reduce the number of features and maintain or even increase the classification performance of using all features. The goal was successfully achieved by developing two new PSO based filter algorithms using mutual information (PSOMI) and entropy (PSOE) in the fitness functions to evaluate the goodness of the selected features. PSOMI and PSOE successfully reduced the number of features and maintained or even improved the classification accuracy. They outperformed two traditional filter feature selection algorithms and even achieved slightly better performance than a traditional wrapper feature selection method. Meanwhile, the feature subsets selected by PSOMI and PSOE are general to the three classification algorithms, i.e. DT, KNN and NB.

This chapter shows that both PSOMI and PSOE are fast algorithms due to the filter fitness functions, which do not need a learning/classification process for each evaluation (like in wrappers). PSOMI based on mutual

information of each pair of features is faster and selects a smaller number of features than PSOE using the information gain of a group of features. However, the pair evaluation in PSOMI considers less feature interaction than the group evaluation in PSOE. PSOMI has a higher probability to select individually good features while PSOE is more likely to select a group of complementary features. Therefore, the classification performance of the PSOE is slightly better than PSOMI. The features selected by PSOMI were noticed to slightly favour to NB, which is based on the assumption that features are (conditionally) independent to each other.

In PSOMI and PSOE, the classification performance is indicated by the relevance of the features and the number of features is indirectly shown by the redundancy among the feature subset. A weight was used to balance the relative importance of the relevance and redundancy in the fitness function. A large weight for redundancy in PSOMI or PSOE can further reduce the number of features, but it does not always reduce the classification performance, and may even increase the classification performance due to the removal of unnecessary complexity. However, it is difficult to pre-determine the best value of the weight.

This problem can be avoided if they are treated as two separate objective functions rather than combining them into a single fitness function. Meanwhile, treating the relevance and redundancy (or the number of features) as two separate objectives in multi-objective feature selection is hypothesised to better solve the task to obtain a set of non-dominated solutions instead of a single solution, where the obtained Pareto front can assist users in choosing their preferred solutions to meet their own requirements. However, evolutionary computation techniques, including multi-objective PSO, NSGAII and SPEA2, have never been applied to filter based multi-objective feature selection. Therefore, in the next chapter, we will investigate the use of evolutionary computation techniques and information theory in filter based multi-objective feature selection.

Chapter 6

Filter Based Multi-Objective Feature Selection

6.1 Introduction

Feature selection, by its nature, is a multi-objective task, which aims to maximise the classification performance and minimise the number of features. However, there are only a few works treating it as a multi-objective problem and almost all of them are wrapper approaches. Filter approaches are argued to be computational less expensive and more general than wrapper approaches, but no work has been conducted on multi-objective filter feature selection.

To develop a multi-objective filter feature selection algorithm, two key factors are needed. The first one is an evaluation measure, which determines the goodness of the selected feature subsets. The second one is a search technique, which searches the solution space to find the optimal feature subsets. In Chapter 5, mutual information and entropy in information theory have been shown to be effective evaluation measures in filter feature selection. Multi-objective EC techniques, such as non-dominated sorting based multi-objective genetic algorithm II (NSGAII) [23], strength Pareto evolutionary algorithm 2 (SPEA2) [24] and multi-

objective PSO, have been widely used in many areas [176]. In Chapter 4, we have shown that the two multi-objective PSO algorithms (i.e. NSPSO [103] and CMDPSO [104]), and the NSGAII and SPEA2 algorithms can be successfully used in multi-objective wrapper feature selection. However, the use of such multi-objective EC algorithms in filter feature selection has not been investigated to date.

6.1.1 Chapter Goals

The overall goal of this chapter is to use EC techniques and information theory to develop a multi-objective, filter feature selection approach to searching for a set of non-dominated solutions (feature subsets). The selected feature subsets are expected to include a smaller number of features and achieve similar or even better classification performance than using all features.

To achieve this goal, four multi-objective feature selection frameworks are developed based on four multi-objective EC algorithms, which are NSPSO, CMDPSO, NSGAII and SPEA2. Two information measures, which are the mutual information measure and entropy measure, are used in each framework to develop two filter multi-objective algorithms. Thus eight multi-objective feature selection algorithms will be proposed by applying the two information measures to the four frameworks. These proposed algorithms will be examined and compared with the single objective algorithms developed in Chapter 5 on eight benchmark problems of varying difficulty. Specifically, we will investigate:

- whether NSPSO based multi-objective feature selection algorithms can evolve a set of good feature subsets, which include a smaller number of features and achieve better classification performance than using all features, and can outperform PSO based single objective algorithms;
- whether CMDPSO based multi-objective feature selection algorithms

can evolve a set of good feature subsets, and can achieve better performance than PSO based single objective algorithms and NSPSO based algorithms;

- whether NSGAII based multi-objective feature selection algorithms can evolve a set of good feature subsets, and can achieve better performance than the PSO algorithms above, and
- whether SPEA2 based multi-objective feature selection algorithms can evolve a set of good feature subsets, and can outperform all other methods mentioned above.

Note that this thesis mainly focuses on the use of PSO for feature selection. The NSGAII and SPEA2 based approaches to feature selection are mainly proposed to investigate whether NSGAII and SPEA2 can be directly applied to feature selection and can achieve better results than the PSO based approaches.

6.1.2 Chapter Organisation

The remainder of this chapter is organised as follows. The second section describes the proposed filter based multi-objective feature selection algorithms. The third section describes the design of the experiments. The results and discussions are presented in the fourth section. The fifth section provides a summary of this chapter.

6.2 Proposed Algorithms

This section presents the multi-objective filter feature selection algorithms, which starts from the fitness functions, and the representation used in this chapter, then describes the proposed multi-objective algorithms, which are based on NSPSO, CMDPSO, NSGAII and SPEA2.

Fitness functions. In filter approaches, the classification performance is represented by a relevance measure. Generally, a multi-objective filter feature selection algorithm aims to maximise the relevance measure (i.e. representing the classification performance) and minimise the number of features. In this chapter, the relevance of the selected features will be evaluated by the two measures developed in Chapter 5, which are the mutual information measure (Rel_{mi}) shown by Equation 5.1 (Page 135), and the entropy measure (Rel_e) shown by Equation 5.4 on Page 138.

When using the mutual information based measure (Rel_{mi}), the fitness function of a filter, multi-objective feature selection algorithm is shown by Equation 6.1.

$$F_{mi}(X) = [Rel_{mi}(X, c), Size(X)] \tag{6.1}$$

where

$$Rel_{mi}(X, c) = max(\sum_{x \in X} I(x; c)),$$
$$Size(X) = min(|X|)$$

where X means a subset of features, $X = (x_1, x_2, x_3, ...)$. c shows the class labels. $|X|$ represents the number of features in X. The detailed explaination and calculation of Rel_{mi} are shown by Equation 5.1 on Page 135.

When using the entropy based measure (Rel_e), the fitness function of a filter, multi-objective feature selection algorithm is shown by Equation 6.2. The detailed explaination and calculation of Rel_e are shown by Equation 5.4 on Page 138.

$$F_e(X) = [Rel_e(X, c), Size(X)] \tag{6.2}$$

where

$$Rel_e(X, c) = max((IG(c|X))$$
$$Size(X) = min(|X|)$$

Representations. In the proposed algorithms, each individual in the population/swarm, i.e. a particle in NSPSO and CMDPSO or a chromosome in NSAGAII and SPEA2, is represented by a n-bit binary string, where n is the total number of available features in the dataset. n is also the dimensionality of the search space. In the binary string, "1" represents that the corresponding feature is selected and "0" otherwise.

6.2.1 New Algorithms: NSMI and NSE

PSOMI and PSOE in Chapter 5 with the mutual information measure and the entropy measure can be successfully used for filter feature selection. However, the weights in the fitness functions of PSOMI and PSOE need to be predefined. This problem is avoid by using multi-objective feature selection. NSPSO [103] is a simple multi-objective PSO based on the idea of fast non-dominated sorting in NSGAII. Based on NSPSO, a multi-objective filter feature selection framework is investigated in this section. Two multi-objective filter feature selection algorithms are then proposed based on NSPSO and the *mutual information* measure and the *entropy* measure, which are named as *NSMI* and *NSE*, respectively.

NSMI is based on the mutual information measure and aims to optimise the objective function shown by Equation 6.1. NSE is based on the entropy measure and aims to optimise the objective function shown by Equation 6.2. Their main difference is the fitness function, which is also the main difference between the two *filter* algorithms, NSMI or NSE, and the multi-objective *wrapper* feature selection algorithm (NSPSOFS) developed in Chapter 4 on Page 107. Figure 6.1 shows the flow chart of both NSMI and NSE. In Figure 6.1, the key steps are coloured in gray. The main idea is to use a non-dominated sorting mechanism (Step 7) to select a $gbest$ for each particle and update the swarm during the evolutionary process.

As shown in Figure 6.1, in each iteration, the algorithms select a $gbest$ for each particle from the non-dominated solutions in the swarm (Step 2).

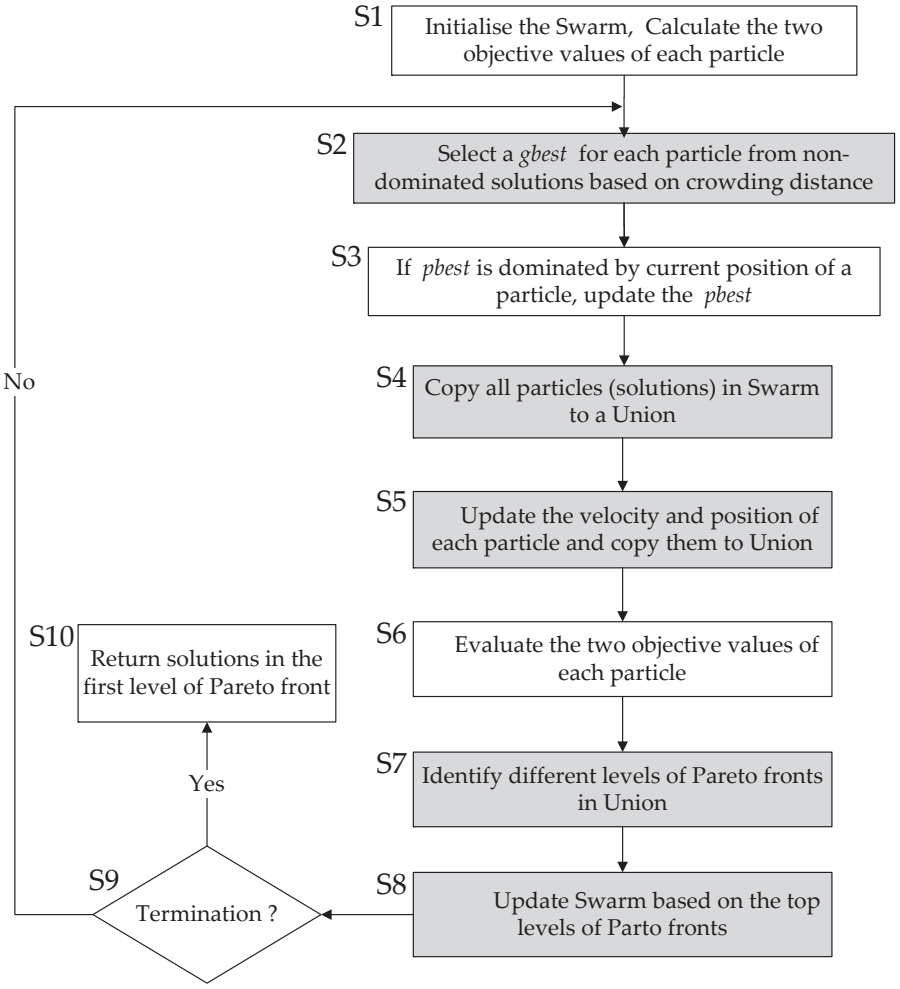

Figure 6.1: The flowchart of NSMI and NSE (showing Steps (S) 1-10).

Specifically, the algorithms identify the non-dominated solutions in the swarm and calculate the crowding distance, then all the non-dominated solutions are sorted according to the crowding distance. Then a $gbest$ is randomly selected from the least crowded solutions (the highest ranked part) of the sorted non-dominated solutions. $pbest$ of each particle is then determined in Step 3. To update the swarm for the next iteration, all the particles in the swarm are firstly copied to a union in Step 4. The velocity and the position of each particle are updated according the Equations 2.2

and 2.4 on Page 38 and the updated particles are also added into the union (Step 5). The two objective values of each particle are evaluated in Step 6. Step 7 shows the non-dominated sorting of the solutions in the union. Specifically, the non-dominated solutions in the union are called the first non-dominated front, subsequently excluded from the union. Then the non-dominated solutions in the new union are called the second non-dominated front. The following levels of non-dominated fronts are identified by repeating this procedure. Step 8 shows the process of updating the swarm for the next iteration. Specifically, particles are selected from the top levels of the non-dominated fronts, starting from the first front. If the number of solutions needed is larger than the number of solutions in the current non-dominated front, all the solutions are added into the next iteration. Otherwise, the solutions in the current non-dominated front are ranked according to the crowding distance and the highest ranked solutions are added into the next iteration. Steps 2 to 8 are repeated until the termination criteria is met. The algorithms return the non-dominated solutions in the union, which are also the first level of Pareto front achieved by the non-dominated sorting in Step 7.

6.2.2 New Algorithms: CMDMI and CMDE

Multi-objective PSO (CMDPSO [104]) using the ideas of crowding, mutation and dominance is able to keep the diversity of the swarm, which is particularly important for feature selection problems, since they have many local optima in the search space. To further investigate the use of PSO in feature selection, a multi-objective feature selection framework is investigated based on CMDPSO. Two multi-objective filter feature selection algorithms are then proposed based on this framework and the mutual information measure and the entropy measure, which are named as CMDMI and CMDE, respectively. *CMDMI* using the *mutual information* measure aims to optimise the objective function shown by Equation 6.1.

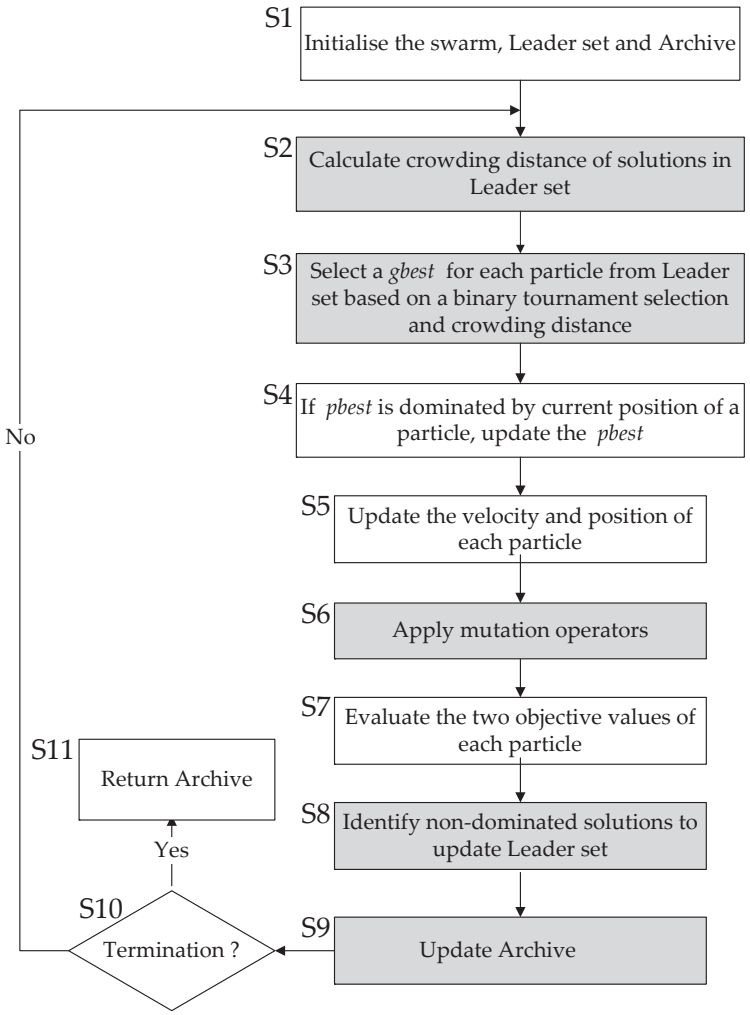

Figure 6.2: The flowchart of CMDMI and CMDE.

CMDE using the *entropy* measure and aims to optimise the objective function shown by Equation 6.2. The main difference between CMDMI and CMDE is the fitness function, which is also the main difference between these two *filter* algorithms and the multi-objective *wrapper* feature selection algorithm (CMDPSOFS) developed in Chapter 4 on Page 110.

Figure 6.2 shows the flow chart of both CMDMI and CMDE. Basically, CMDMI and CMDE follow the basic steps of the PSO algorithm except for the steps related to the selection of $gbest$, mutation and dominance,

which are shown in gray colour. In order to address the main issue of determining a good leader ($gbest$), CMDMI and CMDE employ a leader set to keep the non-dominated solutions as the potential leaders for each particle. A crowding factor (Step 2) is employed to decide which non-dominated solutions should be added into the leader set and kept during the evolutionary process. Step 3 shows the selection of $gbest$ for each particle. Specifically, a binary tournament selection is performed to choose two solutions from the leader set and the less crowded one is selected as $gbest$. After updating $pbest$ in Step 4, CMDMI and CMDE update the position and velocity of each particle in Step 5 and mutation operators are applied in Step 6 to maintain the diversity of the swarm. CMDMI and CMDE identify the non-dominated solutions and then update the leader set in Step 8 and update the archive in Step 9. Steps 2 to 9 are repeated until the termination criterion is met. The algorithms return the solutions in the archive, where the number of non-dominated solutions is determined by the dominance factor.

6.2.3 New Algorithms: NSGAIIMI and NSGAIIE

NSGAII has been successfully used in many areas [176]. However, it has never been directly applied to filter based feature selection. In this section, we develop a multi-objective, filter feature selection framework based on NSGAII. Further, two new multi-objective, filter feature selection algorithms, NSGAIIMI and NSGAIIE, are proposed by applying the mutual information measure and the entropy measure. *NSGAIIMI* using the *mutual information* measure aims to optimise the objective function shown by Equation 6.1. *NSGAIIE* using the *entropy* measure aims to optimise the objective function shown by Equation 6.2.

The main principle of NSGAII is the use of fast non-dominated sorting technique and the diversity preservation strategy. The fast non-dominated sorting technique is used to rank the parent and child populations to dif-

Input : A Training set and a Test set;
Output: A set of non-dominated solutions, training and test accuracies.

begin
  Initialise $Population$ based on S (Population size) and D (Dimensionality, number of features);
  Evaluate two objectives of each individual ; `/* number of features and the relevance (` Rel_{mi} `in NSGAIIMI and` Rel_e `in NSGAIIE) on the Training set */`
  Generate $Child$ (new population) by conducting selection, crossover and mutation operators;
  while *$Maximum\ Number\ of\ Generations$ is not reached* **do**
    Evaluate two objectives of each individual in new $Child$;
    Merge $Child$ and $Population$ to $Union$;
    Empty $Population$ and $Child$ for new generation;
    Identify different levels of non-dominated fronts $F = (F_1, F_2, F_3, \ ...)$ in $Union$; `/* Fast non-dominated sorting */`
    while $|Population| < S$ **do**
      if $|Population| + |F_i| \leq S$ **then**
        Calculate crowding distance of each individual in F_i;
        Add F_i to $Population$;
        $i = i + 1$;
      else
        Calculate crowding distance of each particle in F_i;
        Sort particles in F_i;
        Add the $(S - |Population|)$ least crowded particles to $Population$;
    Generate $Child$ (new population) by conducting selection, crossover and mutation operators;
  Calculate the number of features in each solution in F_1;
  Calculate the classification error rate of the solutions (feature subsets) in F_1 on the test set ; `/*` F_1 `is the achieved Pareto front */`
  Return the solutions in F_1;
  Return the number of features and the test classification error rate of each solution in F_1;

Algorithm 7: Pseudo-Code of NSGAIIMI and NSGAIIE

ferent levels of non-dominated solution fronts. A density estimation based on the crowding distance is adopted to keep the diversity of the population. More details can be seen in the literature [23]. Algorithm 7 shows the pseudo-code of NSGAIIMI and NSGAIIE. After the intilisation and the evaluation of individuals, a child population is generated by applying selection, crossover and mutation operators. Line 7 shows the idea of merging the parent and child populations into a union. Then, the fast non-dominated sorting is performed to identify different levels of Pareto fronts in the union (in Line 9), which is the same as described in Step 7 in NSMI and NSE. In this procedure, the non-dominated solutions in the union are called the first non-dominated front, which are then excluded from the union. Then the non-dominated solutions in the new union are called the second non-dominated front. The following levels of non-dominated fronts are identified by repeating this procedure. For the next generation, solutions (individuals) are selected from the top levels of the non-dominated fronts, starting from the first front (from Line 10 to Line 18). When selecting individuals for the new generation, crowding distance is adopted to keep the diversity of the population, which can be seen in Lines 12 and 16. The algorithms repeat the procedures from Line 5 to Line 19 until the predefined maximum generation has been reached.

6.2.4 New Algorithms: SPEA2MI and SPEA2E

In order to further investigate the use of multi-objective EC techniques for filter based feature selection, we propose another multi-objective feature selection framework based on SPEA2, which has never been directly applied to filter based feature selection. Further, the mutual information measure and the entropy measure are applied to this framework to propose two new filter multi-objective algorithms, SPEA2MI and SPEA2E. *SPEA2MI* using the *mutual information* measure aims to optimise the objective function shown by Equation 6.1. *SPEA2E* using the *entropy* measure

Input : A Training set and a Test set;
Output: A set of non-dominated solutions, training and test accuracies.

begin
  Initialise the $Population$ based on S (Population size) and D (Dimensionality, number of features);
  Create the $Archive$ (empty);
  while $Maximum\ Number\ of\ Generations$ *is not reached* **do**
    Evaluate two objectives of each individual ; `/* number of features and the relevance (` Rel_{mi} `in SPEA2MI and` Rel_e `in SPEA2E) on the Training set */`
    Merge $Population$ and $Archive$ to $Union$;
    Calculate the raw fitness of each individual in $Union$;
    Calculate the density of each individual in $Union$;
    Calculate the fitness of each individual in $Union$; `/* fitness is the sum of the raw fitness and the density value */`
    Identify the non-dominated solutions in $Union$ and add them to $Archive$;
    if $|Archive| < Maximum\ Archive\ Size$ **then**
      Add the non-dominated solutions from the remaining $Population$ to $Archive$; `/* Remaining` $Population$ `excludes the non-dominated solutions that have already been added to` $Archive$ `*/`
    else if $|Archive| > Maximum\ Archive\ Size$ **then**
      Remove similar solutions to reduce the size of $Archive$;
    Generate new $Population$ by performing crossover and mutation operators based on $Archive$ and $Population$;
  Calculate the number of features in each solution in $Archive$;
  Calculate the classification error rate of the solutions in $Archive$ on the test set;
  Return the solutions in $Archive$;
  Return the number of features and the test classification error rate of each solution in $Archive$;

Algorithm 8: Pseudo-Code of SPEA2MI and SPEA2E

aims to optimise the objective function shown by Equation 6.2.

Algorithm 8 shows the pseudo-code of SPEA2MI and SPEA2E. The main principle of SPEA2 is the fine-gained fitness assignment strategy and the use of an archive truncation method. The fine-gained fitness assignment is shown from Line 7 to Line 9, where the fitness of each individual is the sum of its raw fitness and a density estimation. Line 3 shows the intilisation of the archive. The updating process of the archive can be seen from Line 10 to Line 13. When the number of non-dominated solutions is larger than the predefined maximum archive size, the archive truncation method is applied to determine whether a non-dominated solution should be included in the archive (Line 14). This archive truncation method is based on a similarity measure, which is the distance between each solution and its neighbours. A new population is constructed by the non-dominated solutions in both the original population and the archive (Line 15). The algorithms will repeat the procedures from Line 4 to Line 15 until the predefined maximum number of generations is reached.

6.3 Design of Experiments

6.3.1 Benchmark Techniques

To test the performance of the proposed multi-objective algorithms, the single objective algorithms developed in Chapter 5, i.e. PSOMI and PSOE, are used as benchmark techniques. In PSOE and PSOMI, a weight (α_{mi} and α_e) is used to balance the relative importance of the classification performance (relevance measure) and the number of features (redundancy measure). α_{mi} and α_e are in [0.5, 1.0) because the classification performance is usually more important (at least the same) than the number of features.

In Chapter 5, five different values of α_{mi} and α_e were used in the experiments, which are 0.5, 0.6, 0.7, 0.8, and 0.9. Among these five val-

ues, the classification performance is treated as the most important when $\alpha_{mi} = \alpha_e = 0.9$ and the number of features is treated as the most important when $\alpha_{mi} = \alpha_e = 0.5$. Therefore, PSOMI with $\alpha_{mi} = 0.5$ and 0.9, and PSOE with $\alpha_e = 0.5$ and 0.9 are chosen in this chapter to test the performance of the multi-objective feature selection approaches.

6.3.2 Datasets and Parameter Settings

Eight datasets [25] of varying difficulty are used in the experiments, which can be seen from Table 1.1 on Page 16 and they are the same as in Chapter 5 for comparisons purposes. The Musk1 and Lung datasets are not used here because these two datasets have a relatively small number of instances and can not sufficiently show the capability of the algorithms for feature selection. All the data in the selected datasets are categorical values because mutual information and entropy are mainly used for discrete variables.

In the experiments, all the instances in each dataset are randomly divided into two sets: 70% as the training set and 30% as the test set, which is the same as in the previous chapters. The filter feature selection algorithms firstly run on the training set to select feature subsets and then the classification performance of the selected features will be evaluated on the test set by a classification algorithm. In the filter approaches, the training process is independent of any classification algorithm. In the testing process, any algorithm can be used here. To evaluate the claim that filter approaches are general to different classification algorithms, three commonly used algorithms, DT, NB, and KNN with K=5, are used to evaluate the classification performance of the selected features. Since the results of all the three classification algorithms show a similar pattern, only the results of DT are presented in the next section. These settings are the same as in Chapter 5 for consistency and comparison purposes.

A library named jMetal [177] is used in the experiments. The parame-

ters of NSPSO based algorithms, NSMI and NSE, are set as follows: inertia weight $w = 0.7298$, acceleration constants $c_1 = c_2 = 1.49618$, maximum velocity $v_{max} = 6.0$, population size is 30, the maximum number of iterations is 500. The fully connected topology is used. These values are chosen based on the common settings in the literature [84, 89]. In CMDMI and CMDE, w is a random value in [0.1,0.5], c_1 and c_2 are random values in [1.5, 2.0], and the mutation rate is $1/n$, where n is the number of available features (dimensionality). These values are based on the settings of CMDPSO [104].

In the NSGAII and SPEA2 based algorithms, NSGAIIMI, NSGAIIE, SPEA2MI and SPEA2E, the population size and the maximum number of generations are set the same as the PSO based algorithms for comparison purposes. A bit-flip mutation operator and single point crossover operator are applied. The mutation rate is the same as in the PSO based algorithms, i.e. $1/n$, where n is the total number of features in the dataset. The crossover probability is 0.9. Other parameters are set as the default values in the jMetal library since they can lead to good results. For each dataset, all the algorithms have been conducted for 40 independent runs.

6.4 Results and Discussions

In this section, we firstly discuss the results of multi-objective algorithms using the mutual information measure, which are NSMI, CMDMI, NSGAIIMI and SPEA2MI. Then the performance of the entropy measure based algorithms are discussed, which are NSE, CMDE, NSGAIIE and SPEA2E. Finally, the computational time and further discussions are given.

For each dataset, PSOMI and PSOE obtain a single solution in each of the 40 independent runs. The multi-objective algorithms obtain a set of non-dominated solutions in each run. In order to compare these two kinds of results, 40 sets of feature subsets achieved by a multi-objective algorithm are firstly combined into an union set. In the union set, for the

feature subsets including the same number of features (e.g. m), their classification error rates are averaged. The average classification error rate is assigned as the average classification performance of the subsets with m features. Therefore, a set of average solutions are obtained by using the average classification error rates and the corresponding numbers (e.g. m). The set of average solutions is called the *average* front and presented here. Besides the average front, the non-dominated solutions in the union set are also presented to compare with the solutions achieved by the single objective algorithms. This is the same as the comparisons conducted in Chapter 4.

6.4.1 Results of NSMI and CMDMI

Figure 6.3 compares the results of NSMI, CMDMI, and PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$, which employ the *mutual information* measure to evaluate the relevancy and redundancy between a pair of features. In filter feature selection approaches, the performance of the solutions are evaluated by its classification performance on the unseen test data. The results in Figure 6.3 are the Pareto front solutions obtained in the *mutual information* measure space, but their classification performances shown in the figure were evaluated by DT on the test set in each dataset.

In Figure 6.3, on the top of each chart, the numbers in the brackets show the number of the available features and the classification error rate using all features. In each chart, the horizontal axis shows the number of features selected and the vertical axis shows the classification error rate evaluated by DT. In Figure 6.3, "NSMI-A" stands for the average front resulting from NSMI in the 40 independent runs. "NSMI-B" represents the non-dominated solutions resulting from NSMI in the 40 independent runs. $\alpha_{mi} = 0.5$ means the 40 solutions of PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$ means the 40 solutions of PSOMI with $\alpha_{mi} = 0.9$.

In some datasets, PSOMI and PSOE may evolve the same feature sub-

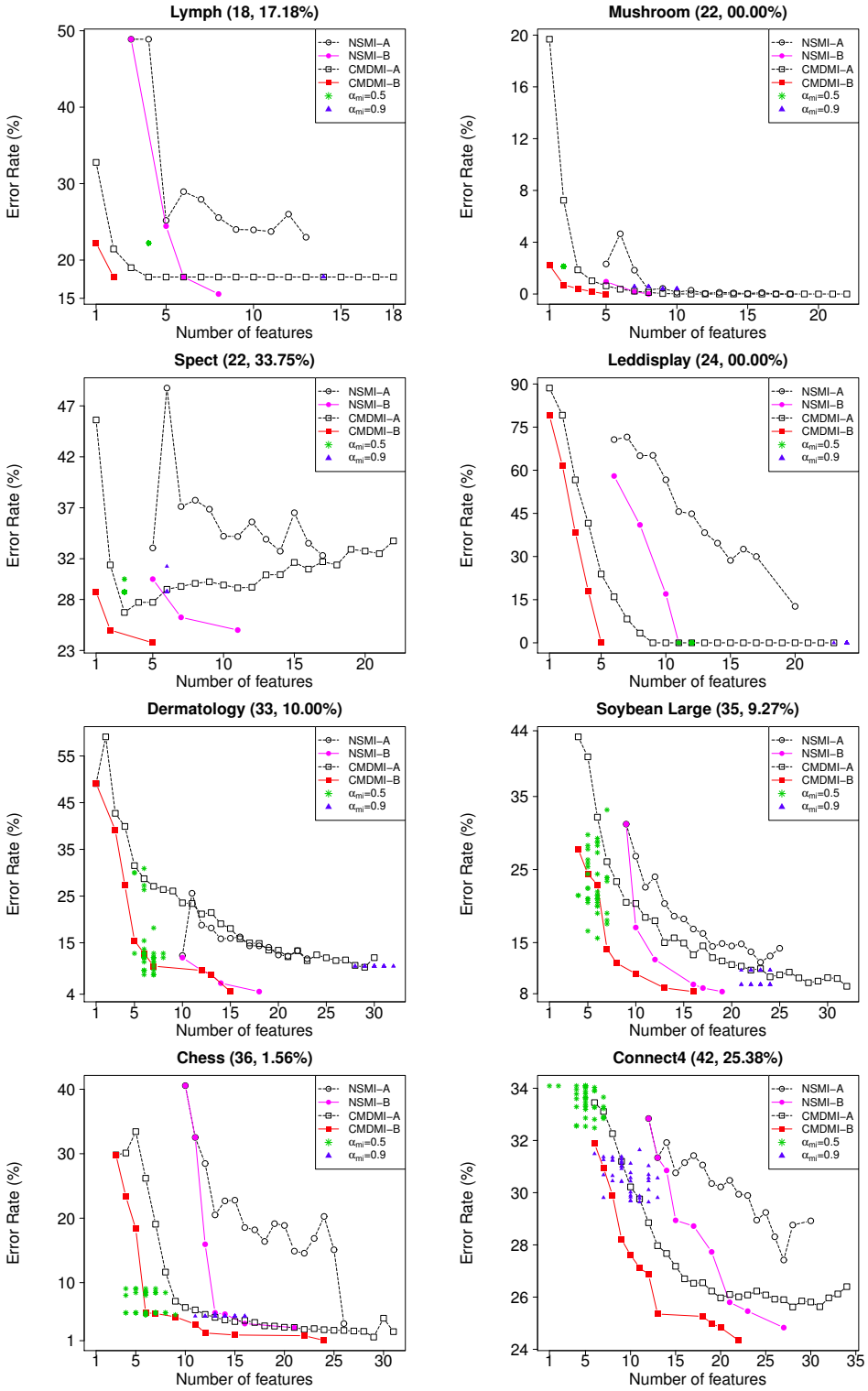

Figure 6.3: Experimental Results of PSOMI, NSMI and CMDMI.

set in different runs and they are shown in the same point in the chart. Therefore, although 40 results are presented, there may be less than 40 distinct points shown in a chart. For "NSMI-B" and "CMDMI-B", each of these non-dominated solution sets may also have duplicate feature subsets. They are also shown in the same point in the chart. This is also the case for other multi-objective algorithms. For the same number of features, there are a variety of combinations of features with different classification performances. In different runs, NSMI or NSE may select the same number of features with the same fitness evaluated by mutual information (Equation 5.1), but the same (or better) goodness does not necessarily result in the same (or better) classification performance. Therefore, they may have different classification error rates. Although NSMI or CMDMI obtained a set of non-dominated solutions in each run, the average solutions in NSMI-A or CMDMI-A may dominate each other (This also happens in other multi-objective algorithms).

Results of NSMI

According to Figure 6.3, on the Mushroom and Spect datasets, the average fronts of NSMI (NSMI-A) contain two or more solutions that selected a smaller number of features and achieved the same or a lower classification error rate than using all features. In almost all datasets, the non-dominated solutions (NSMI-B) include one or more feature subsets, which included less than 50% of the available features and achieved better classification performance than using all features. For example, on the Spect dataset, one non-dominated solution selected 11 features from the 22 available features and the classification error rate was decreased from 33.75% to 25%.

The results suggest that NSMI as a multi-objective algorithm can effectively search the solution space and automatically evolve a set of feature subsets to reduce the number of features and improve the classification performance.

Results of CMDMI

According to Figure 6.3, the average fronts of CMDMI (CMDMI-A) include two or more solutions that selected a smaller number of features and achieved better classification performance than using all features on *all* datasets (or similar classification performance only on the Connect4 dataset). On almost all datasets (except for the Soybean Large dataset), CMDMI-B evolved feature subsets including less than one third of the available features and achieved better classification performance than using all features.

Figure 6.3 also shows that achieving better classification performance usually needs more features, but there are occasionally some feature subsets that include a smaller number of features and achieve better classification performance. For example, on the Spect dataset, CMDMI-B selected only one feature and decreased the classification error rate from 33.75% to 28.75%.

The results suggest that as a multi-objective algorithm, CMDMI can effectively explore the Pareto front of a feature selection problem to reduce both the classification error rate and the number of features needed for classification.

Comparisons between PSOMI, NSMI, and CMDMI

Comparing NSMI with PSOMI, as can be seen in Figure 6.3, in most cases, NSMI (NSMI-B) achieved lower classification error rates than PSOMI with $\alpha_{mi} = 0.5$, although the number of features is slightly larger. In most cases, NSMI (NSMI-B) outperformed PSOMI with $\alpha_{mi} = 0.9$ in terms of both the number of features and the classification performance.

Comparing CMDMI with PSOMI, in almost all cases, feature subsets evolved by CMDMI (CMDMI-B) achieved better performance than feature subsets evolved by PSOMI with $\alpha_{mi} = 0.5$ and with $\alpha_{mi} = 0.9$ in terms of both the number of features and the classification performance. The

comparisons show that NSMI and CMDMI as multi-objective algorithms could better explore the solution space than the single objective algorithm, PSOMI. Both NSMI and CMDMI can obtain non-dominated feature subsets that use a smaller number of features and achieve better classification performance.

In *all* datasets, CMDMI achieved better performance than NSMI in terms of both the classification performance and the number of features. The main reasons are that feature selection tasks are difficult problems with many local optima. CMDMI employs different mechanisms to maintain the diversity of both the leader set and the swarm. Specifically, it selects and filters out crowded leaders and uses different mutation operators to maintain the diversity of the swarm to avoid stagnation in local optima. By contrast, NSMI is less effective than CMDMI in terms of avoiding stagnation in local optima. NSMI employs different levels of Pareto fronts to store the already found non-dominated solutions. Therefore, all the non-dominated solutions will be kept in the swarm from iteration to iteration. Such non-dominated solutions may be duplicated and the swarm may lose diversity quickly, which will lead to the problem of premature convergence.

6.4.2 Results of NSGAIIMI and SPEA2MI

Figure 6.4 compares the results of NSGAIIMI, SPEA2MI and PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$, which employ the *mutual information* measure to evaluate the relevancy of the selected features.

Results of NSGAIIMI

According to Figure 6.4, in *all* datasets, the average front of NSGAIIMI, NSGAIIMI-A, contained one or more solutions that selected a smaller number of features and achieved similar or even better classification performance than using all features. In *all* cases, feature subsets in NSGAIIMI-B

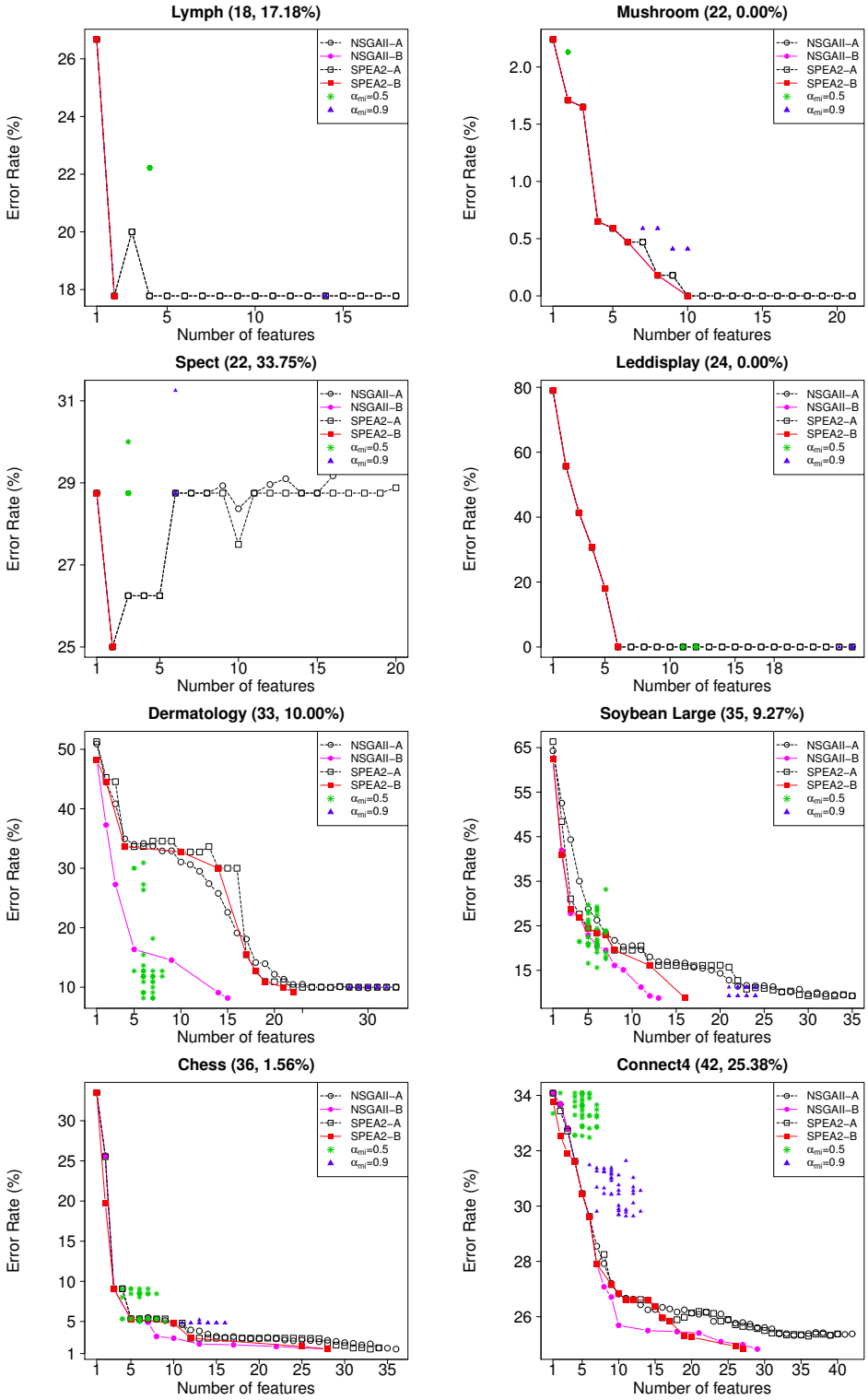

Figure 6.4: Experimental Results of PSOMI, NSGAIIMI and SPEA2MI.

selected less than half of the available features and achieved similar or better classification performance than using all features. For example, on the Spect dataset, NSGAIIMI-B selected only one feature and improved the classification performance over using all features. The results show that NSGAIIMI can be successfully used to address feature selection problems.

Results of SPEA2MI

According to Figure 6.4, in *all* datasets, SPEA2MI-A includes one or more feature subsets that selected a small number of features with which DT achieved better classification performance than with all features. In *all* datasets, SPEA2MI-B achieved better classification performance than using all features by selecting only less than half of the available features. The results show that SPEA2MI can be successfully used to address feature selection problems.

Comparisons Between PSOMI, NSGAIIMI and SPEA2MI

Comparing NSGAIIMI and SPEA2MI with PSOMI in Figure 6.4, it can be seen that in most cases, feature subsets in NSGAIIMI-A, NSGAIIMI-B, SPEA2MI-A and SPEA2MI-B outperformed PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$ in terms of both the number of features and the classification performance.

On the Lymph, Mushroom, Spect and Leddisplay datasets, similar results (almost the same results) are shown for NSGAIIMI and SPEA2MI in terms of both the number of features and the classification performance. The main reason is that all these four datasets have a relatively small number of features and both NSGAIIMI and SPEA2MI can obtain the good solutions. However, NSGAIIMI and SPEA2MI also obtained other different solutions which are not shown in the figure because they are dominated by the solutions presented in the figure.

The results in Figure 6.4 suggest that NSGAIIMI and SPEA2MI with *mutual information* as the evaluation criterion can automatically evolve a

Pareto front of feature subsets that can reduce the number of features needed for classification and improve the classification performance over using all features. As multi-objective algorithms, NSGAIIMI and SPEA2MI achieved better performance than the single objective algorithm PSOMI.

6.4.3 Comparisons on CMDMI, NSGAIIMI and SPEA2MI

The performance of the multi-objective algorithms using the mutual information measure are compared with each other in this section. Since the performance of CMDMI is better than that of NSMI (shown in Section 6.4.1), only CMDMI is used here to compare with NSGAIIMI and SPEA2MI. Figure 6.5 shows the results of CMDMI, NSGAII and SPEA2MI.

According to Figure 6.5, it can be seen that on the four datasets with a relatively small number of features, the performance of the three algorithms are generally similar to each other in terms of both the average results shown by "-A" and the non-dominated results shown by "-B". This might be because using mutual information as the evaluation criterion the objective space is relatively easy, all the three algorithms can find the good results, but they also obtained other different results that are not plotted in the figures. Since they may achieve different combinations of individual features, the classification error rates evaluated by DT are different.

On the other four datasets with a relatively large number of features, NSGAIIMI achieved slightly better results than SPEA2MI in terms of both the number of features and the classification performance, especially on the Dermatology dataset. CMDMI achieved better results than NSGAIIMI and SPEA2MI on the Dermatology and Soybean Large datasets, but slightly worse results on the Chess and Connect4 datasets. However, on all these four datasets, CMD-B achieved slightly better classification performance than NSGAIIMI and SPEA2MI, which shows that CMDMI has a potential to evolve better solutions than NSGAIIMI and SPEA2MI.

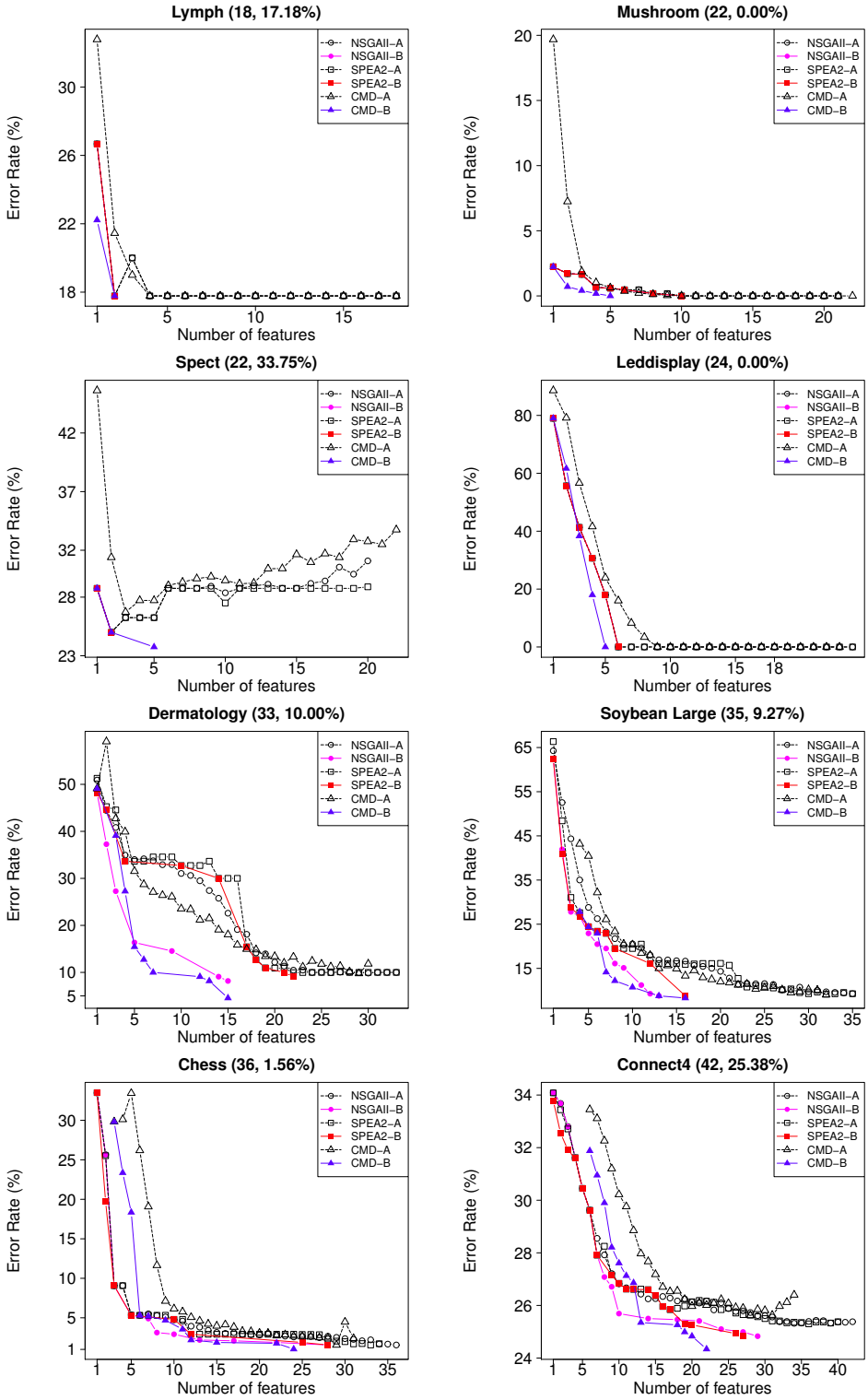

Figure 6.5: Comparisons Between CMDMI, NSGAIIMI and SPEA2MI.

6.4.4 Results of NSE and CMDE

Figures 6.6 compares the results of NSE, CMDE, and PSOE with $\alpha_e = 0.5$ and $\alpha_e = 0.9$, which employ the *entropy* measure to evaluate the relevancy of the selected features.

Results of NSE

According to Figure 6.6, in most cases, the average fronts of NSE (NSE-A) contained more than one solution that selected a smaller number of features and achieved better classification performance than using all features. In almost all datasets, NSE-B reduced the classification error rate by only selecting around half of the available features. Taking the Spect dataset as an example, NSE reduced the classification error rate from 33.75% to 25% by selecting only 9 features from the 22 available features.

The results suggest that the proposed NSE with the entropy measure can automatically evolve a set of feature subsets to simultaneously reduce the number of features and improve the classification performance over using all features.

Results of CMDE

According to Figure 6.6, on *all* datasets, the average front of CMDE (CMDE-A) evolved feature subsets that selected a smaller number of features (less than half in most cases) and achieved better classification performance than using all features. In most cases, CMDE-B maintained or even increased the classification performance by selecting less than 25% of the available features.

The results in Figure 6.6 suggest that as a multi-objective algorithm, CMDE can automatically evolve a Pareto front of feature subsets, which decrease the classification error rate and substantially reduce the number of features needed for classification.

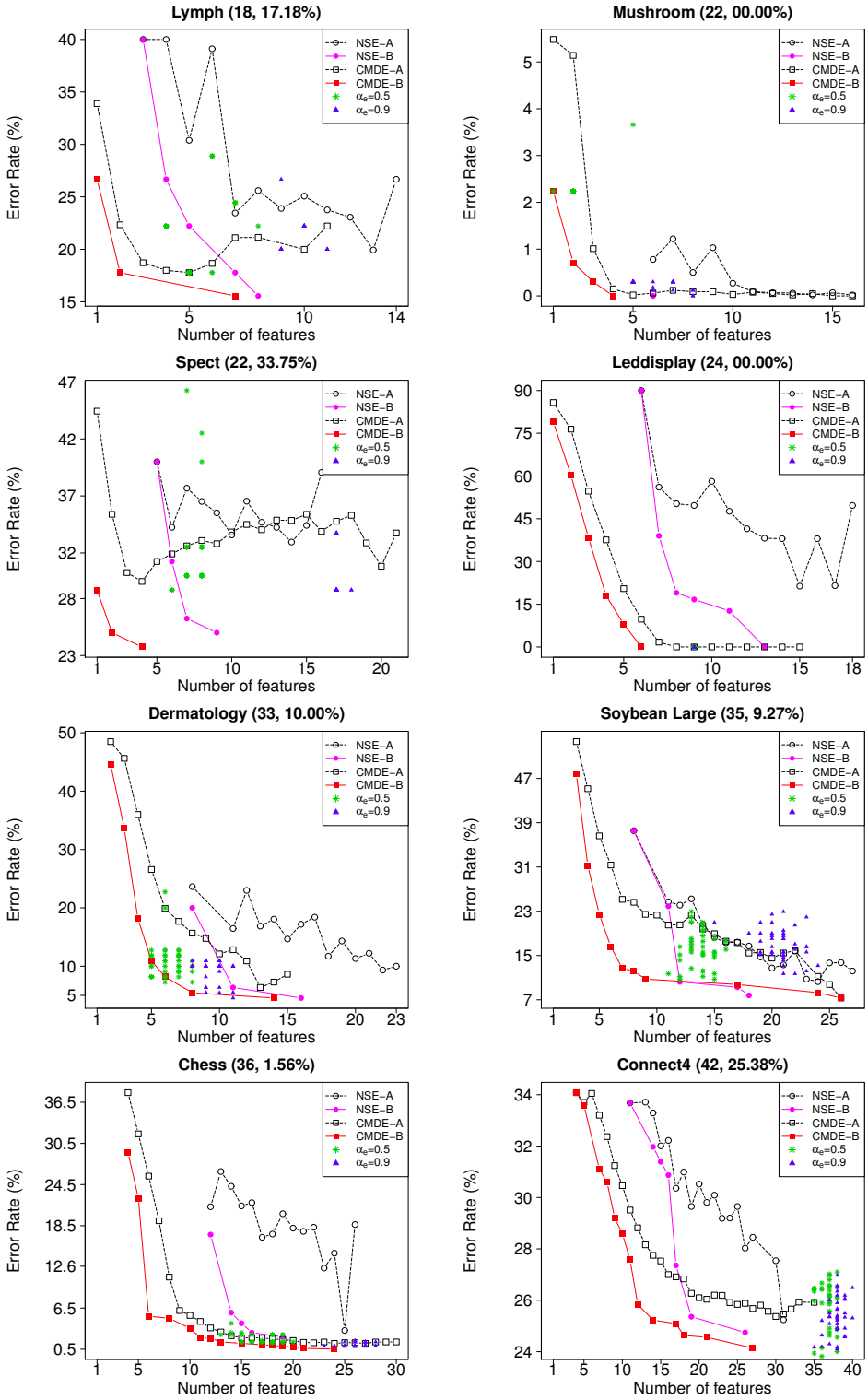

Figure 6.6: Experimental Results of PSOE, NSE and CMDE.

Comparisons Between PSOE, NSE, and CMDE

Comparing NSE with PSOE, in most cases, NSE (NSE-B) achieved better classification performance than PSOE with both $\alpha_e = 0.5$ and $\alpha_e = 0.9$ although the number of features is slightly larger in some cases. One can conclude that NSE outperformed PSOE when increasing the classification performance is considered more important than minimising the number of features.

Comparing CMDE with PSOE, in almost all datasets, CMDE evolved a smaller number of features and achieved better classification performance than PSOE with both $\alpha_e = 0.5$ and $\alpha_e = 0.9$. Only on the Connect4 dataset, CMDE achieved similar results to PSOE with $\alpha_e = 0.5$, but better results than PSOE with $\alpha_e = 0.9$.

CMDE outperformed NSE in terms of both the classification performance and the number of features. The main reasons are the same as discussed in Section 6.4.1. The comparisons show that with the *entropy* measure, the proposed multi-objective feature selection algorithms (NSE and CMDE) can better explore the solution space and achieve better feature subsets than the single objective feature selection algorithm (PSOE).

6.4.5 Results of NSGAIIE and SPEA2E

According to Figure 6.7, in seven of the eight datasets (the exception being the Soybean Large dataset), NSGAIIE-A contains one or more feature subsets that selected a smaller number of features and achieved similar or even better classification performance than using all features. In almost all cases, NSGAIIE-B achieved better classification performance by selecting around one third of the available features. Figure 6.7 shows that the performance of SPEA2E is similar to that of NSGAIIE in terms of both the classification error rate and the number of features in all datasets.

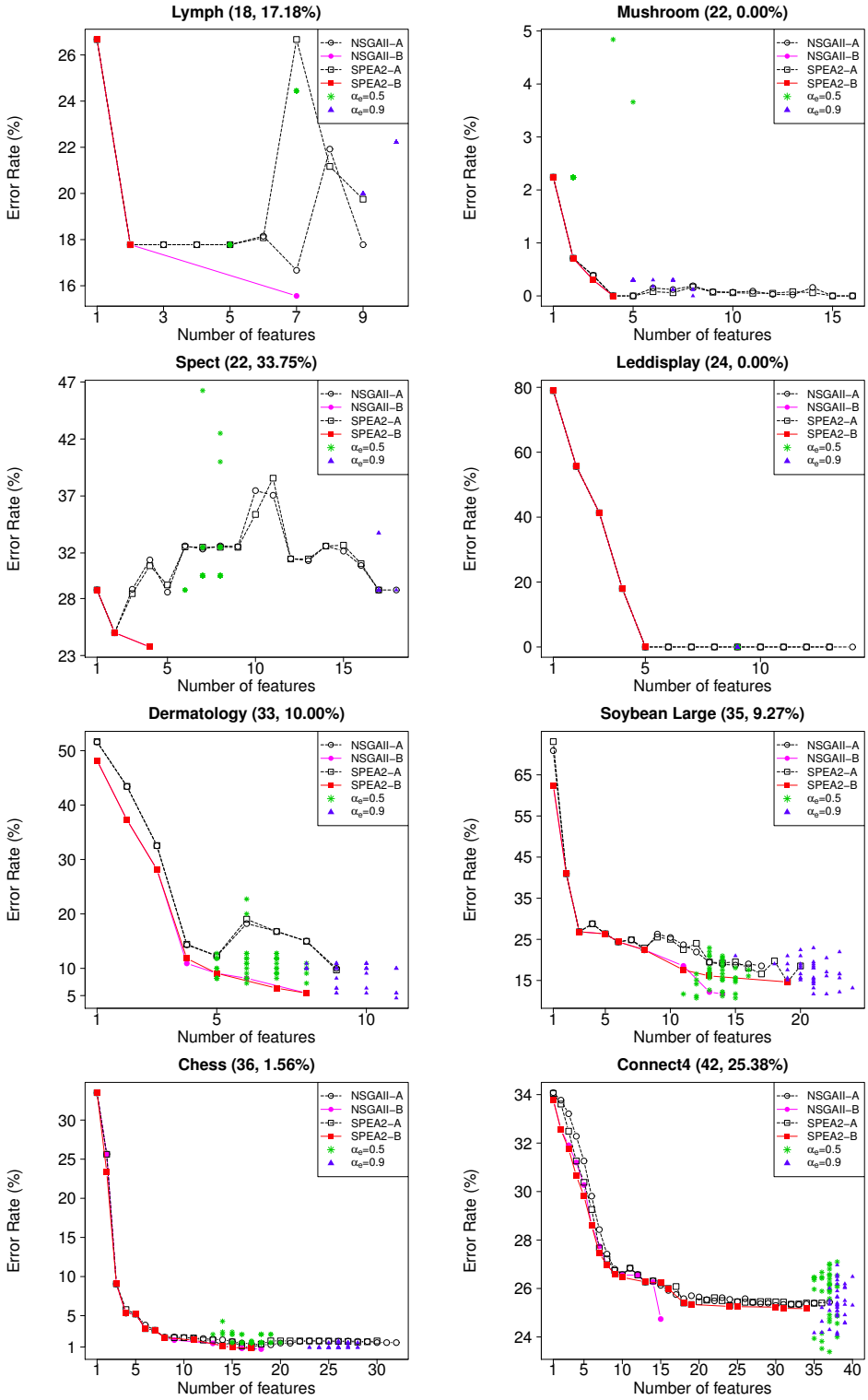

Figure 6.7: Experimental Results of PSOE, NSGAIIE and SPEA2E.

Comparisons Between PSOE, NSGAIIE, and SPEA2E

Comparing NSGAIIE and SPEA2E with PSOE, in many cases, the average fronts, NSGAIIE-A and SPEA2E-A outperformed PSOE with $\alpha_{mi} = 0.5$ in terms of the number of features and the classification performance. In most cases, NSGAIIE-A and SPEA2E-A achieved similar results with PSOE with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$, but NSGAIIE-B and SPEA2E-B outperformed PSOE.

The results in Figure 6.7 suggest that NSGAIIE and SPEA2E with the *entropy* measure can automatically evolve a Pareto front of feature subsets that can reduce the number of features needed for classification and improve the classification performance over using all features.

6.4.6 Comparisons on CMDE, NSGAIIE and SPEA2E

The performances of the multi-objective algorithms using the entropy measure are compared with each other in this section. Since the performance of CMDE is better than that of NSE (shown in Section 6.4.4), only CMDE is used here to compare with NSGAIIE and SPEA2E. Figure 6.8 shows the results of CMDE, NSGAIIE and SPEA2E.

According to Figure 6.8, it can be seen that the performance of the three algorithms are generally similar to each other in terms of both the average results shown by "-A" and the non-dominated results shown by "-B". In all cases, CMD-B slightly outperformed (or achieved the same performance as) NSGAIIE and SPEA2E in terms of the classification performance, which shows that CMDE has a potential to evolve better solutions than NSGAIIE and SPEA2E.

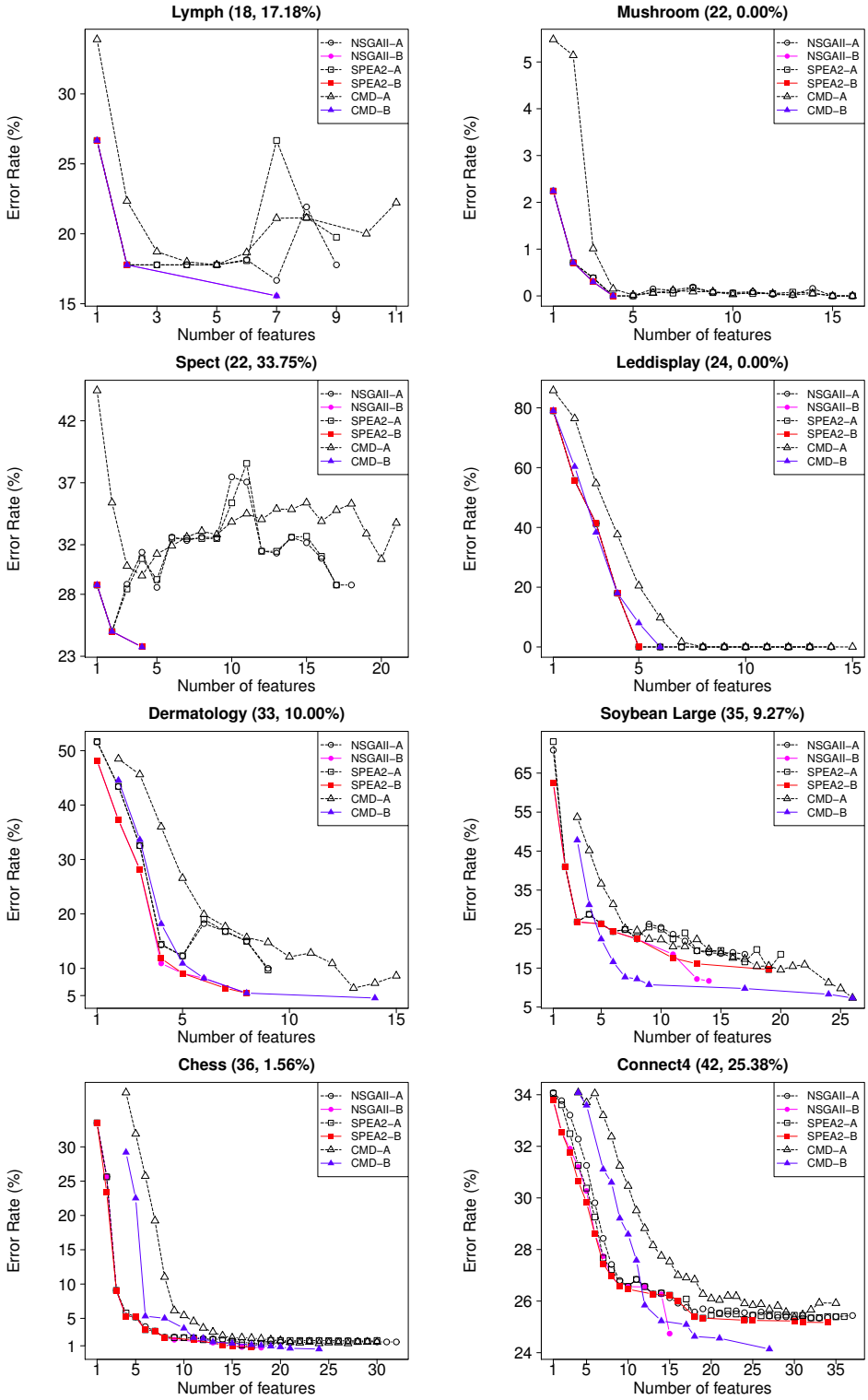

Figure 6.8: Comparisons Between CMDE, NSGAIIE and SPEA2E.

Table 6.1: Computational Time (In seconds).

Dataset	Lymph	Mushroom	Spect	Leddisplay	Dermatology	Soybeanlarge	Chess	Connect4
$\alpha_{mi} = 0.9$	0.13	0.67	0.27	0.23	0.24	0.38	0.57	102.57
$\alpha_{mi} = 0.5$	0.12	0.65	0.25	0.22	0.22	0.35	0.56	102.6
NSMI	0.43	0.95	0.57	0.51	0.56	0.7	0.86	102.89
CMDMI	0.13	0.65	0.25	0.21	0.22	0.34	0.55	102.57
NSGAIIMI	0.18	0.69	0.29	0.26	0.25	0.37	0.58	102.64
SPEA2MI	0.32	0.84	0.43	0.41	0.38	0.48	0.73	102.93
$\alpha_e = 0.9$	3.8	149.88	12.21	35.13	17.51	55.31	358.04	19243.5
$\alpha_e = 0.5$	3.19	121.51	8	34.31	15.42	48.66	256.17	18443.4
NSE	0.84	36.62	1.89	6.32	2.68	5.79	26.32	1395.02
CMDE	0.77	32.46	1.7	6.24	2.48	5.35	23.57	1463.54
NSGAIIE	0.93	36.02	2.06	4.36	1.71	5.53	17.78	1540.08
SPEA2E	0.84	27.41	1.73	5.63	2.14	4.36	22.79	1779.22

6.4.7 Comparisons Between Mutual Information and Entropy

Comparing the *mutual information* measure with the *entropy* measure, the figures show that NSE, CMDE, NSGAIIE and SPEA2E using the entropy measure usually achieved better classification performance than NSMI, CMDMI, NSGAIIMI and SPEA2MI using the mutual information measure. The main reason is that the entropy measure can discover the multiple-way relevancy and redundancy among a group of features to search for a subset of complementary features.

The number of features selected by entropy based algorithms is relatively large because the evaluation is based on a group of features (instead of a pair of features in mutual information based algorithms "-MI"). However, the number of features in the proposed multi-objective algorithms is always smaller than the single objective algorithms, which shows that they can explore the search space more effectively to minimise the number of features. The algorithms using the entropy measure can utilise their search ability and the discover multiple-way relevancy to reduce the number of features and simultaneously increase the classification performance.

6.4.8 Comparisons on Computational Time

Table 6.1 compares the average computational time used by the two single objective algorithms (PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$ and PSOE with $\alpha_e = 0.5$ and $\alpha_e = 0.9$), and the eight multi-objective algorithms for the evolutionary training process, where the time is expressed in seconds.

According to Table 6.1, it can be seen that on average, all the *mutual information* based multi-objective algorithms, NSMI, CMDMI, NSGAIIMI, and SPEA2MI, can finish the evolutionary training process within 1 seconds except for the Connect4 dataset. The multi-objective algorithms and the single objective algorithms (PSOMI with $\alpha_{mi} = 0.5$ and $\alpha_{mi} = 0.9$) have a similar computational cost. The Connect4 dataset usually used much longer time than other datasets because it has a much larger number of instances (44473) than other datasets.

According to Table 6.1, when using the *entropy* measure, all the multi-objective algorithms, NSE, CMDE, NSGAIIE, and SPEA2E, can finish the evolutionary training process within 40 seconds except for the Connect4 dataset. There is no much difference between the time used by the multi-objective algorithms. The single objective algorithm (PSOE with $\alpha_e = 0.5$ and $\alpha_e = 0.9$) used much longer time than the multi-objective algorithms, which is more than 10 times longer on the Chess dataset. The reason is that the number of features in the multi-objective algorithms is directly counted as one objective, which needs a much shorter time than the redundancy measure Red_e in the fitness function in PSOE. Clearly, the entropy based algorithms used a longer time than the mutual information based algorithms. The main reasons are the same as discussed in Chapter 5 on Page 151.

6.4.9 Selected Features

In order to show the stability of the multi-objective algorithms, it is necessary to analyse the features selected by each algorithm in different runs.

Table 6.2: Percentage of Appearance for the Chess Dataset.

Feature	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
NSMI	53	54	53	50	52	43	52	47	55	50	52	46	52	50	50	58	50	47	57	49	50	47	45	42	46	52	46	47	45	51	47	51	51	51	56	42
CMDMI	19	19	30	20	21	58	54	81	29	91	30	17	37	26	74	64	18	63	23	18	95	38	30	25	20	20	45	19	59	22	47	75	90	25	56	17
NSGAIIMI	14	10	30	14	15	52	52	80	31	93	33	11	36	32	67	59	16	59	22	14	100	39	37	22	18	19	43	18	54	24	47	73	87	26	50	14
SPEA2MI	7	1	39	3	5	69	66	89	36	96	42	1	49	32	82	76	5	79	19	1	100	52	46	22	12	14	56	2	72	26	59	86	92	29	62	1
NSE	52	48	52	43	55	50	53	52	58	52	48	46	50	42	54	50	51	53	55	53	49	51	49	49	48	55	51	55	49	52	46	52	49	54	50	51
CMDE	65	27	25	43	34	78	60	18	28	92	46	26	30	21	77	45	42	39	16	35	95	27	57	34	18	28	35	17	19	29	25	42	91	60	79	31
NSGAIIE	60	5	34	41	29	73	51	1	4	93	52	12	12	5	69	40	44	35	2	28	100	6	49	26	17	16	30	4	1	33	21	26	87	54	73	22
SPEA2E	68	12	31	48	27	79	61	0	10	94	53	6	7	4	75	49	55	32	0	29	99	10	58	24	14	8	35	0	0	39	23	22	88	65	78	13

In the 40 independent runs, each algorithm returns a number of feature subsets, which usually include different individual features. We take the Chess dataset as an example to analyse the selected features and the other datasets show a similar pattern. Table 6.2 shows the appearance percentage of each feature, where 100 means the corresponding feature appearances in all the solutions achieved over the 40 runs and 0 means the corresponding feature has never been selected by the algorithm.

Table 6.2 shows that the multi-objective algorithms (except for NSMI and NSE) are quite stable across different independent runs, where the most important feature is always selected. For the mutual information based algorithms, Features 21, 33, 10 and 8 are the most frequently selected features and have much higher percentages than other features, e.g. Feature 12. For the entropy based algorithms, Features 21, 10, 33 and 6 are the most frequently selected features. There are no such features in NSMI and NSE, which indicates that NSMI and NSE are not as stable as other algorithms. This is consistent with the results shown in the previous sections, where the performance of other multi-objective algorithms are better than that of NSMI and NSE.

6.4.10 Further Discussions

In the figures, the solutions used in the charts are the Pareto front solutions obtained using the filter evaluation criteria, but their classification

performances shown in the figures was evaluated by DT on the test sets.

As can be seen in the figures, some solutions in the *average* front (represented by "-A") dominate others although they are non-dominated solutions in the filter evaluation criterion space. This shows that the Pareto front in the filter evaluation criterion space does not necessarily involve the same subsets as the Pareto front in the DT-based evaluation space. Even the true Pareto front achieved by exhaustive search in the two filter evaluation criteria objective space may not correspond to the true Pareto front of using DT-based evaluation space.

The main reason is that the goodness of a feature subset evaluated by *mutual information* or *entropy* on the training set does not necessarily show its exact classification performance. Feature subsets with the same (better or worse) filter goodness do not necessarily achieve exactly the same (better or worse) classification performance evaluated by DT. For example, two feature subsets may have the same number of features, but different combinations of individual features. These two feature subsets may have the same goodness values evaluated by the filter evaluation criterion on the training set. So they are non-dominated to each other. However, when using DT (or any other learning/classification algorithm) to evaluate their classification performances on the unseen test set, their classification performances may be (slightly) different. The feature subset with better classification performance will dominate the other one. This is also the case for other filter criteria and other learning/ classification algorithms. Therefore, the Pareto front in the filter evaluation criterion space are usually not the same as the Pareto front in the DT-based evaluation space.

6.5 Chapter Summary

This chapter presents the first study on using PSO, NSGAII and SPEA2 for filter based multi-objective feature selection. Two multi-objective PSO (i.e. NSPSO and CMDBPSO), NSGAII and SPEA2 were investigated to pro-

pose four multi-objective feature selection frameworks. Based on the mutual information measure and the entropy measure, eight multi-objective filter feature selection algorithms were developed by using the two measures in each of the four frameworks.

This chapter shows that all the multi-objective algorithms successfully evolved a set of feature subsets with a smaller number of features and better classification performance than using all features, and outperformed the single objective algorithms, PSOMI and PSOE. The NSPSO based algorithms achieved slightly worse performance than other multi-objective algorithms. The main reason is the updating mechanism in NSPSO is less effective in terms of maintaining the diversity of the population and avoiding premature convergence, which are particularly important in feature selection tasks. The algorithms based on CMDPSO, NSGAII and SPEA2 achieved similar results, especially on the datasets with a relative small number of features. On the datasets with a relative large number of features, the CMDPSO based algorithms obtained the best classification performance, which shows that it has a potential to achieve better performance than other algorithms.

From Chapter 3 to Chapter 6, we have investigated four different types of feature selection algorithms, which are wrapper based single objective approaches, wrapper based multi-objective algorithms, filter based single objective algorithms, and filter based multi-objective algorithms. In the next chapter, we will further compare and investigate different types of feature selection algorithms to examine their advantages and disadvantages.

Chapter 7

Discussions

7.1 Introduction

Chapters 3-6 present four different types of feature selection algorithms, which are single objective wrapper algorithms, multi-objective wrapper algorithms, single objective filter algorithms, and multi-objective filter algorithms. In this chapter, we will further compare and discuss these four types of feature selection algorithms. The structure of this chapter is shown by Figure 7.1.

Wrapper and filter approaches are argued to have their own advantages and disadvantages. Wrappers can achieve better classification performance than filters, but filters are computationally less expensive and more general than wrappers. However, no thorough investigations have been made on how much difference there probably is between the two approaches in terms of the classification performance and the computational cost. Therefore, we will compare wrapper algorithms with filter algorithms in terms of the classification performance and the computational cost.

Wrapper approaches employ a learning/classification algorithm during the feature selection process. Therefore, they are argued to be less general than filter approaches, i.e. the features selected by a wrapper al-

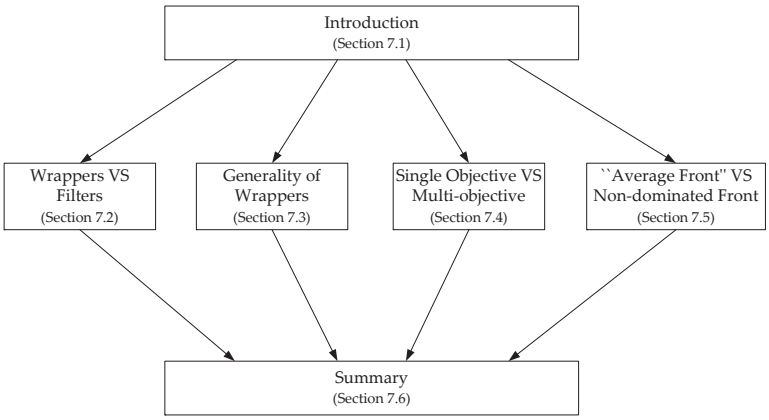

Figure 7.1: Overall Structure of Chapter 7.

gorithm can not achieve good performance when used with other classification algorithms. However, no investigation has been conducted to test this statement. In Section 7.3, we will investigate the generality of wrapper algorithms.

In Chapters 4 and 6, direct comparisons have been made between single objective algorithms and multi-objective algorithms. In Section 7.4, further discussions will be conducted to investigate the advantages of a multi-objective approach compared with a single objective approach. Meanwhile, the results of the multi-objective algorithms were presented in two ways, which are the "average front" and the non-dominated front. The difference between these two ways will be discussed in Section 7.5.

7.2 Wrappers VS Filters

Design of Experiments.

Four wrapper approaches and four filter approaches are used as examples to compare the computational time and the classification performance of wrappers and filters. The four wrapper algorithms are shown as "W-

SVM", "W-KNN", "W-DT", and "W-NB" and the four filter algorithms are shown as "F-MI", "F-E", "F-RS", and "F-PRS".

"W-SVM", "W-KNN", "W-DT", and "W-NB" use SVM, KNN, DT and NB to evaluate the classification performance (goodness) of the selected features during the evolutionary feature selection (training) process, respectively. The fitness function of the four wrapper algorithms contains the classification performance only (W-KNN is the same as PSOFS in Chapter 3). The settings of the four wrapper algorithms are the same as PSOFS described in Section 3.3 on Page 83. F-MI and F-E represent PSOMI and PSOE proposed in Chapter 5 without any weights in the fitness functions (Equations 5.3 and 5.6 on Page 137). The settings of F-MI and F-E are also the same as described on Page 142. F-RS and F-PRS are two PSO and rough set theory based filter feature selection algorithms, which were proposed in our recent paper [178]. F-RS is based on PSO and standard rough set theory and F-PRS is built on PSO and probabilistic rough set theory [179]. The fitness function aims to maximise the relevance measure, which represents the classification performance. The settings of F-RS and F-PRS are similar to that of F-MI and F-E and more detailed description can be seen in [178].

All the four filter algorithms only work for discrete/categorical datasets and the available discrete datasets have a relatively small number of features. Therefore, the Hillvalley and Madelon datasets, which are continuous datasets, were discretised in the experiments and used as the representative examples of datasets with a large number of features to test the classification performance and computational time of wrapper approaches and filter approaches. In total, 12 datasets chosen from UCI machine learning repository [25] were used in the experiments, where the details of the datasets can be seen in Table 1.1 on Page 16. All the algorithms are firstly run on the training set to obtain a good feature subset. Note that during the evolutionary feature selection (training) process, each evaluation in a wrapper approach involves a classification process that needs a training

set and a test set [5]. Therefore, for all the wrapper algorithms, the training set is further split into a sub-training set and a sub-test set. The reason for this is described in detail in [5]. Note that the results of W-SVM on the Madelon dataset is not available because it could not finish the running process within one week, which is shown as "N/A" in the figures.

7.2.1 Computational Time

Figure 7.2 shows the computational time of the four wrapper algorithms and the four filter algorithms. Each chart in the figure corresponds to one of the twelve datasets used in the experiment. The numbers in the bracket shows the number of features and the number of instances included in the dataset.

Computational Time of Wrappers

W-SVM. According to Figure 7.2, it can be seen that W-SVM used longer time than other methods in most cases. As a wrapper approach, each evaluation in W-SVM needs a training and testing classification process of a SVM to evaluate the goodness of the selected features. The SVM used here is the library (LIBSVM) developed by Chang and Lin [180], which involves a number of iterations and a cache method during the training of LIBSVM. The time complexity of LIBSVM is between $O(n * p^2)$ and $O(n * p^3)$, where n is the number of features and p is the number of instances. The computational time depends on how efficiently the cache method is used (dataset dependent) and the number of iterations. There is no theoretical analysis on the number of iterations needed in LIBSVM. Empirically, the number of iterations may be higher than linear to the number of training instances [180]. Therefore, LIBSVM may take very long time for large datasets. For the Madelon dataset with 500 features, W-SVM could not finish the evolutionary feature selection (training) process within one week. The possible reason is that on Madelon with a large number of features, the number of

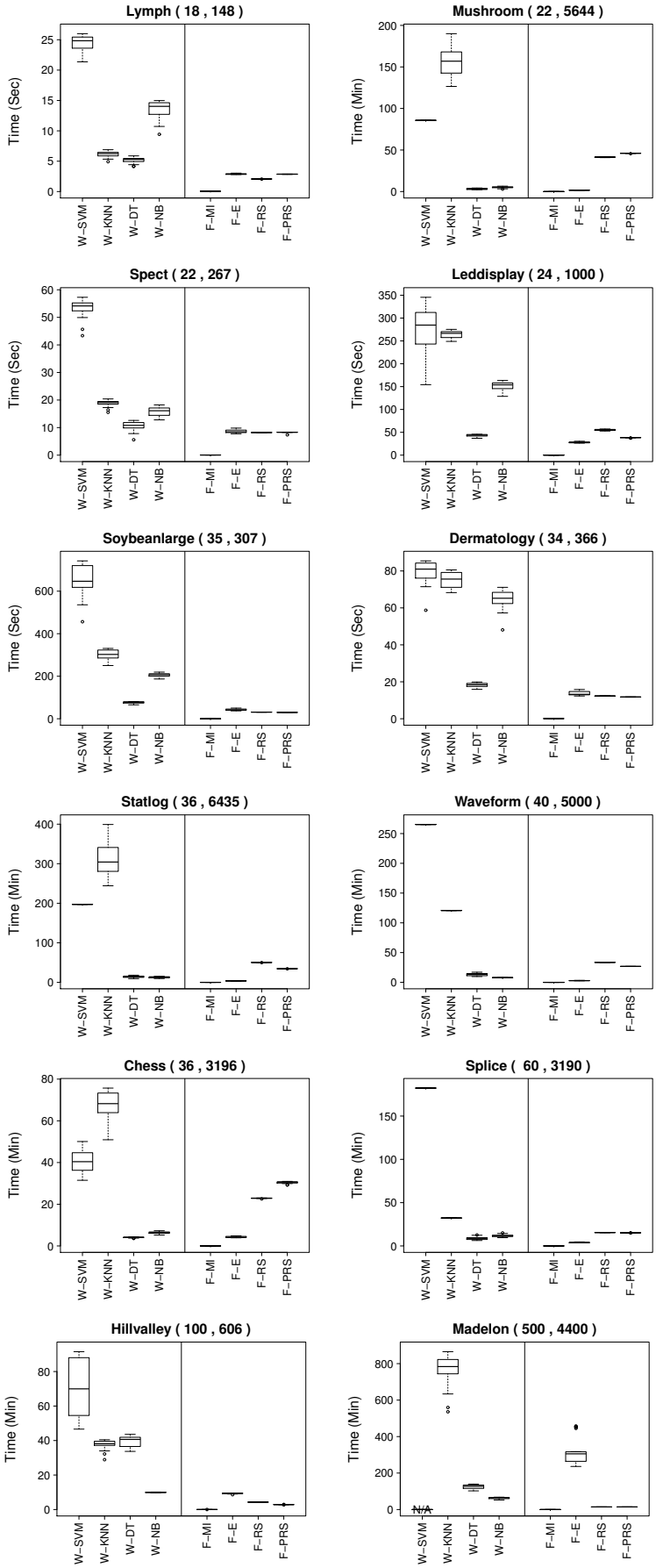

Figure 7.2: Comparisons on Computational Time.

iterations needed for LIBSVM is huge and the cache method was not able to be used efficiently. Another reason for LIBSVM using longer time than other wrapper algorithms is that W-SVM selected a larger number of features than the other algorithms. A large number of features need longer computational time for each evaluation/classification than a smaller number of features.

W-KNN. In most cases, the computational time used by W-KNN is shorter than W-SVM, but longer than other algorithms, especially on the datasets with a large number of instances. The main reason is that in the classification process of KNN, each instance in the sub-testing set is compared with all the instances in the sub-training set to determine its class label. The time complexity for each testing instance is around $O(n * p)$ [181]. If there are q instances in the sub-testing set, the time complexity of KNN is $O(n * p * q)$. Therefore, the increase of the number of instances causes a significant increase in the computational time of W-KNN.

From Figure 7.2, it can be observed that the computational time used by W-KNN is significantly influenced by the number of instances while that of W-SVM is significantly influenced by the number of features in the datasets. For example, the Mushroom and the Spect datasets have the same number of features, but different numbers of instances. W-KNN spent a shorter time than W-SVM on the Spect dataset with a smaller number of instances, but spent a longer time than W-SVM on the Mushroom dataset with a larger number of instances. By contrast, the Chess and Splice datasets have a similar number of instances, but different numbers of features. W-SVM spent a shorter time than W-KNN on the Chess dataset with a smaller number of features, but W-SVM spent a longer time than W-KNN on the Splice dataset with a larger number of features. On the Madelon dataset with the largest number of features, W-SVM even could not finish the feature selection process within one week.

W-DT and W-NB. According to Figure 7.2, the computational time of W-DT and W-NB is shorter than that of W-SVM and W-KNN in almost all cases. The main reason is that the time complexity of DT and NB, which are around $O(n^2 * p)$ [*actually for C4.5*] [182] in DT and $O(n * p)$ in NB, is less than that of KNN and SVM on these datasets. The number of features influences more in W-DT than in W-NB. Therefore, on the datasets with a large number of features, i.e. Hillvalley and Madelon, W-NB is faster than W-DT.

Computational Time of Filters

F-MI. As can be seen from Figure 7.2, the computational time used by F-MI is the shortest in all datasets. F-MI as a filter approach does not involve any classification process during the evolutionary feature selection process. The mutual information based fitness function of F-MI (shown by Equation 5.7 on Page 139) takes very short time to calculate. Meanwhile, the calculation of the possible mutual information between each feature and the class labels, and the possible mutual information between each pair of features, only needs to be performed once for each dataset before the evolutionary process. During the evolutionary feature selection process, the calculation of the fitness only needs to refer to these values. Therefore, F-MI is fast for all the datasets used in the experiments.

F-E. As can be seen from Figure 7.2, the computational time used by F-E is longer than F-MI in all cases and longer than F-RS and F-PRS on datasets with a large number of features. The main reason is that the fitness function (Equation 5.8 in Page 139) of F-E is more complex than the fitness function used in F-MI. The complexity of the fitness function in F-E increases rapidly along with the number of features selected. Therefore, on the datasets with a relatively small number of features, F-E is often faster than F-PR and F-PRS, but on the datasets with a large number of features, F-E is usually slower than F-RS and F-PRS.

F-RS and F-PRS. According to Figure 7.2, there is no much difference between the time used by F-RS and F-PRS. The main reason is that the calculation of the standard rough set based measure in F-RS is similar to that of the probabilistic rough set based measure in F-PRS. The only difference is that probabilistic rough set (F-PRS) determines equivalent classes based on a threshold while the standard rough set (F-RS) does not have any threshold. F-PRS used a shorter time than F-RS on some datasets because F-PRS selected a smaller number of features than F-RS. The calculation of the measures in both F-RS and F-PRS is significantly influenced by the number of instances. Therefore, on the datasets with a relatively small number of instances, F-RS and F-PRS is faster than F-E, but on the datasets with a large number of instances, F-RS and F-PRS is slower than F-E.

Comparisons Between Filters and Wrappers

According to Figure 7.2, it can be seen that wrapper algorithms using SVM and KNN spent longer time than all other algorithms. Wrappers using DT and NB used similar or even shorter time than filter algorithms in some cases, such as on the Chess dataset and the Madelon dataset. When the number of features increases, the computational time used by W-SVM, W-DT and F-E increased more than other algorithms. When the number of instances increases, the computational time used by W-KNN, F-RS and F-PRS increased more than other algorithms. Overall, F-MI (filter) is always the fastest algorithm regardless of the number of features and the number of instances. For wrapper algorithms, when the number of features is large, the fastest wrapper algorithm is W-NB. When the number of instances is small, one can choose the wrapper algorithm W-DT.

7.2.2 Classification Performance

Figure 7.3 shows the classification performance of the four wrapper algorithms and the four filter algorithms. In the wrapper approaches, the

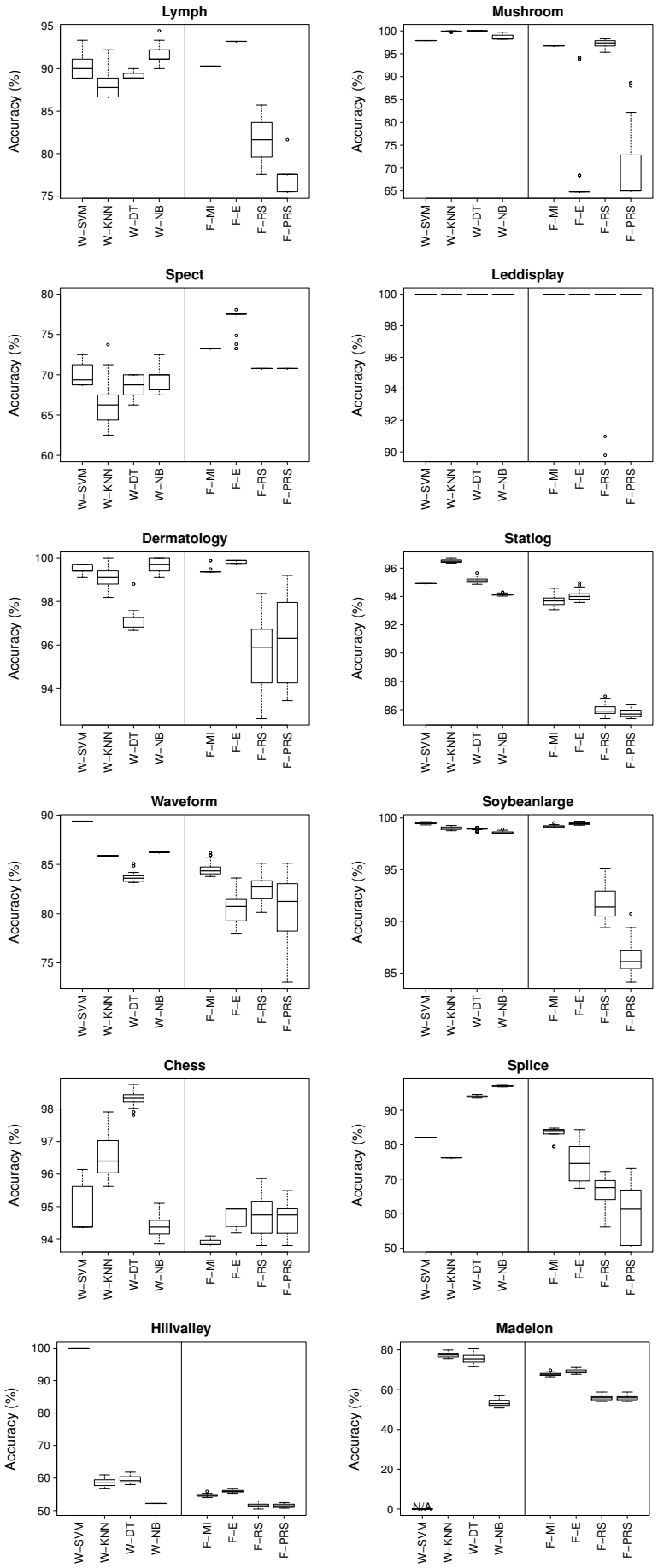

Figure 7.3: Comparisons on Classification Performance.

classification performance is evaluated by the internal classification algorithm used during the evolutionary feature selection process, e.g. the classification performance of the features selected by W-KNN is evaluated by KNN. All the classification algorithms can be used to evaluate the classification performance of the four filter algorithms. Filter approaches do not involve any classification process during the evolutionary feature selection process. Therefore, SVM with which W-SVM achieved better performance than others is used to test the classification performance to slightly bias the four filter algorithms. The performance of using other classification algorithms is often similar or slightly worse than that of SVM, which is not presented here.

As can be seen in Figure 7.3, in almost all cases, the best classification performance is achieved by one of the wrapper approaches. The worst classification performance is achieved by one of the filter approaches. For the four filter algorithms, the performance of F-MI and F-E are better than that of F-RS and F-PRS. F-E is slightly better than F-MI because the fitness function of F-E considers the selected features as a whole group rather than each pair of features in F-MI. All the four wrapper algorithms achieved almost the same performance on the Leddisplay dataset. W-SVM achieved better performance than the other three algorithms on 5 of the 11 datasets (except for Leddisplay). This number is 2 for W-KNN, 2 for W-DT, and 2 for W-NB. Clearly, different classification algorithms perform better on different datasets, depending on characteristics of the algorithm and the data itself. The choice of the best classification algorithm for a certain type of data is beyond the scope of this thesis, but the results here show that any of the four wrapper algorithms can be used for feature selection to obtain reasonable good classification performance.

Overall, considering both the classification performance and the computational cost, if users have enough time, wrapper approaches W-SVM and W-KNN are good choices. If the demand of users is more on the computational time than the classification performance, it is better to use a

filter algorithms, such as F-MI. Meanwhile, if users need to avoid poor classification performance, fast wrapper algorithms, such as W-DT and W-NB, are good choices.

7.2.3 Further Comparisons

This section further investigates the individual features selected by the eight different algorithms (i.e. W-SVM, W-KNN, W-DT, W-NB, F-MI, F-E, F-RS and F-PRS) to test their consistency. Figure 7.4 takes the Chess dataset as an example to show the number of appearance of each individual feature over the 40 independent runs in the eight algorithms, where each chart corresponds to one of the eight algorithms. In each chart, the vertical axis shows the frequency of the feature being selected, where 40 means that the feature was selected in all the 40 independent runs while 0 means the feature was never been selected. The horizontal axis shows the index of all the 36 features in the Chess dataset.

Figure 7.4 shows that the eight algorithms selected different numbers of features. For example, W-NB and F-MI usually selected a relatively small number of features while F-RS and F-PRS usually selected a relatively large number of features. This is mainly because they used different criteria to evaluate the quality of the feature subsets, where the best feature subset for one criterion may not be the best feature subset for another criterion. Meanwhile. Figure 7.4 also shows that all the eight algorithms *consistently* selected Features 10, 21 and 33 on all their 40 independent runs. These three features may be the "core" features of the Chess dataset and all the eight algorithms selected them. The detailed results also reveal that they selected other different individual features. This is due mainly to the feature interaction problem, where some other different (complementary) features are needed to work together with such "core" features to optimise their fitness functions, which are based on different criteria.

The other datasets show a similar pattern to the Chess dataset, i.e.

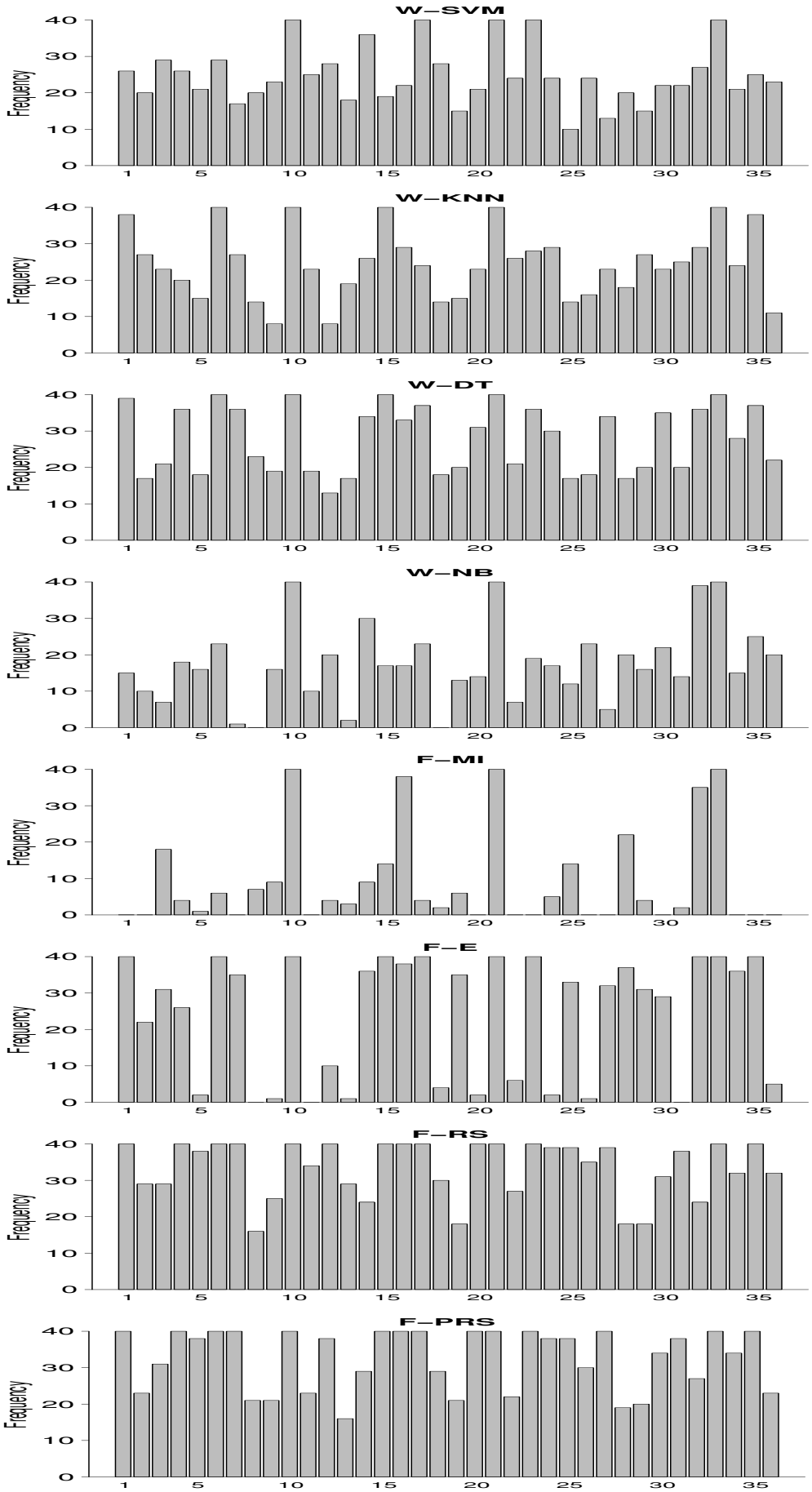

Figure 7.4: Comparisons on Classification Performance.

there are always some "core" features selected by all the eight algorithms and also some different complementary features selected by different algorithms. The further investigation of "core" features and feature interaction in different datasets may require domain knowledge and we will work on it in future.

7.3 Generality

This section tests the generality of four wrapper algorithms, i.e. W-SVM, W-KNN, W-DT, and W-NB. The classification performance of the features selected by a wrapper algorithm is tested using all the four classification algorithms, i.e. SVM, KNN, DT and NB. The datasets that were used in Chapters 3 and 4 (i.e. wrapper approaches) are used in the experiments in this section.

The experimental results are shown in Table 7.1. In order to make the results easy to observe, the detailed classification performance are not presented. Only the results of the statistical significance tests between the classification performance of the *selected* features and that of *all* features are shown in the table. The pairwise Student's T-test [171] with a significance level of 0.05 (or confidence interval of 95%) was performed here. In the table, "+" (or "-") means the classification performance of the selected feature subsets is significantly better (or worse) than using all features. "=" means they are similar to or not significantly different from each other. The last row summaries the total number of datasets where the classification performance of a classification algorithm is similar or significantly better than using all features. In a few cases, the classification algorithm could not obtain any results because of either the code threw a no pointer error or the dataset is too big and the evolutionary feature selection process could not finish within two days, which happens more when using SVM. These cases are indicated by empty cells in the table.

According to Table 7.1, when using KNN during the evolutionary fea-

ture selection process, KNN itself using the selected feature subsets increased or maintained the classification performance on 13 out of the 14 datasets. On 7 of the 14 datasets, the classification performance of NB, DT and SVM were maintained or improved overing using all features. The results show that W-KNN benefits KNN itself more, but it also benefits other classifiers. The main reason is that KNN is very simple, which has a small probability to overfit the problem.

When using NB during the feature selection process, NB itself using the selected feature subsets increased or maintained the classification performance on 8 of the 13 datasets (no results obtained for the Isolet5 dataset). The classification performance of KNN, DT and SVM were maintained or increased overing using all features on 8, 9 and 6 out of the 13 cases, respectively. The results show that W-NB benefits NB itself more than others. Although the total number of "+" and "=" for DT is 9 and larger than that of NB (8), 7 of these 9 cases is "=", which means the classification performance of DT using the feature selected by W-NB is similar to that of all features. The possible reason is that NB assumes the features are conditionally independent to each other and it is easy to select a group of individually good features but not complementary features. Such features usually contains most of the useful information of the original features, but may also have redundancy.

When using DT during the feature selection process, as can be seen from Table 7.1, the classification performance of DT, KNN, NB and SVM were the similar or increased over using all features on 6, 7, 7 and 6 of the 13 datasets, respectively. When using SVM during the feature selection process, the classification performance of SVM, KNN, NB and DT were the same or increased over using all features on 4, 7, 5 and 4 of the 9 datasets with available results, respectively.

Overall, the results show that wrapper approaches can be reasonably general. Wrappers using a relatively simple classification algorithms, e.g. KNN and NB, can be general to different classification algorithms. Wrap-

Table 7.1: Classification Performance Compared with All Features

	KNN in Training				NB in Training			
	KNN	NB	DT	SVM	NB	KNN	DT	SVM
Wine	+	-	+	+	+	+	=	+
Australian	+	+	-	+	+	+	+	+
Zoo	+	-	+	+	-	+	=	+
Vehicle	+	-	=	-	-	=	=	-
German	=	=	-	-	+	=	-	-
WBCD	+	-	=	+	+	-	+	+
Ionosphere	+	+	-	-	+	+	=	-
Lung	+	-	-	=	-	+	=	+
Sonar	+	-	=	-	-	-	-	-
Movementlibras	-	-	=	=	-	-	=	-
Hillvalley	+	=	-	-	=	=	-	-
Musk1	+	+	=	-	+	-	=	-
Madelon	+	=	-	+	+	-	-	+
Isolet5	+	=	-	-				
Total NO. of "+"	12	3	2	5	7	5	2	6
Total NO. of "+" and "="	13	7	7	7	8	8	9	6

	DT in Training				SVM in Training			
	DT	KNN	NB	SVM	SVM	KNN	NB	DT
Wine	+	+	-	+				
Australian	+	+	+	+				
Zoo					+	+	-	=
Vehicle	+	-	-	-	-	-	-	-
German	-	=	=	-	-	+	+	-
WBCD	=	-	-	+				
Ionosphere	-	+	+	-	=	+	=	-
Lung	=	+	-	+	=	+	-	=
Sonar	-	=	-	-	-	=	-	+
Movementlibras	-	-	-	=	-	-	=	=
Hillvalley	-	=	=	-	=	+	=	-
Musk1	+	-	+	-				
Madelon	-	-	=	+				
Isolet5	-	-	=	-	-	=	=	-
Total NO. of "+"	4	4	3	5	1	5	1	1
Total NO. of "+" and "="	6	7	7	6	4	7	5	4

pers using a relatively complicated classification algorithm, e.g. DT and SVM, are less general than using KNN and NB. The possible reason is that the complicated classification process of DT and SVM may select features that are particularly suit themselves. The classification performance

of SVM can be increased by features selected resulted from W-KNN and W-NB, which is faster and even better classification performance than W-SVM. Meanwhile, SVM as a complicated algorithm using either all features or the selected features usually obtains high classification performance and the best classification performance is often achieved by SVM. Therefore, if users want to reduce the number of features but still achieve high classification performance, it is good to use W-KNN and W-NB to select features and use SVM for classification on the unseen test set.

7.4 Single Objective VS Multi-Objective

Chapters 4 and 6 have presented the direct comparisons between single objective algorithms and multi-objective algorithms. The results show that the multi-objective approaches can discover multiple and better solutions than the single objective algorithms. In this section, we further compare the evolutionary feature selection process of single objective and multi-objective algorithms. The best wrapper based single objective algorithm PSOIniPG developed in Chapter 3 and the best multi-objective algorithm CMDPSOFS developed in Chapter 4 are used as the representative methods for analysis.

Table 7.2: Computational Time (In minutes)

Method	Wine	Australian	Zoo	Vehicle	German	WBCD	Ionosp	Lung	Hillvalley	Musk1	Madelon	Isolet5
PSOIniPG	0.21	2.57	0.07	6.02	9.37	2.07	0.71	0.02	14.6	7.22	651.88	247.12
CMDPSOFS	0.25	4	0.09	6.59	9.3	2.71	1.54	0.01	23.49	6.02	394.45	200.71

7.4.1 Computational Time.

Table 7.2 shows the computational time used by PSOIniPG and CMDPSOFS. As wrapper approaches, most of the computational time is spent on the evaluation of the selected feature subsets, which involves a classification process and depends mainly on the number of features in a certain

dataset. From Table 7.2, it can be observed that on the datasets with a relatively small number of features, both PSOIniPG and CMDPSOFS finished the evolutionary training (feature selection) process within 25 minutes. CMDPSOFS often used slightly longer time than PSOIniPG on such datasets. The main reason is that there is no significant difference between the evaluation time (depending on the number of selected features) used by PSOIniPG and CMDPSOFS. Meanwhile, CMDPSOFS involves additional procedures related to the multi-objective mechanism, which takes a slightly longer time than PSOIniPG. However, on the large datasets, i.e. Madelon and Isolet5, CMDPSOFS used a much shorter time than PSOIniPG. The main reason is that CMDPSOFS selected a much smaller number of features than PSOIniPG on these datasets, which needs a much shorter time for each evaluation during the evolutionary feature selection process.

7.4.2 Evolutionary Process.

The number of features selected by PSOIniPG is larger than CMDPSOFS due mainly to its single objective updating mechanism. Although PSOIniPG considers both the classification performance and the number of features during the evolutionary process, the classification performance is treated as the priority. When available, PSOIniPG will search toward the area of the solution space with a low classification error rate regardless of the number of features.

Figure 7.5 shows the change of the classification error rate and the number of features selected by $gbest$ during a single evolutionary process of PSOIniPG on the Madelon dataset. The horizontal axis show the number of iterations from 1 to 100. Since the total number of features is 500, which is much larger than the error rate in [0, 100], the numbers of features are divided by 5 to scale the range to [0, 100] and shown by the vertical axis.

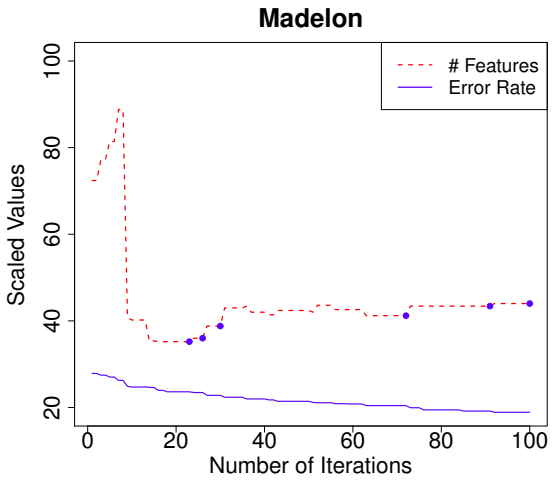

Figure 7.5: The number of features and the classification error rate of $gbest$ in PSOIniPG during the evolutionary feature selection process.

According to Figure 7.5, it can be seen that the error rate of $gbest$ always became smaller and smaller since the classification performance is the fitness function (the priority). The number of features fluctuated because $gbest$ was updated whenever the error rate became smaller, but the number of features might be larger. The blue points in the red line shows the solutions that are non-dominated to each other and dominate all other solutions of $gbest$ during the evolutionary process. Since the single objective mechanism only keeps one single solution, only the solution with the lowest classification error rate was returned by PSOIniPG. Note that PSOIniPG selected smaller feature subsets than other single objective algorithms (see Chapter 3 on Page 99) because PSOIniPG considers the number of features in the $pbest$ and $gbest$ updating procedure. For other single objective algorithms, there are more (non-dominated) solutions like the blue points found during the evolutionary process, but none of them were kept and reported by the algorithm.

Figure 7.6 shows the non-dominated solutions found by CMDPSOFS and all the non-dominated solutions found by PSOIniPG (i.e. the blue points in Figure 7.5). It can be seen that the solutions of CMDPSOFS have

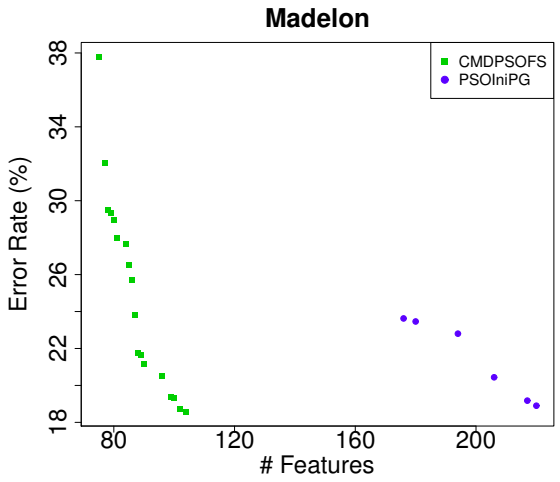

Figure 7.6: The solutions obtained by CMDPSOFS and PSOIniPG.

a smaller number of features and a lower classification error rate than that of PSOIniPG. This is due mainly to the multi-objective mechanism in CMDPSOFS, where the non-dominated solutions obtained during the evolutionary process are kept as potential leaders ($gbest$) to guide the algorithm to search around to find better solutions with smaller numbers of features and higher classification performance. Meanwhile, CMDPSOFS returns multiple solutions, which provide more choices than PSOIniPG.

7.5 "Average Front" VS Non-dominated Front.

In Chapters 4 and 6, the results of the multi-objective algorithms were presented in two ways, which are the "average front" and the non-dominated front, see the results in Figure 4.2 on Page 120 and Figure 6.3 on Page 173. As stated previously, the results from the 40 sets of feature subsets achieved by one multi-objective algorithm from the 40 independent runs are firstly combined into one union set. The classification error rates of the feature subsets with the same number of features (e.g. m) are averaged and presented as the "average front". The non-dominated solutions in the union set are presented as the non-dominated front.

Both the "average front" and the non-dominated front can show the performance of a multi-objective algorithm, but the non-dominated front is a more appropriate way to present the results in feature selection tasks. There are two main reasons. The first reason is that a solution in the "average front" is not necessarily a complete/meaningful solution for a feature selection task. Each average solution is formed by the number m and the average classification error rate of all feature subsets of size m in the union set. However, feature selection problems do not only involve the number of features and the classification performance, but also involve the selected individual features. There can be many feature subsets with m features, but with different combinations of m features. So strictly speaking, the combinations of different individual features can not be averaged. Therefore, the solutions in the "average front" is not a complete solution and should not send to users. The second reason is that the non-dominated front involves a simple further selection process, which provides a better set of non-dominated solutions to users. By selecting only the non-dominated solutions from the union set, the non-dominated front usually has a small number of solutions and the solutions usually have a smaller number of features than the "average front" solutions. It therefore provides fewer but better solutions to the users and reduces their cost for selecting a single solution. Meanwhile, each solution in the non-dominated front is a complete solution of a feature selection problem. Multiple solutions with the same number of features and the same classification performance are presented at the same point in the figures, but all of them are complete/meaningful solutions. Therefore, for a certain feature number m, the non-dominated front could provide different combinations of individual features to users.

Accordingly, the non-dominated front is more appropriate than the "average front" to show the performance of a multi-objective feature selection algorithm.

7.6 Summary

This chapter investigated four issues in feature selection. The first one is the comparisons on the classification performance and computational time of wrapper approaches and filter approaches. The second one is the generality of wrapper approaches. The third one is the discussions on the advantages of multi-objective feature selection algorithms over single objective algorithms. The fourth one is the difference between the "average" front and the non-dominated front.

Wrapper approaches are argued to be computationally more expensive and can achieve better classification performance than filter approaches. This chapter shows that when using NB or DT in a wrapper algorithm, it can be computationally cheaper than a filter algorithm with a complex measure such as rough set based measures. Meanwhile, because of the interaction between features and a certain classification algorithm, wrapper algorithms are often reasonable good in terms of the classification performance, although not always better than filter approaches.

Wrapper approaches are also argued to be lack of generality. This chapter shows that wrappers using a simple classification algorithm, e.g. KNN and NB, during the feature selection process are often general to other classification algorithms. Wrappers using a relatively complicated classification algorithm, e.g. SVM, are usually not general to other algorithms because SVM involves a complicated classification process and the features selected are more specifically suitable to itself rather than other classification algorithms.

This chapter also shows that the multi-objective mechanism is a more appropriate way than the single objective mechanism for feature selection tasks. Single objective algorithms only keep one single solution $gbest$ to guide the search, which is more likely to become stuck in a local optima. Multi-objective algorithms keep the non-dominated solutions found during the evolutionary process, which are used as potential leaders to guide

the algorithm to search around and find better solutions.

Finally, this chapter discusses the differences between the "average" front and the non-dominated front in multi-objective algorithms. It shows that the non-dominated front is a better way to present the results of a multi-objective feature selection approach because it has a further selection process and all solutions it provides are complete/meaningful solutions.

Overall, this chapter shows that wrapper algorithms can be faster than (some) filter approaches and general to different classification algorithms. If users have a high demand on the computational time and also need to avoid poor classification performance, a filter algorithm (like F-MI) and a fast wrapper algorithm (W-DT or W-NB) can be good choices. If users have a high demand on the classification performance, a fast classification algorithm (i.e. NB) can be used in a wrapper to select features, and the selected features can be used with SVM for classification. Meanwhile, a multi-objective algorithm is a better choice than a single objective algorithm and it is better to examine the performance of a multi-objective algorithm using the non-dominated front than the "average" front.

Chapter 8

Conclusions

This thesis focuses on PSO for feature selection in classification problems. The overall goal was to investigate and improve the capability of PSO for feature selection by developing a new PSO based approach to feature selection for reducing the number of features and achieving similar or even better classification performance than using all the original features. This goal was successfully achieved by developing a number of new methods using PSO to automatically evolve one or more feature subsets with a small number of features and maintain or even increase the classification performance. The proposed methods were examined and compared with existing methods on a range of classification problems of varying difficulty. The results show a clear pattern that PSO can be effectively used for feature selection and dimensionality reduction in classification.

The rest of this chapter provides conclusions for each of the research objectives of this thesis and gives main findings and highlights from each individual chapter, and then presents potential research areas for future work.

8.1 Achieved Objectives

This thesis has achieved the following research objectives:

- Proposes new initialisation and new updating mechanisms in PSO for single objective wrapper feature selection. The proposed initialisation method simulates both the traditional forward and backward feature selection methods to utilise their advantages and avoid their disadvantages. The proposed $pbest$ and $gbest$ updating mechanism considers the number of features, which avoids the limitation of the original updating mechanism, i.e. missing the small feature subset with high classification performance. By combining the new initialisation and updating mechanisms, the proposed PSO based algorithm can significantly reduce the number of features, improve the classification performance over using all features, and reduce the computational time. It also outperforms two traditional methods, a standard PSO based method and a new PSO based two-stage feature selection method.

- Proposes a new PSO based approach to wrapper multi-objective feature selection. Different from existing PSO based feature selection algorithms, the proposed multi-objective approach treats the two main objectives separately during the evolutionary process, which aims to maximise the classification performance and simultaneously minimise the number of features. By considering the trade-off between the two main objectives, the proposed approach can successfully find a set of non-dominated solutions, which have a smaller number of features and achieve better classification performance than using all features. The proposed multi-objective approach can obtain more and better feature subsets than single objective algorithms.

- Introduces information theory based measures to PSO for feature selection to propose a filter based single objective approach. Two new PSO based algorithms were developed, where the first one measures the relevance and redundancy between a pair of features while the second algorithm considers all the selected features as a group to

evaluate their relevance and redundancy. Both two proposed algorithms can reduce the number of features and maintain or even increase the classification performance in most cases, and outperform two traditional filter methods. The second algorithm achieves better classification performance than the first algorithm, but selects a larger number of features and uses longer computational time. The feature subsets selected by both two methods are general to three different classification algorithms.

- Proposes a multi-objective filter feature selection approach using PSO and information theory based measures. The proposed multi-objective approach aims to minimise the number of features and the maximise the relevance between the selected features and the class labels. The proposed approach can successfully evolve a set of non-dominated feature subsets with a smaller number of features and better classification performance than using all features, and outperform the above PSO based single objective filter algorithms.

- Investigates the difference between wrapper and filter approaches in terms of the classification performance and the computational time, and also examines the generality of different wrappers. It is found that wrapper approaches generally achieve better or similar (but not worse) classification performance than filters, but wrapper approaches do not necessarily always need longer computational time than filter approaches. Wrapper approaches were claimed not general to different classification algorithms, but this thesis finds that wrappers built with a simple classification algorithm can be general to other classification algorithms.

8.2 Main Conclusions

This thesis finds that PSO can effectively address feature selection problems in filter or wrapper, single objective or multi-objective ways.

This section discusses the main conclusions for the five research objectives drawn from the five contribution chapters (Chapter 3 to Chapter 7).

8.2.1 Initialisation and Updating Mechanisms in PSO for Feature Selection

Chapter 3 proposes a new PSO based single objective wrapper feature selection algorithm based on a new initialisation strategy and a new updating mechanism.

Initialisation Mechanisms

It is found that the initialisation mechanism can significantly influence the performance of a PSO based algorithm to reduce the number of features and the computational time (Chapter 3).

Initialisation determines where the algorithm starts the search process. A better starting point can help the algorithm find better solutions and converge faster. This is particularly important in PSO based feature selection algorithms due to two main reasons. The first reason is that the evolutionary process of a PSO algorithm is led by the personal best ($pbest$) and global best ($gbest$). A better starting point means that a better leader appears earlier, which can then benefit the whole evolutionary process. The second reason is that one of the two main objectives in feature selection can be easily controlled, i.e. the number of features, which provides a chance to specify the initial positions of particles to have a small number of features. Meanwhile, a small number of features uses less computational time than a large number of features.

Updating Mechanism

It is found that the $pbest$ and $gbest$ updating mechanism can significantly influence the performance of PSO for feature selection.

$pbest$ and $gbest$ updating is one of the most important components in a PSO algorithm. The standard updating mechanism is not suitable for feature selection problems because it may select a feature subset with high classification performance but a large number of features. By considering the size of $pbest$ and $gbest$, the number of features can be significantly reduced while the classification performance is maintained or even increased due to the removal of the redundant features.

By combining the initialisation strategy and $pbest$ and $gbest$ updating mechanism, the PSO based algorithm simultaneously improves the classification performance and reduces the number of features, especially on datasets with a large number of features. By reducing the number of features, the computational time can also be reduced.

8.2.2 Multi-objective Wrapper Feature Selection

The *first* PSO based multi-objective feature selection approach is proposed in this thesis (Chapter 4). From Chapter 4, it is found that PSO can be successfully used for multi-objective feature selection to select a set of non-dominated feature subsets, which have a smaller number of features and achieve better classification performance than using all features. Examining the Pareto front achieved by the multi-objective algorithms can assist users in choosing their preferred solutions to meet their own requirements.

Feature selection is a complex problem with a large search space and many local optima. This thesis finds that a good PSO based multi-objective feature selection approach needs two important factors, which are a good method to select a good $gbest$ for each particle and an effective strategy to maintain the swarm diversity.

Selection of $gbest$

This thesis finds that the selection of $gbest$ in a PSO based multi-objective feature selection approach can significantly influence its performance. A good $gbest$ selection method can filter out crowded (similar) non-dominated solutions, i.e. potential $gbest$. It will then select good leaders for particles, which leads the algorithm to better explore the search space to obtain a better set of feature subsets. CMDPSOFS achieved better performance due partly to the use of a crowding factor together with a binary tournament selection to select $gbest$, which can effectively select and filter out some crowded non-dominated solutions as potential leaders.

Swarm Diversity

This thesis finds that a PSO based multi-objective feature selection approach needs a good strategy to maintain the diversity of the swarm. The search space of a feature selection problem usually has many local optima. This requires a high diversity in the swarm to avoid being stuck into local optima. CMDPSOFS employs different mutation operators in different groups of particles to keep the diversity of the swarm. It contributes the superior performance of CMDPSOFS, especially on the datasets with a large number of features, where the search space is larger and more complex than datasets with a smaller number of features. By contrast, NSPSOFS, which has the potential limitation of losing the diversity of the swarm quickly, can not achieve as good performance as CMDPSOFS.

8.2.3 PSO and Information Theory for Feature Selection

This thesis proposes the *first* PSO and information theory based feature selection approach (Chapter 5). It is found that information theory based measures can be successfully used with PSO to select a small number of features and maintain the classification performance. In filter approaches, the classification performance of the selected features is reflected by its

relevance to the class labels and the number of features is measured by the redundancy amongst the selected features. Two different relevance and redundancy measures were developed (Chapter 5), which are a pair-wise measure based on mutual information and a group based measure using the concept of entropy.

Pair-wise Measure

It is found that PSOMI using the pair-wise measure is faster and selects a smaller number of features, but the classification performance is not as good as the group based measure (PSOE). The main reason is that the pair-wise measure does not involve complex relevance or redundancy calculation and the optimal fitness value usually includes a small number of features. It only considers the relationship between two features and can not deal with complex interactions amongst a group of features, which is a challenge in feature selection.

Group Based Measure

It is found that PSOE using the group based measure can achieve better classification performance, but it is slower and selects more features than PSOMI. The main reason is that the group measure involves a much more complicated calculation than the pair-wise measure. Since it considers the selected features as a whole, it can better deal with feature interaction problems and accordingly achieve better classification performance.

8.2.4 Multi-objective Filter Feature Selection

This thesis proposes the *first* PSO based multi-objective filter feature selection approach (Chapter 6). It also proposes the *first* NSGAII and SPEA2 based multi-objective filter approaches to feature selection.

Chapter 6 further confirms the findings for multi-objective feature selection approaches concluded from Chapter 4, i.e. the diversity of the pop-

ulation/swarm and the selection of $gbest$ can significantly influence the performance of a multi-objective feature selection algorithm.

Filter Measures and Classification Performance

It can be found that the goodness of a feature subset evaluated by a filter evaluation criterion (e.g. mutual information) does not necessarily show its exact classification performance. Feature subsets with the same (better or worse) filter goodness do not necessarily achieve exactly the same (better or worse) classification performance. Therefore, the Pareto fronts in the filter evaluation criterion space are usually not the same as the true Pareto front in the classification accuracy space. This is not only for the proposed information based measures, but also for many other filter evaluation criteria. A good filter evaluation should reflect the classification performance as close as possible.

8.2.5 Wrappers VS Filter

This thesis investigates the differences between wrapper and filter approaches in terms of the classification performance and the computational time.

Classification performance

This thesis finds that in general wrappers achieve better classification performance than filters. For a certain dataset, the best classification performance is always achieved by a wrapper method, but a wrapper method does not necessarily always achieve better classification performance than all filter approaches.

Computational Time

This thesis finds that wrapper approaches do not necessarily always need longer computational time than filter approaches. Most of the computa-

tional time of a wrapper algorithm is spent on the learning/classification process in the evaluation procedure. If a complicated classification algorithm (e.g. SVM) is applied, the wrapper algorithm usually needs longer computational time. However, if a computationally cheap classification algorithm (e.g. NB) is applied, the wrapper algorithm may be faster than a complex filter algorithm, especially on large datasets.

8.2.6 Generality of Wrappers

This thesis finds that wrappers can be general to different classification algorithms, which is different from the existing common view point on wrappers. The generality of wrappers depends much on the classification algorithm used during the feature selection process. Wrappers using a simple classification algorithm (e.g. NB and KNN) can be general to other classification algorithms, i.e. the features selected can increase the classification performance of other classifiers over using all features. On the other hand, a wrapper using a complicated classification algorithm (e.g. SVM) is usually not general to other (simple) classification algorithms.

8.3 Future Work

This section highlights key areas of future work.

8.3.1 Feature Selection on Large-scale Problems

This thesis focuses mainly on the classification problems with less than one thousand features. In many problems, such as biological datasets, the number of features can be a few thousands or more than ten thousands. Those tasks belong to the category of large-scale problems. Feature selection is almost a necessary step before conducting classification because most classification algorithms perform poorly on such datasets. PSO has

shown its ability on datasets tested in this thesis. In future, it is interesting to investigate the capability of PSO for feature selection on large-scale datasets.

8.3.2 Evaluation Methods for Discrete Multi-objective Algorithms

There are many metrics or measures to evaluate the performance of a multi-objective algorithm or compare the performance of two or more multi-objective algorithms. However, almost all of those measures are designed for continuous multi-objective algorithms. There is no well defined metrics or measures for comparing the performance of discrete multi-objective algorithms. Therefore, it is needed to investigate evaluation methods to test the performance of discrete multi-objective algorithms before developing novel multi-objective feature selection approaches.

8.3.3 PSO for Multi-objective Feature Selection

This thesis proposed the *first* multi-objective feature selection approach, which has shown that a multi-objective algorithm can provide better solutions than a single objective algorithm. The capability of PSO for multi-objective feature selection has not been fully investigated, such as the use of the preference on the objective of minimising the number of features during the evolutionary process. Therefore, it is interesting and necessary to further investigate the use of PSO for multi-objective (filter or wrapper) feature selection to better address the problems.

8.3.4 PSO for Feature Construction

Feature construction aims to construct a new high-level feature, which is usually a function of the original low-level features and mathematical/expression operators. The selection of the original features is one of

the key aspects.

This thesis has shown that PSO can successfully select a subset of original features to increase the classification performance, which motivates the idea of using PSO for feature construction. We proposed the first PSO based feature construction algorithm, which can be seen in [183]. The method in [183] is an initial work on PSO for feature construction, but has shown that PSO has the potential to address feature construction problems.

A major limitation in [183] is that PSO was only used to select original low-level features. The selection of function operators were achieved by an inner loop. This is mainly because the standard representation in PSO is not able to evolve nominal/categorical variables. Therefore, a new representation scheme is needed to use PSO for feature construction. Of course, there will be more other works needed to further investigate the capability of PSO for feature construction.

8.3.5 New PSO Algorithms

Most of the PSO applications, including feature selection, use continuous PSO. Only a small part of applications use discrete PSO, i.e. BPSO. BPSO preserves the fundamental concept of the PSO algorithm, that is, the knowledge is optimised by social interactions within the population. However, not all important characteristics of the PSO algorithm are completely present in BPSO. Therefore, from both the application point of view and the development of PSO itself, it is needed to proposed new discrete or binary PSO algorithms. New PSO algorithms can be developed by proposing new representations, new topologies and updating mechanisms.

Representation

In current BPSO, the dimensionality of the search space is the number of variables/features. A binary string is used to encode the potential solution. This encoding scheme makes a feature selection task a high-dimensional

complex problem if the number of available features is large. A good encoding scheme to avoid such a situation needs to be investigated. Meanwhile, the binary string representation only shows the selection of each individual feature. To better address feature selection, it would be helpful to show the relationship/interaction between different features in the representation. In addition, a new representation scheme is also needed to use PSO for feature construction.

There is only one discrete PSO, i.e. binary PSO (BPSO), but not all discrete problems are binary problems. A new representation scheme, which can deal with other types of discrete problems, is needed to extend the use of the PSO algorithm.

Search Mechanisms

The limitation of current BPSO is mainly because of the velocity and position updating mechanisms. Therefore, it is needed to develop new updating mechanisms to show all important characteristics of the PSO algorithm. Meanwhile, new representation schemes may also require new search mechanisms.

Topology

Topology structure is one of the key elements that influence the performance of PSO. Research has shown the effects of topology on the performance of the original continuous PSO, but the influence of topology on discrete PSO has not been investigated. Meanwhile, discrete or binary problems have different characteristics from continuous problems. Therefore, it is needed to develop new topologies to improve the performance of discrete PSO.

Bibliography

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

[2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (Second Edition)*. Pearson Education, 2003.

[3] H. M. Zhao, A. P. Sinha, and W. Ge, "Effects of feature construction on classification performance: An empirical study in bank failure prediction," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2633–2644, 2009.

[4] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern Recognition*, vol. 43, no. 1, pp. 5–13, 2010.

[5] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, vol. 97, pp. 273–324, 1997.

[6] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[7] H. Liu and H. Motoda, *Feature Selection for Knowledge Discovery and Data Mining*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[8] L. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Feature selection using multi-objective genetic algorithms for handwritten digit recognition," in *16th International Conference on Pattern Recognition (ICPR'02)*, vol. 1, pp. 568– 571, 2002.

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

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

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *IEEE International Conference on Evolutionary Computation (CEC'98)*, pp. 69–73, 1998.

[12] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. Evolutionary Computation Series, San Francisco: Morgan Kaufman, 2001.

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

[14] 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.

[15] A. Mohemmed, M. Zhang, and M. Johnston, "Particle swarm optimization based adaboost for face detection," in *IEEE Congress on Evolutionary Computation (CEC'09)*, pp. 2494–2501, 2009.

[16] 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)*, pp. 2093–2098, 2008.

[17] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theoretical Computer Science*, vol. 209, pp. 237–260, 1998.

[18] M. Mitchell, *An Introduction to Genetic Algorithms*. The MIT Press, 1996.

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

[20] K. Waqas, R. Baig, and S. Ali, "Feature subset selection using multi-objective genetic algorithms," in *IEEE 13th International Conference on Multitopic Conference (INMIC'09)*, pp. 1–6, 2009.

[21] L. Ke, Z. Feng, Z. Xu, K. Shang, and Y. Wang, "A multiobjective ACO algorithm for rough feature selection," in *Second Pacific-Asia Conference on Circuits,Communications and System (PACCS)*, vol. 1, pp. 207–210, 2010.

[22] A. Gutierrez, M. Lanza, I. Barriuso, L. Valle, M. Domingo, J. Perez, and J. Basterrechea, "Comparison of different pso initialization techniques for high dimensional search space problems: A test with fss and antenna arrays," in *Antennas and Propagation (EUCAP), Proceedings of the 5th European Conference on*, pp. 965 –969, 2011.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182 –197, 2002.

[24] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," in *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pp. 95–100, 2002.

[25] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[26] E. Alpaydin, *Introduction to machine learning*. The MIT Press, 2004.

[27] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach*. Tioga, 1983.

[28] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[29] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Wadsworth International Group, 1984. BREIMAN84.

[30] B. H.B., "Unsupervised learning," *Neural Computation*, vol. 1, no. 3, p. 295311, 1989.

[31] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st ed., 1998.

[32] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.

[33] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[34] A. O. Finley and R. E. McRoberts, "Efficient k-nearest neighbor searches for multi-source forest attribute mapping," *Remote Sensing of Environment*, vol. 112, no. 5, pp. 2203 – 2211, 2008.

[35] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Chapman & Hall, New York, NY, 1984.

[36] R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[37] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[38] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," 2003.

[39] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[40] J. Weston and C. Watkins, "Support vector machines for multi-class pattern recognition.," *ESANN*, vol. 99, p. 6172, 1999.

[41] U. H.-G. KreBel, "Advances in kernel methods," ch. Pairwise classification and support vector machines, pp. 255–268, Cambridge, MA, USA: MIT Press, 1999.

[42] M. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18 –28, 1998.

[43] J. Valyon and G. Horváth, "A WEIGHTED GENERALIZED LS–SVM," 2003.

[44] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *IN PROCEEDINGS OF THE TENTH NATIONAL CONFERENCE ON ARTI CIAL INTELLIGENCE*, pp. 223–228, MIT Press, 1992.

[45] I. Rish in *IJCAI-01 workshop on "Empirical Methods in AI"*, 2005.

[46] D. Michie, D. J. Spiegelhalter, and C. Taylor, "Machine learning, neural and statistical classification," 1994.

[47] D. Koller and M. Sahami, "Toward optimal feature selection," in *International Workshop And Conference on Machine Learning*, pp. 284–292, Morgan Kaufmann, 1996.

[48] P. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on Computers*, vol. 26, no. 9, pp. 917–922, 1977.

[49] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, pp. 129–134, 1992.

[50] P. Langley, "Selection of relevant features in machine learning," in *Proceedings of the AAAI Fall symposium on relevance*, pp. 127–131, AAAI Press, 1994.

[51] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *Machine Learning: Proceedings of the Eleventh International Conference (ICCCS'11)*, pp. 121–129, Morgan Kaufmann Publishers, 1994.

[52] C. S. Yang, L. Y. Chuang, and J. C. Li, "Chaotic maps in binary particle swarm optimization for feature selection," in *IEEE Conference on Soft Computing in Industrial Applications (SMCIA '08)*, pp. 107–112, 2008.

[53] A. Purohit, N. Chaudhari, and A. Tiwari, "Construction of classifier with feature selection based on genetic programming," in *IEEE Congress on Evolutionary Computation (CEC'10)*, pp. 1–5, 2010.

[54] W. Duch, T. Winiarski, J. Biesiada, and A. Kachel, "Feature ranking, selection and discretization," in *In Proceedings of Int. Conf. on Artificial Neural Networks (ICANN*, pp. 251–254, Bogazici University Press, 2003.

[55] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, "Ranking a random feature for variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1399–1414, 2003.

[56] M. Last, A. Kandel, and O. Maimon, "Information-theoretic algorithm for feature selection," *Pattern Recognition Letters*, vol. 22, pp. 799–811, 2001.

[57] K. Neshatian and M. Zhang, "Genetic programming for feature subset ranking in binary classification problems," in *European Conference on Genetic Programming*, (Berlin, Heidelberg), pp. 121–132, Springer-Verlag, 2009.

[58] K. Neshatian, M. Zhang, and P. Andreae, "Genetic programming for feature ranking in classification problems," in *Simulated Evolution and Learning*, vol. 5361 of *Lecture Notes in Computer Science*, pp. 544–554, Springer Berlin / Heidelberg, 2008.

[59] R. Ruiz, J. C. R. Santos, and J. S. Aguilar-Ruiz, "Fast feature ranking algorithm," in *Knowledge-Based Intelligent Information and Engineering Systems (KES'03)*, vol. 2773 of *Lecture Notes in Computer Science*, pp. 325–331, Springer, 2003.

[60] K. Krawiec, "Genetic programming-based construction of features for machine learning and knowledge discovery tasks," *Genetic Programming and Evolvable Machines*, vol. 3, no. 4, pp. 329–343, 2002.

[61] H. Vafaie and K. DeJong, "Feature space transformation using genetic algorithms," *IEEE Intelligent Systems*, vol. 13, no. 2, pp. 57–65, 1998.

[62] F. Otero, M. Silva, A. Freitas, and J. Nievola, "Genetic programming for attribute construction in data mining," in *Genetic Programming*, vol. 2610 of *Lecture Notes in Computer Science*, pp. 101–121, Springer Berlin/Heidelberg, 2003.

[63] M. A. Muharram and G. D. Smith, "The effect of evolved attributes on classification algorithms," in *Australian Conference on Artificial*

Intelligence (AI'03), vol. 2903 of *Lecture Notes in Computer Science*, pp. 933–941, Springer, 2003.

[64] M. G. Smith and L. Bull, "Genetic programming with a genetic algorithm for feature construction and selection," *Genetic Programming and Evolvable Machines*, vol. 6, no. 3, pp. 265–281, 2005.

[65] J. H. Holland, *Adaptation in natural and artificial systems*. MIT Press, 1992.

[66] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[67] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[68] J. R. Koza, "Introduction to genetic programming," in *Annual conference on Genetic and evolutionary computation (GECCO'07)*, vol. 5, pp. 3323–3365, 2007.

[69] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies – a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, p. 352, 2002.

[70] L. J. Fogel, *Intelligence through simulated evolution: forty years of evolutionary programming*. New York, NY, USA: John Wiley & Sons, Inc., 1999.

[71] Y. Guo, W. Li, A. Mileham, and G. Owen, "Applications of particle swarm optimisation in integrated process planning and scheduling," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 2, pp. 280–288, 2009.

[72] M. AlRashidi and M. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 913–918, 2009.

[73] T. J. Ai and V. Kachitvichyanukul, "Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem," *Computers and Industrial Engineering*, vol. 56, no. 1, pp. 380– 387, 2009.

[74] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Scituate, MA, USA: Bradford Company, 2004.

[75] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *IEEE Congress on Evolutionary Computation (CEC 99)*, vol. 2, pp. 1470–1477, 1999.

[76] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28 –39, 2006.

[77] J. Holland, L. Booker, M. Colombetti, M. Dorigo, and D. e. a. Goldberg, "What is a learning classifier system ?," in *Learning Classifier Systems*, vol. 1813 of *Lecture Notes in Computer Science*, p. 332, Springer Berlin Heidelberg, 2000.

[78] S. Wilson, "Get real! xcs with continuous-valued inputs," in *Learning Classifier Systems*, vol. 1813 of *Lecture Notes in Computer Science*, p. 209219, Springer Berlin Heidelberg, 2000.

[79] R. Storn and K. Price, "Differential evolution a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, p. 341359, 1997.

[80] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series, Berlin, Germany: Springer-Verlag, 2005.

[81] L. R. d. Castro and J. Timmis, *Artificial Immune Systems: A New Computational Intelligence Paradigm*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[82] D. Dasgupta, "An overview of artificial immune systems and their applications," in *Artificial Immune Systems and Their Applications*, p. 321, Springer Berlin Heidelberg, 1999.

[83] L. A. G. COELLO C OELLO, C. and D. VELDHUIZEN, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation Series)*. Springer, 2007.

[84] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *IEEE International Conference on Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation.*, vol. 5, pp. 4104–4108, 1997.

[85] M. Richards and D. Ventura, "Choosing a starting configuration for particle swarm optimization," in *IEEE International Joint Conference on Neural Networks*, vol. 3, pp. 2309–2312, 2004.

[86] Q. Du, V. Faber, and M. Gunzburger, "Centroidal voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, pp. 637–676, 1999.

[87] K. Parsopoulos and M. Vrahatis, "Initializing the particle swarm optimizer using the nonlinear simplex method," *Advances in intelligent systems, fuzzy systems, evolutionary computation*, vol. 216, 2002.

[88] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[89] 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.

[90] H. Jabeen, Z. Jalil, and A. R. Baig, "Opposition based initialization in particle swarm optimization (o-pso)," in *Proceedings of the 11th*

Annual Conference Companion on Genetic and Evolutionary Computation Conference, GECCO'09, pp. 2047–2052, ACM, 2009.

[91] H. Wang, Z. Wu, J. Wang, X. Dong, S. Yu, and C. Chen, "A new population initialization method based on space transformation search," in *Fifth International Conference on Natural Computation (ICNC'09)*, vol. 5, pp. 332 –336, IEEE Computer Society, 2009.

[92] H. Wang, H. Li, Y. Liu, C. Li, and S. Zeng, "Opposition-based particle swarm algorithm with cauchy mutation," in *IEEE Congress on Evolutionary Computation (CEC 2007)*, pp. 4750 –4756, 2007.

[93] 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.

[94] H.-C. Chen and O.-C. Chen, "Particle swarm optimization incorporating a preferential velocity-updating mechanism and its applications in iir filter design," in *IEEE International Conference on Systems, Man and Cybernetics (SMC'06)*, vol. 2, pp. 1190–1195, 2006.

[95] J. Qi and Y. Ding, "An improved particle swarm optimization with an adaptive updating mechanism," in *Proceedings of the Second international conference on Advances in swarm intelligence - Volume Part I*, ICSI'11, (Berlin, Heidelberg), pp. 130–137, Springer-Verlag, 2011.

[96] P. Ngatchou, A. Zarei, and M. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems*, pp. 84–91, 2005.

[97] K. Iswandy and A. Koenig, "Feature-level fusion by multi-objective binary particle swarm based unbiased feature selection for optimized sensor system design," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 365 –370, 2006.

[98] R. Fdhila, T. Hamdani, and A. Alimi, "Distributed mopso with a new population subdivision technique for the feature selection," in *International Symposium on Computational Intelligence and Intelligent Informatics (ISCIII'11)*, pp. 81 –86, 2011.

[99] S. Adra, I. Griffin, and P. Fleming, "A comparative study of progressive preference articulation techniques for multiobjective optimisation," in *Evolutionary Multi-Criterion Optimization*, vol. 4403 of *Lecture Notes in Computer Science*, p. 908921, Springer Berlin Heidelberg, 2007.

[100] M. Reyes-Sierra and C. Coello Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[101] C. A. Coello Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[102] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation," in *Proceedings of the Congress on Evolutionary Computation (CEC'99)*, vol. 1, pp. 98–105, 1999.

[103] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Annual conference on Genetic and evolutionary computation (GECCO'03)*, pp. 37–48, 2003.

[104] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multiobjective optimization using crowding, mutation and epsilon-dominance," in *EMO*, pp. 505–519, 2005.

[105] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *Evolutionary Computation, IEEE Transactions on*, vol. 11, pp. 712–731, Dec 2007.

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

[107] T. M. Cover and J. A. Thomas, *Elements of information theory*. New York, NY, USA: Wiley-Interscience, 1991.

[108] T. M. Cover and J. A. Thomas, *Information Theory and Statistics*, p. 279335. John Wiley & Sons, Inc., 2001.

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

[110] 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.

[111] S. Stearns, "On selecting features for pattern classifier," in *Proceedings of the 3rd International Conference on Pattern Recognition*, (Coronado, Calif, USA), pp. 71–75, IEEE Press, 1976.

[112] P. Pudil, J. Novovicova, and J. V. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[113] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, pp. 525–534, 2009.

[114] M. Gutlein, E. Frank, M. Hall, and A. Karwath, "Large-scale attribute selection using wrappers," in *IEEE Symposium on Computa-*

tional Intelligence and Data Mining (CIDM '09), pp. 332–339, IEEE, 2009.

[115] S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[116] D. Muni, N. Pal, and J. Das, "Genetic programming for simultaneous feature selection and classifier design," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 1, pp. 106–117, 2006.

[117] R. K. Sivagaminathan and S. Ramakrishnan, "A hybrid approach for feature subset selection using neural networks and ant colony optimization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 49–60, 2007.

[118] N. Kwak and C. H. Choi, "Input feature selection for classification problems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 143–159, 2002.

[119] L. Yu and H. Liu, "Efficient feature selection via analysis of relevance and redundancy," *Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[120] H. Yuan, S. S. Tseng, and W. Gangshan, "A two-phase feature selection method using both filter and wrapper," in *IEEE International Conference on Systems, Man, and Cybernetics (SMC'99)*, vol. 2, pp. 132–136, 1999.

[121] P. Estevez, M. Tesmer, C. Perez, and J. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[122] H. Almuallim and T. G. Dietterich, "Efficient algorithms for identifying relevant features," in *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, pp. 38–45, Morgan Kaufmann, 1992.

[123] H. Almuallim and T. G. Dietterich, "Learning boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, pp. 279–305, 1994.

[124] K. Kira and L. A. Rendell, "A practical approach to feature selection," *Assorted Conferences and Workshops*, pp. 249–256, 1992.

[125] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," *Lecture Notes in Computer Science*, vol. 784, p. 171, 1994.

[126] C. Cardie, "Using decision trees to improve case-based learning," in *Proceedings of the Tenth International Conference on Machine Learning (ICML)*, pp. 25–32, 1993.

[127] M. A. Hall, *Correlation-based Feature Subset Selection for Machine Learning*. PhD thesis, The University of Waikato, Hamilton, New Zealand, 1999.

[128] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 143–159, 2002.

[129] S. Foithong, O. Pinngern, and B. Attachoo, "Feature subset selection wrapper based on mutual information and rough sets," *Expert Systems with Applications*, vol. 39, no. 1, p. 574584, 2012.

[130] 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.

[131] S. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, 1992.

[132] H. Liu, J. Sun, L. Liu, and H. Zhang, "Feature selection with dynamic mutual information," *Pattern Recognition*, vol. 42, no. 2, pp. 1330–1339, 2009.

[133] 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.

[134] B. Chakraborty, "Genetic algorithm with fuzzy fitness function for feature selection," in *IEEE International Symposium on Industrial Electronics (ISIE'02)*, vol. 1, pp. 315– 319, 2002.

[135] K. Neshatian and M. Zhang, "Pareto front feature selection: using genetic programming to explore feature space," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation (GECCO'09)*, (New York, NY, USA), pp. 1027–1034, 2009.

[136] R. Jensen, "Performing feature selection with ACO," in *Swarm Intelligence in Data Mining*, vol. 34 of *Studies in Computational Intelligence*, pp. 45–73, Springer Berlin / Heidelberg, 2006.

[137] Y. Chen, D. Miao, and R. Wang, "A rough set approach to feature selection based on ant colony optimization," *Pattern Recognition Letters*, vol. 31, no. 3, pp. 226–233, 2010.

[138] G. Azevedo, G. Cavalcanti, and E. Filho, "An approach to feature selection for keystroke dynamics systems based on PSO and feature weighting," in *IEEE Congress on Evolutionary Computation (CEC'07)*, pp. 3577–3584, 2007.

[139] Y. Marinakis, M. Marinaki, and G. Dounias, "Particle swarm optimization for pap-smear diagnosis," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1645 – 1656, 2008.

[140] S.-W. Lin and S.-C. Chen, "Psolda: A particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis," *Appled Soft Computing*, vol. 9, no. 3, pp. 1008–1015, 2009.

[141] C. L. Huang and J. F. Dun, "A distributed PSO-SVM hybrid system with feature selection and parameter optimization," *Application on Soft Computing*, vol. 8, pp. 1381–1391, 2008.

[142] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of the Second European Conference on Computational Learning Theory*, EuroCOLT '95, (London, UK, UK), pp. 23–37, Springer-Verlag, 1995.

[143] L. Y. Chuang, S. W. Tsai, and C. H. Yang, "Improved binary particle swarm optimization using catfish effect for feature selection," *Expert Systems with Applications*, vol. 38, pp. 12699–12707, 2011.

[144] S. Vieira, L. Mendonca, G. Farinha, and J. Sousa, "Metaheuristics for feature selection: Application to sepsis outcome prediction," in *IEEE Congress on Evolutionary Computation (CEC'12)*, pp. 1 –8, 2012.

[145] E. Alba, J. Garcia-Nieto, and L. Jourdan, "Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms," in *IEEE Congress on Evolutionary Computation (CEC'07)*, pp. 284–290, 2007.

[146] E. G. Talbi, L. Jourdan, J. Garcia-Nieto, and E. Alba, "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, 2008.

[147] Z. Pawlak, "Rough sets," *International Journal of Parallel Programming*, vol. 11, pp. 341–356, 1982.

[148] B. Chakraborty, "Feature subset selection by particle swarm optimization with fuzzy fitness function," in *3rd International Conference on Intelligent System and Knowledge Engineering (ISKE'08)*, vol. 1, pp. 1038–1042, 2008.

[149] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[150] M. A. Esseghir, G. Goncalves, and Y. Slimani, "Adaptive particle swarm optimizer for feature selection," in *international conference on Intelligent data engineering and automated learning (IDEAL'10)*, (Berlin, Heidelberg), pp. 226–233, Springer Verlag, 2010.

[151] H. Ming, "A rough set based hybrid method to feature selection," in *International Symposium on Knowledge Acquisition and Modeling (KAM'08)*, pp. 585–588, IEEE, 2008.

[152] Z. X. Zhu, Y. S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 37, no. 1, pp. 70–76, 2007.

[153] T. M. Hamdani, J.-M. Won, A. M. Alimi, and F. Karray, "Multi-objective feature selection with NSGA II," in *8th International Conference on Adaptive and Natural Computing Algorithms (ICANNGA'07) Part I*, vol. 4431, pp. 240–247, Springer Berlin Heidelberg, 2007.

[154] R. Ramirez and M. Puiggros, "An evolutionary computation approach to cognitive states classification," in *IEEE Congress on Evolutionary Computation (CEC'07)*, pp. 1793–1799, 2007.

[155] B. C. Chien and J. H. Yang, "Features selection based on rough membership and genetic programming," in *IEEE International Conference on Systems, Man and Cybernetics (SMC'06)*, vol. 5, pp. 4124–4129, 2006.

[156] K. Neshatian, *Feature Manipulation with Genetic Programming*. PhD thesis, Victoria University of Wellington, Wellington, New Zealand, 2010.

[157] K. Neshatian, M. Zhang, and M. Johnston, "Feature construction and dimension reduction using genetic programming," in *Australian Conference on Artificial Intelligence (AI'07)*, vol. 4830 of *Lecture Notes in Computer Science*, pp. 160–170, Springer, 2007.

[158] F. Otero, M. Silva, A. Freitas, and J. Nievola, "Genetic programming for attribute construction in data mining," in *Genetic Programming*, vol. 2610 of *Lecture Notes in Computer Science*, pp. 101–121, Springer Berlin/Heidelberg, 2003.

[159] K. Neshatian and M. Zhang, "Genetic programming for performance improvement and dimensionality reduction of classification problems," in *IEEE Congress on Evolutionary Computation (CEC'08)*, pp. 2811–2818, 2008.

[160] K. Neshatian, M. Zhang, and P. Andreae, "A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 5, pp. 645 –661, 2012.

[161] H. H. Gao, H. H. Yang, and X. Y. Wang, "Ant colony optimization based network intrusion feature selection and detection," in *Interna-*

tional Conference on Machine Learning and Cybernetics, vol. 6, pp. 3871–3875, 2005.

[162] R. Bello, A. Nowe, Y. Caballero, Y. Gómez, and P. Vrancx, "A model based on ant colony system and rough set theory to feature selection," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO'05, (New York, NY, USA), pp. 275–276, ACM, 2005.

[163] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1351–1357, 2008.

[164] R. A. A. M. Mishra, D. and T. Jena., "Rough ACO: A hybridized model for feature selection in gene expression data," *International Journal of Computer Communication and Technology*, vol. 1, pp. 85–98, 2009.

[165] R. Jensen and Q. Shen, "Finding rough set reducts with ant colony optimization," in *Proceedings of the 2003 UK Workshop on Computational Intelligence*, pp. 15–22, 2003.

[166] H. R. Kanan and K. Faez, "An improved feature selection method based on ant colony optimization (ACO) evaluated on face recognition system," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 716–725, 2008.

[167] S. Luke and L. Panait, "Lexicographic parsimony pressure," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2002)*, pp. 829–836, Morgan Kaufmann, 2002.

[168] R. Caruana and D. Freitag, "Greedy attribute selection," in *International Conference on Machine Learning (ICML'94)*, pp. 28–36, Morgan Kaufmann, 1994.

[169] K. Dobbin and R. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC Medical Genomics*, vol. 4, no. 1, p. 31, 2011.

[170] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, pp. 83–85, 2005.

[171] A. M. Law, *Simulation Modeling and Analysis, Fourth Edition*. New York, USA: McGraw-Hill Higher Education, 2007.

[172] S. Siegel, *Nonparametric statistics for the behavioral sciences*. New York, USA: McGraw-Hill Higher Education, 2007.

[173] F. Van Den Bergh, *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, Faculty of Natural and Agricultural Science, Pretoria, South Africa, 2001.

[174] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: optimal distributions and the choice of the reference point," in *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*, FOGA '09, (New York, NY, USA), pp. 87–102, ACM, 2009.

[175] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2005.

[176] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001.

[177] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, pp. 760–771, 2011.

[178] 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*, vol. 7691 of *Lecture Notes in Computer Science*, pp. 313–325, Springer, 2012.

[179] Y. Yao and Y. Zhao, "Attribute reduction in decision-theoretic rough set models," *Information Sciences*, vol. 178, no. 17, pp. 3356 – 3373, 2008.

[180] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, 2011.

[181] C. Elkan, "Nearest neighbor classication," Technical Report http://cseweb.ucsd.edu/users/elkan/250Bwinter2011/, University of California, 2011.

[182] J. Su and H. Zhang, "A fast decision tree learning algorithm," in *Proceedings of the 21st national conference on Artificial intelligence – Volume 1*, AAAI'06, pp. 500–505, AAAI Press, 2006.

[183] B. Xue, M. Zhang, Y. Dai, and W. N. Browne, "PSO for feature construction and binary classification," in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pp. 137–144, 2013.