VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Particle Swarm Optimisation and Statistical Clustering for Feature Selection

Mitchell C. Lane

Supervisors: Mengjie Zhang, Bing Xue

Submitted in partial fulfilment of the requirements for
Bachelor of Engineering (Hons).

Abstract

Feature selection is important in classification tasks. By removing irrelevant and redundant features, the accuracy, understandability and training time of a learned classifier can be improved. However, feature selection is complex due to a potentially large search space and unpredictable feature interaction. This project represents the *first* work using statistical clustering information in particle swarm optimisation (PSO) for feature selection in general classification tasks. Seven single-objective PSO-based feature selection approaches are proposed to select a small number of features based on the statistical clustering information. A further two multi-objective PSO-based feature selection approaches are proposed to evolve non-dominated feature subsets in terms of their dimensionality and classification performance. Experimental results show that by using the statistical clustering information in PSO for feature selection, many irrelevant and redundant features are removed and the classification performance is significantly improved over using all features. Furthermore, the two multi-objective algorithms are found to outperform standard Binary PSO for feature selection.

Acknowledgements

I wish to acknowledge the research into the statistical clustering of data undertaken by Ivy Liu, which has been used to cluster features that are used in the PSO algorithms presented in this project. I would like to also express my great appreciation to my two supervisors, Mengjie Zhang and Bing Xue, which have poured countless hours into helping me complete my research project to a high standard.

Contents

Chapter 1

Introduction

Classification is an important task in machine learning and data mining that aims to accurately classify each instance with its correct class label according to the information described by its features. In order to learn how to classify instances with a high level of accuracy, a classification algorithm is trained on a set of instances that are representatives of the classification problem and then tested on the unseen instances. In many cases, the dataset used for training and testing the classification algorithm contains a significantly large number of features. This is known as "the curse of dimensionality" [1] with each feature representing a dimension in the search space. Many of these features are completely irrelevant or redundant, which leads to extensive classifier training time and poor classification performance. An important technique that can reduce the number of irrelevant and redundant features is feature selection [2]. By reducing the number of irrelevant and redundant features, feature selection can simplify the structural complexity of the learned classifiers and/or improve the classification performance [3, 4]. However, developing an effective feature selection algorithm is challenging due to complex feature interaction within a potentially large search space [2]. Consequently, many statistical methods and machine learning techniques have been adopted to perform the task of feature selection.

The task of feature selection is usually over a large search space in which an exhaustive search would take too much time in most cases [5]. Therefore, an effective and efficient search technique is needed. Greedy algorithms such as sequential forward selection (SFS) [6] and sequential backward selection (SBS) [7] have been proposed to reduce the time taken to select optimal feature subsets. However, these greedy approaches are likely to prematurely converge to a local optimum, ineffectively searching the global solution space.

To provide better global search capabilities, evolutionary computation (EC) techniques have been applied to feature selection problems, including genetic algorithms (GAs) [8], genetic programming (GP) [9], ant colony optimisation (ACO) [10] and particle swarm optimisation (PSO) [11, 12, 13, 14, 15]. PSO is based on swarm intelligence and uses a swarm of particles to search for an optimal solution where each particle represents a possible solution in the search space. Compared with GAs and GP, PSO is easier to implement due to its simple particle representation and use of simple mathematical functions. It also has fewer parameters to tune, is computationally less expensive and can converge more quickly [16].

Traditionally, feature selection approaches are single-objective algorithms in which the classification performance is maximised by evaluating the goodness of solutions (feature subset) based solely on their classification performance. However, feature selection is a multi-objective problem where the dimensionality of the feature subset is also desired to be minimised. With fewer features, classifier training time can be reduced and as fewer features are used in constructing the classifier, its structure is likely to be less complex with greater generalisation on the unseen data. Multi-objective feature selection is a complex task as its two objectives often conflict with each other. This happens as the classification error rate is likely to increase when fewer features can be used to distinguish the correct class label of a given instance. Recently, ideas from Multi-Objective Evolutionary algorithms (MOEa's) such as the improved Non-dominated Sorting Genetic Algorithm (NSGA-II) [17], the Pareto Envelope-based Selection Algorithm (PESA) [18] and the improved Stength Pareto Evolutionary Algorithm (SPEA2) [19] have been adopted to solve feature selection as a multi-objective problem, in which the two objectives are to minimise the classification error rate and minimise the dimensionality of the evolved feature subsets. Unlike single-objective feature selection approaches, multi-objective feature selection approaches such as a modified version of NSGA-II for feature selection [20] and Non-dominated Sorting Particle Swarm Optimizer (NSPSO) [21] provide multiple non-dominated feature subsets from which the user can select a single solution according to their desired trade off between the classification error rate and the dimensionality of the feature subset.

Many statistical measures have been used to evaluate the goodness of feature subsets chosen by a feature selection algorithm [3, 22]. Statistical approaches can also be used as a preprocessing step to group related features together into a feature cluster, producing a set of feature clusters that can then be used as input to a feature selection algorithm. By doing this, the search over feature subsets can use the statistical information provided by the feature clusters in a number of beneficial ways. For example, the statistical clustering information can be used to identify and eliminate irrelevant and redundant features within the same cluster. Furthermore, the search can be modified to better uncover complex feature interaction within features of different clusters. In order to utilise the benefits of statistical clustering as a preprocessing step, novel feature selection approaches need to be developed that can select optimal feature subsets based on a set of feature clusters.

1.1 Objectives

The overall goal of this project is to investigate a PSO-based feature selection approach that can effectively use the information from the statistical clustering of features. The statistical clustering method used in this paper groups features together based on a statistical model that considers feature interaction [23, 24]. Features in the same cluster are similar and they are dissimilar to features in other clusters. As feature interaction is an important issue in feature selection, the statistical feature interaction information found by the clustering method can be used to discover good feature subsets. By using this information, the new PSO-based feature selection approaches are expected to minimise the number of features selected, whilst maintaining or increasing the classification accuracy. The implementation of the clustering method is out of scope of this project and hence, will not be discussed in detail. The performance of each feature selection approach will be examined on a number of benchmark datasets with different numbers of features and instances. The overall goal is split into three objectives:

- Objective 1 is to develop a single-objective PSO-based approach to select a single feature from each feature cluster.
- Objective 2 is to extend the single-objective PSO-based approach to select multiple features from each cluster using statistical information to further improve the classification performance.
- Objective 3 is to develop a multi-objective PSO-based approach that uses the statistical clustering information to solve feature selection as a multi-objective task and obtain a set of non-dominated feature subsets.

1.2 Major Contributions

The work in this project represents the *first* research in PSO for feature selection based on statistical clustering information for general classification problems. Although there exists one other known paper that groups features as a preprocessing step to PSO for feature selection [25], the approach presented in this paper is used only for cancer microarray data. The research presented in this project is also significantly different, resulting in a range of novel PSO-based feature selection algorithms. In [25], the clustering step removes many potentially useful features, resulting in a small subset of features being used as input to PSO. However, in this project, the clustering step does not remove any features. Instead, the statistical information provided by the clustering step is used within the newly developed PSO-based feature selection algorithms to intelligently evolve feature subsets containing complimentary features.

Within this project, there are three major contributions. Firstly, four new PSO-based feature selection algorithms are developed which use statistical clustering information for the first time, to select a single feature from each feature cluster. When combined with a statistical clustering technique that allows the specification of the number of feature clusters, these new algorithms offer a unique option to users to predefine the number of features to be selected through the PSO feature selection algorithm. The work that forms this major contribution has been accepted by the 26th Australasian Joint Conference on Artificial Intelligence [26] and will be published by Springer in the Lecture Notes in the Artificial Intelligence (LNAI) series.

The second major contribution is three new PSO-based feature selection algorithms that use statistical clustering information to automatically discover complimentary features and feature interaction, intelligently evolving feature subsets that can contain multiple or zero features from each feature cluster. This contribution represents the first work that uses statistical clustering information within a PSO-based approach to discover complimentary features and feature interaction to select feature subsets with low dimensionality and that achieve high classification accuracy.

The third major contribution is two new Multi-objective PSO-based feature selection algorithms that can evolve a set of non-dominated feature subsets with different classification performances and dimensionalities. Ideas from NSGA-II [17] and a PSO-based multi-objective optimisation method [27] are adopted in the development of these two new algorithms, which are based on non-dominated sorting and statistical clustering information. The research in this contribution represents the first time PSO has been used together with statistical clustering information to solve feature selection as a multi-objective problem. The second and third major contributions are also under preparation for submission to the EvoStar 2014 conference [28] and another related Evolutionary Computation journal.

1.3 Organisation

The remainder of this report is organised as follows. Background information is provided in Chapter 2. Chapters 3, 4 and 5 discuss Objectives 1, 2 and 3, respectively. These chapters include in-depth design and implementation details of each novel feature selection approach. Each of these chapters also describe the experimental design that is used to test the effectiveness of each feature selection approach, accompanied by a discussion upon the experimental results and observations. Chapter 6 provides a discussion of the major conclusions drawn from this project and some remaining future work.

Chapter 2

Background

2.1 Machine Learning and Classification

Machine learning is a major branch of Artificial Intelligence that focuses on the design and development of systems that can learn from data in order to perform some useful task. There are many different algorithms used for machine learning and there exists a wide variety of tasks that machine learning has been successfully applied to. Classification algorithms are one of the most prominently used machine learning techniques with many different implementations applied to a range of different areas, including autonomous robotics, medical data analysis, text classification, and personal software agents [29, 30, 31, 32, 33]. These classification algorithms are trained on a set of instances and tested on unseen instances, aiming to accurately classify each instance with its correct class label according to its features. Although many different classification algorithms have been developed to solve a range of problems, the training process of any classifier can be harmed when training on a dataset that contains a large number of features. The problem of having a large number of features is known as "the curse of dimensionality" [1] and can lead to extensive classifier training time, an overly complex classifier structure and reduced classification accuracy.

2.2 Feature Selection

In most datasets that contain a large number of features, there exist many redundant or irrelevant features which may negatively impact the classification accuracies of trained classification algorithms. Incorporating these features into the classifier training process can result in the evolution of a complex classifier structure that is incomprehensible to humans and fails to generalise well when used on unseen instances. Instead, a small subset of representative and relevant features are desired for training and testing the classification algorithm. Feature selection is a task that can achieve this goal through the selection of relevant features that contribute to high classification accuracy [4]. This can improve the classification accuracy of the learned classifiers, make the classifier learning process faster and make the learned classifier structure more comprehensible due to a reduction in complexity [2]. For these reasons, it has become a popular topic of research in the field of machine learning and data mining. However, feature selection is a challenging task due to complex feature interaction within a potentially large search space. In particular, removing a redundant feature may have a damaging effect on classification performance as two or more features that are found to be redundant by themselves may be useful in combination with each other [2].

There are two main approaches to performing the task of feature selection: wrapper and filter approaches [2]. Wrappers use the power of a learning algorithm to select a useful subset of features from a dataset. The learning algorithm is used to evaluate the goodness of a feature subset according to its classification accuracy on a set of training data. Wrappers are computationally expensive, but can provide a set of features that can achieve better classification accuracy with the same learning algorithm. Contrastingly, filters act as a pre-processing step that aims to select useful subsets of features, independent of any learning algorithm. They are computationally cheaper when compared with wrappers and are able to provide a subset of features without any relation to any learning algorithm.

2.3 Evolutionary Computation

Evolutionary Computation (EC) is a field of Artificial Intelligence that involves techniques based on or inspired by nature, biological mechanisms, social interaction and/or swarm intelligence. Compared to other machine learning techniques such as Neural Networks which optimise a single solution in terms of its error over a series of iterations, EC techniques use a population of individuals as potential solutions and evolves each of the individuals over a series of iterations. In many cases, each individual is updated at each iteration with respect to its fitness (which is given by a user-defined fitness function). At the end of the evolutionary process, the solution with the highest fitness is selected as the best evolved solution. As EC techniques are effective global search methods, they can be used to address feature selection problems with a large search space. Consequently, EC techniques such as genetic algorithms (GAs) [8], genetic programming (GP) [9], ant colony optimisation (ACO) [10] and particle swarm optimisation (PSO) [11, 12, 13, 14, 15] have been successfully applied to feature selection problems, often outperforming traditional feature selection approaches. In the rest of this section, we briefly review PSO and Multi-Objective Evolutionary algorithms since they are used in this project.

2.3.1 Particle Swarm Optimisation

Particle swarm optimisation (PSO) is an evolutionary computation technique proposed in 1995 by Kennedy and Eberhart [34]. The idea for PSO originated from imitating the social behaviours of birds flocking and fish schooling. The algorithm itself maintains a simple representation of solutions and requires only primitive mathematical operators to harmonise local and global knowledge through social interaction [34]. PSO uses a swarm of particles to search for the optimal solution over a number of iterations, where each particle represents a possible solution in the search space.

A crucial step of PSO is the updating of particle positions within the swarm during the evolutionary process. At the position update step, each particle uses its own local knowledge in combination with the neighbourhood's global knowledge to determine a velocity which is then used to move the particle to a new position [34]. By sharing local and global information within the swarm, every particle has the ability to influence the direction of the swarm towards better solutions within the search space. This is a demonstration of adaptability, the fifth basic principle of swarm intelligence [35] which provides an effective heuristic for updating particle positions.

Within PSO, each particle has a position and velocity vector. The position vector $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ represents the position of particle i, where D is the dimensionality of the search space and the velocity vector $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$ is used to represent the velocity of particle i. During the evolutionary process each particle remembers their previous best position (*pbest*) and the best position found so far by any particle in the swarm (*gbest*). These two values are used to update the velocity and the position 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 search process. d denotes the dth dimension in the search space. 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 value of *pbest* and *gbest* in the dth dimension, respectively. w is the inertia weight. The velocity v_{id}^{t} is limited by a predefined maximum velocity v_{max} where $v_{id}^{t} \in [-v_{max}, v_{max}]$.

The original representation of a particle in PSO uses a vector of real values for both the position and velocity vectors, which suits problems operating in continuous space. However, this representation does not suit the discrete feature selection problem, where features are either included or excluded from a particular solution. In 1997, Kennedy and Eberhart proposed binary particle swarm optimisation (BPSO) [36]. In BPSO, each entry p_i in the position vector is a binary value. Each entry v_i in the velocity vector should represent the probability of an element in the position taking the value of 1. To achieve this, a sigmoid function $s(v_{id})$ is used to transform v_{id} to the range of $[0, 1]$. Equation (2.2) is used to update the velocity of each particle. However, a new equation is used for updating the position of each particle as x_{id}, p_{id} and p_{gd} must be restricted to the values of 0 or 1. BPSO updates the position of each particle according to the following equations.

$$x_{id} = \begin{cases} 1 & \text{if } rand() < s(v_{id}) \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

$$s(v_{id}) = \frac{1}{1 + e^{-v_{id}}} \tag{2.4}$$

where $rand()$ is a random number selected from a uniform distribution in $[0, 1]$.

2.3.2 Multi-Objective Evolutionary Algorithms

Multi-objective optimisation involves optimising multiple objective functions which are often in conflict with each other. For example, let x_1 and x_2 be two solutions in a n-objective minimisation problem. If x_1 has an equal or smaller value than x_2 in each of the n objectives and x_1 has a smaller value than x_2 in at least one objective, x_1 is said to *dominate* x_2. When a solution is not dominated by any other solution, it is referred to as a non-dominated or Pareto-optimal solution. The set of these non-dominated solutions is known as the *Pareto front* and forms the trade-off surface over the objectives in the search space.

Recently, Multi-Objective Evolutionary algorithms (MOEa's) such as NSGA-II [17], PESA [18] and SPEA2 [19] have been found to be effective methods for solving multi-objective optimisation problems. NSGA-II [17] is one of the most popular MOEa's and has been used in a variety of multi-objective problems. NSGA-II uses non-dominated sorting to discover a set of solutions that are not dominated by any other solution found so far in the search. A crowded comparison operator is used in NSGA-II as a secondary evaluation measure that ranks each solution within a particular non-dominated front by how crowded the solution is. The crowding distance is then used in the construction of the new population and helps to preserve the diversity throughout the evolutionary process.

Li [21] introduces a modified version of PSO, the Non-dominated Sorting Particle Swarm Optimizer (NSPSO) for multi-objective optimisation. The results on four mathematical functions show NSPSO is fast, reliable and often converges to the true *Pareto front* of the search space with a good spread of solutions. Consequently, NSPSO is found to be highly competitive with NSGA-II and other existing evolutionary multi-objective algorithms.

As feature selection has the two conflicting objectives of minimising the classification error rate and minimising the number of features selected, feature selection tasks can be expressed and solved as multi-objective problems. MOEa's such as NSGA-II [20] and NSPSO [21] have been used to solve feature selection as a multi-objective problem and can evolve multiple non-dominated feature subsets. The user can then choose a solution based on their desired trade off between the classification error rate and feature subset dimensionality. This gives the user more flexibility than single objective feature selection algorithms which output a single, potentially large feature subset that achieves optimal classification performance, regardless of its dimensionality.

2.4 Related Work

2.4.1 Traditional Feature Selection Approaches

The FOCUS algorithm [37] is a traditional filter feature selection algorithm. It first starts with an empty feature subset and exhaustively examines all subsets of features. The algorithm then selects the smallest subset of features that can sufficiently determine the class labels of instances in the training set. The FOCUS algorithm is computationally intensive due to the exhaustive search that results in an exponential increase in computation time as the number of features increase. Hence, the approach is only feasible for use with datasets that contain a small number of features.

In order to avoid a computationally intensive exhaustive search, greedy algorithms have been introduced to solve feature selection problems. SFS [6] and SBS [7] are two greedy wrapper feature selection approaches that both use a hill-climbing search strategy to search for optimal sets of features. SFS starts with an empty set of features and sequentially adds features until the point in which there is no improvement in classification accuracy. Contrastingly, SBS starts with a set of all features and sequentially removes features until a point in which no improvement in classification accuracy can be achieved. As SFS and SBS sequentially select features in a greedy manner, the computation time needed to discover feature subsets is relatively small compared to other feature selection approaches. However, both SFS and SBS suffer from the so-called nesting effect [38], which means that once a feature is selected by SFS or discarded by SBS, it cannot be discarded by SFS or selected by SBS. This traps SFS and SBS searches in local optima.

The "plus-*l*-take-away-*r*" method [39] is proposed to address the limitations of the strict sequential adding or removing of features in SFS and SBS by performing forward selection l times followed by r backward feature removals. The main issue with this method is to determine the optimal values for l and r. To address this problem, the sequential forward floating selection (SFFS) [40] and sequential backward floating selection (SBFS) [40] are developed to automatically determine the values of l and r. Like SFS and SBS, SFFS and SBFS also suffer from getting stuck in local optima, but are at least as good as the best sequential method.

2.4.2 EC Approaches (Non-PSO) for Feature Selection

EC algorithms are well-known for their efficient and effective global search ability. They can often outperform traditional feature selection approaches and perform significantly better on datasets that contain a large number of features.

Yuan et al. [8] propose a two-phase feature selection approach based on GA. The first phase is a filter approach that uses GA to search for optimal subsets of features. An inconsistency criterion is used to evaluate the fitness of solutions, removing irrelevant features during the first phase. The second phase is a wrapper approach that uses a feedforward neural network in which the input nodes are features obtained from the first phase. This phase aims to remove redundant features from the feature subset. However, by splitting the removal of irrelevant features and redundant features into two separate phases, the algorithm fails to consider feature interactions. This happens because features that are considered irrelevant by themselves in the first phase are removed. However, these features may become useful in combination with other features [2]. Furthermore, the second phase can be computationally expensive if the number of features selected is significantly smaller than the number of features obtained from the first phase. This is because each feature represents an input node to the neural network and when a single feature is removed, the entire neural network has to be retrained with the feature's associated node removed.

Neshatian and Zhang [9] propose a wrapper GP-based feature selection approach that uses a bit-mask representation for feature subsets and a set of feature subset operators that are used as primitive functions. During the evolutionary process, GP combines the feature subsets and operators together to discover sets of features. Through this process, the algorithm discovers a number of relevant features that should be included in the program tree. It also discovers a number of redundant and irrelevant features that should be excluded in the final feature subset. In order to evaluate the performance of the evolved feature subsets, a variation of Naïve Bayes (NB) is used in the fitness function. The performance of the algorithm is evaluated on a highly unbalanced face detection problem and achieves a significant reduction in dimensionality and processing time compared to using all features.

Ming [10] proposes a hybrid feature selection approach that combines ACO and rough set theory. The hybrid algorithm starts with the features included in the core of the rough set. The search is then guided by ACO using forward selection to search for optimal feature subsets. The proposed algorithm discovers feature subsets that contain a smaller number of features than the C4.5 algorithm and achieve higher classification accuracy. However, the algorithm is not compared against any commonly used feature selection algorithms.

2.4.3 PSO-Based Feature Selection Approaches

Many algorithms have been used for feature selection, but one of the most powerful algorithms is PSO. Compared with GA and GP, PSO is easier to implement, has fewer parameters, is computationally less expensive and can converge more quickly [16]. BPSO is often chosen over continuous PSO for feature selection as it is designed for discrete problems such as feature selection where a feature is either included (value of 1) or not (value of 0).

The inertia weight used in PSO controls how much of the velocity from the last iteration is carried over to the current iteration. Inertia weight is an important parameter in BPSO as it can potentially improve the performance by balancing the local search and global search abilities of the swarm during the BPSO process. Yang et al. [12] propose two BPSO based wrapper feature selection approaches that change the inertia weight during the BPSO process. The results presented show that the two new algorithms can outperform SFS, SFFS, sequential GA and different hybrid GAs.

The *gbest* represents the global knowledge of the swarm and is used to lead particles towards optimal solutions in the search space. In standard PSO, *gbest* is updated when a new solution is better than the current *gbest*. Chuang et al. [13] propose a new BPSO based feature selection algorithm that resets *gbest* if it maintains the same value after several iterations. Experiments using cancer-related gene expression datasets show that the proposed

BPSO approach outperforms the approach proposed in [14] in most cases. Chuang et al. [15] extends this idea by introducing the so-called catfish particles into BPSO for feature selection after consecutive iterations in which the *gbest* has not improved. These catfish particles replace 10% of particles within the swarm that have the worst fitness values. This helps BPSO avoid premature convergence by guiding particles trapped in local optima towards more promising regions of the search space. The experimental results show that the proposed algorithm outperforms sequential GA, SFS, SFFS and other methods.

Wang et al. [11] redefine the velocity in BPSO as the number of elements that should be changed in the position of a particle. The fitness function used is given in terms of both classification accuracy and feature subset size, with a 90% weighting on classification accuracy and a 10% weighting on the feature subset size. The inertia weight starts at a large value of 1.4 as to provide greater exploration of the search space during the start of BPSO. With every iteration, the inertia weight decreases, ending at 0.4 on the last iteration. This is done to achieve a finer grained search during the end of the BPSO process. The performance of the proposed approach is shown to be computationally less expensive than that of GA in a filter feature selection model based on rough sets theories in terms of both memory and running time. However, most of the running time is consumed through computation of the rough sets, which is a drawback of using rough sets in feature selection problems.

2.4.4 Multi-Objective Evolutionary Algorithms for Feature Selection

Hamdani et al. [20] apply NSGA-II to solve feature selection as a multi-objective problem in which the two competing objectives are the minimisation of both the number of features and the classification error evaluated by the 1-Nearest Neighbour algorithm [20]. Although the authors do not compare NSGA-II for feature selection against any other feature selection approaches, they analyse the convergence of NSGA-II and the diversity of evolved solutions, both of which are important aspects of multi-objective algorithms. NSGA-II is found to converge quickly to the *Pareto front* of non-dominated solutions, whilst the diversity of solutions is continuously improved throughout the evolutionary process.

Xue et al. [41] introduce a binary multi-objective PSO framework (NSBPSO) for filter feature selection, based on the idea of non-dominated sorting in NSGA-II. They develop two new multi-objective BPSO-based algorithms for filter feature selection, each of which are tested with two different statistical evaluation measures (mutual information and entropy). In most cases, the BPSO-based multi-objective feature selection approaches are found to outperform a single-objective filter feature selection algorithm based on BPSO in terms of both classification performance and the number of features selected. In most cases, the algorithms developed also outperform PSORSFS [11] which is a filter feature selection based on BPSO and rough sets.

2.5 Summary

Classification is an important task in machine learning that aims to accurately classify instances with their correct class labels. In order to learn how to classify instances correctly, classification algorithms must first be trained on instances which are representatives of their associated class labels. However, in many cases, instances are represented by a large number of features. Many of these features are irrelevant or redundant which can harm the classification performance. Feature selection is a task that can remove a number of irrelevant and redundant features by selecting a subset of useful features. However, feature selection is complex due to the large search space and unpredictable feature interaction. As

EC techniques provide good global search abilities, they are commonly used for feature selection. PSO is a recent EC technique that has been successfully used for feature selection and has been found to outperform traditional feature selection techniques as well as other EC techniques. Feature selection has traditionally been solved as a single-objective problem in which the objective is to minimise the classification error rate. However, feature selection also has another objective: to minimise the number of features selected. Consequently, feature selection can be solved as a multi-objective problem. However, this is rarely investigated, especially in combination with EC techniques such as PSO.

There are three main limitations in the existing feature selection approaches which are provided in summary below.

1. There is little research investigating the benefits of using statistical clustering information within the feature selection process. By clustering features into a set of feature clusters as a statistical preprocessing step, the search space and time taken to select an optimal feature subset during the feature selection process could potentially be reduced.

2. Feature interaction within a set of features is complex and hard to model. Better techniques can be developed to find these interactions more effectively.

3. There are very few papers on multi-objective PSO for feature selection, which aims to reduce both classification error rate and dataset dimensionality. These two objectives are usually conflicting with each other, resulting in a trade off between them. Furthermore, most methods do not provide a set of solutions to choose from. This set would allow the user to control the trade-offs between classification error rate and dataset dimensionality.

Objectives 1, 2, and 3 are aimed towards addressing these limitations, through development of new PSO-based feature selection approaches that use statistical clustering information provided by a set of feature clusters.

Chapter 3

A Constrained PSO-based Approach to Feature Selection

This chapter addresses Objective 1, which is to develop a single-objective PSO-based approach that selects a subset of features from a set of feature clusters with the constraint of selecting a single feature from each cluster. As each feature cluster groups a set of similar features, selecting a single feature from each cluster is expected to provide enough information about all of the features in the same feature cluster. This is expected to allow a PSO-based feature selection approach to evolve feature subsets that are significantly reduced in size and that provide competitive classification accuracy compared to that of using all features. In order to achieve Objective 1, the statistical information gained from the preprocessing step of feature clustering will be used to select an optimal subset of relevant features from a large feature set. The clustering algorithm statistically assigns all features to a set of feature clusters that will be used as input to a newly developed PSO-based feature selection approach. The particles used within the PSO-based feature selection approach will then automatically evolve optimal feature subsets that contain a single relevant feature from each feature cluster. This clustering algorithm is out of the scope of this project and hence, will not be discussed in detail. Objective 1 is broken down into two key sub-objectives:

- Sub-objective A is to develop a PSO-based feature selection approach that can automatically select an optimal feature from each feature cluster and evolve a feature subset that provides competitive classification accuracy compared to using all features.

- Sub-objective B is to investigate whether the introduction of a greater amount of stochasticity can improve the classification accuracy of the feature subset evolved by the PSO-based feature selection approach.

BPSO is chosen over standard PSO as the framework for the development of the algorithms in Objective 1 as it uses a binary position vector for particles which naturally suits feature selection problems [36], where a feature is either included in the solution (value of 1) or not (value of 0). Furthermore, BPSO has been successfully used in many feature selection problems [25, 42, 41, 12, 13, 14, 15, 11, 43]. Due to a lack of research in the use of statistical grouping with PSO, there exists no current encoding scheme in BPSO that can represent particle solutions over a set of feature clusters. Therefore, a constraint is introduced to the BPSO representation in which only a single feature is allowed to be selected from each feature cluster. This constraint is upheld in the particle position update process within the newly developed PSO-based feature selection approaches.

3.1 Development of PSO-based Feature Selection Approaches

With standard BPSO for feature selection, a particle's position consists of a finite set of features. The probability of selecting a feature is obtained through squashing the feature's associated velocity value through the sigmoid function shown in Equation (2.1). As this value is not normalised, each feature can potentially have a very low or very high chance of being selected and anywhere in between. This allows particle positions to contain any number of features in the dataset. However, Objective 1 constrains each particle position to consist of features selected from feature clusters in which only a single feature is allowed to to be selected from each cluster. This is a challenging task as a cluster may contain many useful features that are competing for selection. On the contrary, there may also exist clusters that contain only useless features. However, the discovery of a good feature subset within this constraint is important as this feature subset determines the classification accuracy of the learned classifiers that use the feature subset. In order to let each particle in BPSO select a single feature from each feature cluster, four novel particle position update (PPU) algorithms are developed for the particle position update step in BPSO. The way in which these PPU algorithms fit within the evolutionary process of BPSO is outlined with the pseudocode in Algorithm 1. These four algorithms are expected to evolve feature subsets through BPSO that can either maintain or improve classification accuracy compared to that of using all features. Detailed PPU design and architectural implementation decisions can be viewed in Section 3 of Appendix A.

Algorithm 1: Pseudocode of BPSO using a set of feature clusters and a PPU algorithm

```
Input : Training set, Test set, $FC$: a set of feature clusters, where each feature cluster is a set of feature indices
1  begin
2      randomly initialise the position and velocity of each particle in the swarm (Swarm);
3      while Maximum Iterations has not been met do
4          Evaluate the fitness of each particle according to its classification performance;
5          for i=1 to Swarm Size do
6              update pbest and gbest of particle $i$ ;
7          end
8          for i=1 to Swarm Size do
9              update the velocity $v_i$ of particle $i$ according to Equation (2.2);
10             update the position $p_i$ of particle $i$ according to the selected PPU algorithm;
11         end
12         calculate the training and testing classification accuracy of the evolved feature subset return the position of
           gbest, the training and testing classification accuracies.
13     end
14 end
```

3.1.1 Maximum Probability PPU

The Maximum Probability PPU (MPPPU) algorithm is proposed to take advantage of the swarm intelligence that PSO provides, guiding particles towards promising regions of the search space. The MPPPU algorithm uses the velocity vector of a particle to update its position through selection of features with the highest velocities. As a larger velocity value v_i represents a greater probability for selection of feature i in BPSO, the MPPPU algorithm guides particles within the swarm towards promising features in the search space. Algorithm 2 can be used to show the pseudo-code of the MPPPU algorithm by replacing line 7 with the following line.

$$\text{numCompetitors} \leftarrow \text{numFeatures} \tag{3.1}$$

The MPPPU algorithm starts by erasing the given particle's position. Following this, the algorithm iterates over each feature cluster and discovers the feature with the maximum probability (highest velocity) within the feature cluster and includes it in the particle's position. This algorithm allows BPSO to select a small number of features, where each feature has the maximum probability within its associated feature cluster.

The MPPPU algorithm selects features based entirely upon the probability of each feature. This results in particles quickly swarming to *gbest* with minimal needed computing time for convergence. However, a quick progression to this optimum skips over many good solutions in between the particle's current position and *gbest*. Furthermore, due to a lack of bias and randomness, the swarm quickly loses diversity within only a small number of iterations with all particles swarming to *gbest*. This results in premature convergence with other promising regions of the search space going unexplored.

3.1.2 Maximum Probability PPU with Tournament Feature Selection

Due to a lack of stochasticity when selecting features in the MPPPU algorithm, particles quickly swarm to *gbest* which can result in promising regions of the search space going unexplored. To resolve this problem, the maximum probability PPU with tournament feature selection (MPTSPPU) introduces a tournament selection over each feature cluster that selects a random subgroup of features for consideration of selection within each cluster. This introduces more stochasticity to the evolutionary process, counteracting the loss of swarm diversity that leads to premature convergence in the MPPPU algorithm. The pseudo-code of the MPTSPPU algorithm is shown in Algorithm 2.

Algorithm 2: Pseudocode of MPTSPPU

```
Input : $Particle_i$: the $ith$ Particle; $FC$: a set of feature clusters, where each feature cluster is a set of feature indices
begin
    set all elements in the position vector of $Particle_i$ equal to 0;
    foreach cluster in $FC$ do
        featureIndex ← -1 ;
        maxVelocity ← $-\infty$ ;
        numFeatures ← $|FC[\text{ cluster }]|$ ;
        numCompetitors ← $\lceil \frac{\text{numFeatures}}{2} \rceil$ ;      /* Considers 50% of features in each feature cluster */
        featureIndices ← get numCompetitors random feature indices from the interval $[0, \text{numFeatures}]$ ;
        foreach feature in featureIndices do
            featureVelocity ← velocity of feature at index $FC$[ cluster ][ feature ] ;
            if featureVelocity > maxVelocity then
                maxVelocity ← featureVelocity
                featureIndex ← $FC$[ cluster ][ feature ] ;
            end
        end
        $Particle_i$.position[ featureIndex ] = 1 ; /* Include the selected feature in the position of $Particle_i$ */
    end
end
```

The MPTSPPU algorithm first erases the given particle's position. For each feature cluster, a tournament selection is used to select a feature. The tournament selection starts by selecting a random subgroup of features for consideration within the given cluster. The feature with the maximum probability (highest velocity) from this subgroup is then included in the particle's position.

It was discovered early in testing that a static number of competitors lead to a significant amount of deviation in the classification accuracies of the learned classifiers using the discovered feature subsets. This is because the number of competing features within a given feature cluster determines the stochasticity of the MPTSPPU algorithm and a static number of competitors leads to differing amounts of stochasticity for feature clusters differing in size. For example, if a number of random competitors are chosen equal to the number of features within each cluster, the MPTSPPU algorithm acts like the MPPPU algorithm. i.e. selecting the feature with the highest velocity in each cluster. On the other hand, if a single feature is chosen as the subgroup of features for consideration from each cluster, the MPTSPPU algorithm essentially ignores feature velocities. i.e. selecting a random feature from each cluster. For this reason, the number of features considered in the tournament selection is chosen as 50% of the size of the given feature cluster. If this is not a discrete value, the number of competitors is rounded up to the nearest discrete value. As an example, a feature cluster with 5 features result in $\lceil \frac{5}{2} \rceil = 3$ competitors. The consideration of 50% of features within the given feature cluster maintains consistent stochasticity regardless of the feature cluster size. This allows the swarm diversity to be maintained and avoids the problem of premature convergence in the MPPPU algorithm

3.1.3 Roulette Wheel Feature Selection PPU and Incremental Maximum Probability PPU

The Roulette Wheel Feature Selection (RWSPPU) algorithm introduces a roulette wheel selection to select a feature from each feature cluster based on its normalised probability within its associated cluster. This gives features that are unlikely to be selected a small chance of selection, allowing the exploration of regions in the search space that may go ignored when using the MPPPU and MPTSPPU algorithms. The Incremental Maximum Probability PPU (IMPPPU) algorithm combines the advantages of the MPPPU, MPTSPPU and RWSPPU algorithms with a linearly decreasing stochastic approach that selects a feature from a feature cluster either randomly or with the highest probability (highest velocity). The detailed design and implementation of RWSPPU and IMPPPU can be viewed in Sections 1 and 2 of Appendix A.

3.2 Experimental Design

In the experiments, the classification performance of the four newly proposed PPU algorithms (MPPPU, MPTSPPU, RWSPPU and IMPPPU) are compared against the classification performance of using all features and a baseline Greedy forward selection algorithm (GFS). The GFS algorithm is designed by Bing Xue that uses a set of feature clusters to select a single feature from each cluster. The GFS algorithm first starts with an empty set of features. The classification performance of each feature is then evaluated through the chosen evaluation method. Following this, GFS selects a single feature with the highest evaluated classification performance from any feature cluster. All features contained within the feature cluster associated with the selected feature are then removed. Following this, the algorithm sequentially selects the feature that has the highest evaluated classification performance combined with the set of selected features so far from any remaining feature cluster and then removes all other features from its associated cluster.

Eight benchmark datasets chosen from the UCI machine learning repository [44] are used in the experiments, which can be seen in Table 3.1. These datasets are selected as they exhibit a range of different numbers of features, classes and instances. For each dataset, the features are clustered into a number of feature clusters which are used as input to each of the four new feature selection approaches. The instances in each dataset are split randomly into a training set and a test set with each set containing 70% and 30% of the instances, respectively. In order to examine the performance of the proposed algorithms, K-Nearest Neighbour (K-NN) is used as the classification algorithm with K=5 (5-NN). The classification accuracy of each solution is evaluated by 5NN implemented in the Java machine learning library (Java-ML) [45].

The parameters of BPSO are set as follows: inertia weight w = 0.7298, acceleration constants $c_1 = c_2$ = 1.49618, minimum velocity v_{min} = -6.0, maximum velocity v_{max} = 6.0, population size s = 30, Iterations i = 100. The fully connected topology is used in BPSO. These values are chosen based on the common settings in the literature [46]. As the GFS algorithm is deterministic, only 1 run is needed to establish the performance of the algorithm. BPSO is a stochastic approach that needs a greater number of runs to establish an accurate measure of performance. Therefore, each PPU algorithm is run 50 independent times within BPSO on each dataset.

Table 3.1: Datasets

Dataset	#Features	#Feature clusters	# Classes	# Instances
Australian Credit Approval	14	7	2	690
Vehicle	18	5	4	846
German	24	10	2	1000
World Breast Cancer Diagnostic (WBCD)	30	8	2	569
Lung Cancer	56	7	3	32
Sonar	60	10	2	208
Musk1	166	12	2	476
Multiple Features	649	4	10	2000

3.3 Experimental Results

Experimental results of the proposed PPU algorithms on the eight datasets are shown in Table 3.2. In this table, “All” means that all of the available features are used for classification. “GFS” is the greedy forward selection algorithm proposed by Bing Xue. “MPPPU”, “MPTSPPU”, “RWSPPU” and “IMPPPU” are the four newly proposed PPU algorithms. “NumFeatures” represents the number of features selected by each technique which is the same as the number of clusters. All of the PPU algorithms and the GFS algorithm result in solutions that contain the same number of features as each algorithm must select a single feature from each feature cluster. “Ave-Train-Acc, Ave-Test-Acc, Best-Train-Acc and Best-Test-Acc” represent the average and best training and test accuracies of the feature subsets selected by each approach in the 50 runs. ”Std-Train-Acc” and ”Std-Test-Acc” represent the standard deviation of the 50 training and testing accuracies, respectively. “T1” represents the results of a T-Test with a 95% confidence interval between the *testing* classification accuracies achieved by each of the four new algorithms (GFS, MPPPU, MPTSPPU, RWSPPU) and the classification accuracy when using all features. ”T2” represents the results of a T-Test with a 95% confidence interval between the *testing* classification accuracies achieved by each of the four PPU algorithms and the classification accuracy achieved by the GFS algorithm.

3.3.1 Results of MPPPU

The results in Table 2 show that in five out of the eight datasets, the feature subsets selected by the MPPPU algorithm result in a higher average testing accuracy than using all features. Furthermore, all the evolved feature subsets contain a reduced number of features. For example, the MPPPU algorithm selected 7 out of a total of 14 features from the Australian Credit Approval dataset and improved testing accuracy by 3.38% over using all features. In one of the remaining datasets (Musk1), the MPPPU algorithm selected 12 features and achieved higher training accuracy than using all of the 166 features. Although, the average testing accuracy was lower than using all of the 166 features, its best test accuracy is better than using all features.

Compared to the GFS algorithm, the MPPPU algorithm discovered feature subsets with higher average testing accuracy in four out of the eight datasets. For example, in the WBCD

Table 3.2: Experimental Results

Dataset	Method	Num-features	Ave-Train-Acc (Best-Train-Acc)	Std-Train-Acc	Ave-Test-Acc (Best-Test-Acc)	Std-Test-Acc	T1	T2
Australian Credit Approval	All	14	67.08		70.05			
	GFS	7	73.71		70.53		+	
	MPPPU	7	74.92 (74.92)	0.00	73.43 (73.43)	0.00	+	+
	MPTSPPU	7	74.92 (74.92)	0.00	73.43 (73.43)	0.00	+	+
	RWSPPU	7	74.92 (74.92)	0.00	73.43 (73.43)	0.00	+	+
	IMPPPU	7	74.92 (74.92)	1.11E-16	73.43 (73.43)	2.22E-16	+	+
Vehicle	All	18	81.5		83.86			
	GFS	5	82.85		84.84		+	
	MPPPU	5	83.17 (83.19)	3.30E-4	84.41 (84.84)	8.13E-3	+	-
	MPTSPPU	5	83.19 (83.19)	0.00	84.84 (84.84)	1.00E-15	+	=
	RWSPPU	5	83.19 (83.19)	0.00	84.84 (84.84)	1.00E-15	+	=
	IMPPPU	5	83.19 (83.19)	3.33E-16	84.84 (84.84)	5.55E-16	+	=
German	All	24	70.00		68.33			
	GFS	10	71.29		68.67		+	
	MPPPU	10	75.87 (76.14)	3.93E-3	69.67 (72.00)	4.67E-3	+	+
	MPTSPPU	10	76.14 (76.14)	0.00	69.67 (69.67)	1.00E-15	+	+
	RWSPPU	10	76.14 (76.14)	0.00	69.67 (69.67)	1.00E-15	+	+
	IMPPPU	10	76.14 (76.14)	1.11E-16	69.67 (69.67)	5.55E-16	+	+
WBCD	All	30	93.22		92.98			
	GFS	8	92.71		89.47		-	
	MPPPU	8	95.03 (95.21)	1.04E-3	93.91 (94.74)	5.74E-3	+	+
	MPTSPPU	8	95.21 (95.21)	1.00E-15	92.98 (92.98)	0.00	=	+
	RWSPPU	8	95.20 (95.21)	3.32E-4	93.01 (94.15)	1.64E-3	=	+
	IMPPPU	8	95.21 (95.21)	6.66E-16	92.98 (92.98)	1.11E-16	=	+
Lung Cancer	All	56	81.82		70.00			
	GFS	7	90.91		90.00		+	
	MPPPU	7	98.90 (100.00)	1.90E-2	80.20 (90.00)	4.24E-2	+	-
	MPTSPPU	7	100 (100.00)	0.00	80.80 (90.00)	3.37E-2	+	-
	RWSPPU	7	99.93 (100.00)	4.67E-3	80.00 (90.00)	3.46E-2	+	-
	IMPPPU	7	100 (100)	0E0	79.4 (90)	3.69E-2	+	-
Sonar	All	60	74.48		76.19			
	GFS	10	77.93		76.19		=	
	MPPPU	10	81.95 (84.86)	1.18E-2	75.65 (82.54)	3.22E-2	=	=
	MPTSPPU	10	83.07 (84.86)	6.79E-3	76.29 (85.71)	3.37E-2	=	=
	RWSPPU	10	83.04 (84.86)	7.31E-3	76.44 (82.54)	3.12E-2	=	=
	IMPPPU	10	83.18 (84.86)	70.6E-4	76.6 (82.54)	2.97E-2	=	=
Musk1	All	166	81.68		83.92			
	GFS	12	83.48		79.02		-	
	MPPPU	12	87.48 (89.18)	8.82E-3	80.8 (86.01)	2.33E-2	-	+
	MPTSPPU	12	88.37 (90.68)	7.46E-3	81.62 (87.41)	2.66E-2	-	+
	RWSPPU	12	87.45 (89.47)	6.49E-3	81.38 (88.11)	2.45E-2	-	+
	IMPPPU	12	89.76 (91.28)	84.8E-4	82.06 (87.41)	2.13E-2	-	+
Multiple Features	All	649	99.01		98.60			
	GFS	4	95.63		95.30		-	
	MPPPU	4	93.65 (95.77)	9.33E-3	93.48 (95.43)	9.61E-3	-	-
	MPTSPPU	4	95.47 (95.97)	3.22E-3	95.12 (96.17)	4.66E-3	-	-
	RWSPPU	4	94.00 (95.16)	5.87E-3	93.82 (95.00)	5.99E-3	-	-
	IMPPPU	4	95.54 (95.76)	27.8E-4	95.15 (95.47)	33.4E-4	-	-

dataset, the MPPPU algorithm achieved an average testing accuracy of 93.91%, which is 4.44% higher than that of the GFS algorithm. In the Lung Cancer dataset, the MPPPU algorithm discovered a higher average training accuracy (98.90%) than GFS's training accuracy (90.91%). However, the feature subset evolved by the GFS algorithm resulted in a higher test accuracy (90.00%) than MPPPU's average test accuracy of 80.2%, but achieves the same best accuracy.

The results suggest that the MPPPU algorithm can successfully evolve feature subsets through selection of a single feature from each feature cluster within BPSO. The evolved feature subsets achieved higher training accuracy than using all features and the GFS algorithm in almost all datasets (except for the Multiple Features dataset). This results in maintaining or improving the testing accuracy over that of using all features and the GFS algorithm in over half of the datasets.

3.3.2 Results of MPTSPPU, RWSPPU and IMPPPU

According to the results shown in Table 2, The MPTSPPU, RWSPPU and IMPPPU algorithms discovered feature subsets with higher average testing accuracies than using all features in four out of the eight datasets. In two out of the eight datasets (WBCD and Sonar), the three algorithms achieved competitive average testing accuracy compared to that of all features with a much smaller number of features in both datasets, but their best test accuracies are the same or better than using all features.

In comparison to the GFS algorithm, the MPTSPPU, RWSPPU and IMPPPU algorithms achieved higher average test accuracies in four out of the eight datasets. The algorithms achieved higher training accuracies, but similar testing accuracies for 2 of the remaining datasets (Vehicle and Sonar). Similarly, the three algorithms achieved a higher training accuracy than the GFS algorithm on the Lung Cancer dataset, but a lower average test accuracy and the same best accuracy.

Compared to the MPPPU algorithm, the MPTSPPU algorithm achieved a higher average testing accuracy in five out of the eight datasets, whilst the RWSPPU and IMPPPU algorithms achieved a higher average testing accuracy in four out of the eight datasets. In two other datasets (Australian Credit Approval and German), all algorithms achieved the same average testing accuracy. On another dataset (WBCD), the MPTSPPU, RWSPPU and IMPPPU algorithms achieved an average training accuracy of 95.21%, 95.20% and 95.21%, respectively. This is greater than the average training accuracy of the MPPPU algorithm which is 95.03%. However, the MPTSPPU, RWSPPU and IMPPPU algorithms achieved an average testing accuracy of 92.98%, 93.01% and 92.98%, respectively, which is lower than that of the MPPPU's average testing accuracy of 93.91%. These results suggest that the MPTSPPU, RWSPPU and IMPPPU algorithms have the potential to improve the classification performance of the MPPPU algorithm through a greater amount of stochasticity.

The performance of the MPTSPPU, RWSPPU and IMPPPU algorithms are similar on most datasets. However, on three out of the four datasets that contain large numbers of features (Lung Cancer, Musk1 and Multiple Features), the MPTSPPU algorithm achieved higher training and testing accuracies than the RWSPPU algorithm. For example, on the Multiple Features dataset, the MPTSPPU achieved an average testing accuracy of 95.12% which is 1.30% greater than that of the RWSPPU algorithm. Furthermore, the IMPPPU algorithm outperformed the RWSPPU and MPTSPPU algorithms on the 3 datasets that contain the largest number of features (Sonar, Musk1 and Multiple Features) in terms of both average training and testing accuracy. For example, on the Musk1 dataset, the IMPPPU algorithm achieves an average training accuracy that is 2.31% greater than the RWSPPU algorithm and 1.39% greater than the MPTSPPU algorithm. It achieved an average testing accuracy that is 0.68% greater than the RWSPPU algorithm and 0.44% greater than the MPTSPPU algorithm. These results suggest that the MPTSPPU and IMPPPU algorithms have a better search ability than the RWSPPU algorithm on datasets with a greater number of features. Furthermore, the results suggest that the IMPPPU algorithm can improve classification performance over the MPTSPPU algorithm.

The results show that the MPTSPPU, RWSPPU and IMPPPU algorithms can use a set of feature clusters to evolve feature subsets with a reduced number of features within BPSO that maintain or improve the classification accuracy. The results suggest that a greater amount of stochasticity within these algorithms provide a better search strategy that maintains greater swarm diversity. This allows the MPTSPPU, RWSPPU and IMPPPU algorithms to achieve higher classification accuracy than the MPPPU algorithm in most cases.

3.3.3 Further Analysis

The two datasets in which the GFS and PPU algorithms both achieved lower test accuracies (Musk1 and Multiple Features) contain a significantly large number of features, whilst the number of feature clusters used for feature selection in each dataset are relatively small. This constrains the feature subsets discovered through the GFS and PPU algorithms to contain a small number of features compared to the total number of features within the Musk1 and Multiple Features datasets (12 out of 166 and 4 out of 649 features, respectively). This small percentage of features was found to be incapable of providing enough information to accurately describe the instances in each dataset. Resultantly, the GFS and PPU algorithms failed to discover feature subsets that could achieve competitive classification accuracy compared to using all features on these two datasets. In both datasets, a greater number of feature clusters is needed for the GFS and PPU algorithms to achieve better classification performance than using all features. Within all PPU algorithms, the standard deviation of classification accuracies are found to be small (less than 0.05). This means that all PPU algorithms are stable and can be used to produce consistent results.

3.4 Summary

The goal of Objective 1 was to develop a single-objective PSO-based approach that selects a subset of features from a set of feature clusters with the constraint of selecting a single feature from each cluster. This approach was expected to evolve feature subsets that are significantly reduced in size and that provide competitive classification accuracy compared to that of using all features. The goal of Objective 1 was achieved by developing four new algorithms which are MPPPU, MPTSPPU, RWSPPU and IMPPPU. The performance of each algorithm was compared to one another, the GFS algorithm and to that of all features using the overall classification performance of the evolved feature subsets on eight datasets with varying numbers of features, classes and instances.

The results suggest that selecting a single feature from each feature cluster can effectively reduce the number of features selected in the GFS, MPPPU, MPTSPPU, RWSPPU and IMPPPU algorithms. By doing so, a smaller number of features are used when training classifiers, resulting in a reduction in computing time. It was also observed that in most cases, the classification accuracy is either maintained or improved through the removal of redundant and irrelevant features. Incorporating the search over feature clusters into BPSO with the MPPPU algorithm has resulted in a feature selection approach that uses the swarm intelligence within BPSO to provide an effective global search over a set of feature clusters. The MPPPU algorithm was found to make substantial improvements over the GFS algorithm by achieving higher classification accuracies with the same number of features in most datasets. By introducing a greater amount of stochasticity with the MPTSPPU, RWSPPU and IMPPPU algorithms, the classification performance is improved over the MPPPU algorithm on most datasets with feature subsets of equal size.

Chapter 4

Single-Objective PSO for Feature Selection

As features in a feature cluster are relatively similar, a single feature can be selected as a representative for its associated cluster. However, a number of similar features can be complementary to each other which can potentially be part of the best subset of features [2] and hence, are desired for selection. Therefore, selecting one feature from each cluster in Objective 1 may limit the classification performance of the selected feature subset. Furthermore, some clusters may contain only useless features. In these cases, we want to select no features from a cluster. This chapter explores Objective 2 which aims to address these issues by developing a new PSO-based feature selection approach that allows the selection of zero or multiple features from each feature cluster. This approach is expected to evolve feature subsets that contain complimentary features and can achieve a higher classification accuracy than the approaches developed in Objective 1. Objective 2 is broken down into two key sub-objectives:

- Sub-objective A is to develop a PSO-based feature selection approach that can automatically select zero or more features from each cluster and evolve feature subsets that can improve the classification accuracy over the PSO-based feature selection approaches developed in Objective 1.
- Sub-objective B is to develop a new heuristic that leads particles to evolve feature subsets containing a minimal number of features from each cluster.

4.1 Development of PSO-based Feature Selection Approaches

The PSO-based feature selection approaches developed in Objective 2 use the binary representation for particles, the velocity update equation shown in Equation (2.2) for updating particle velocities, and newly developed PPU algorithms to update particle positions. Unlike Objective 1, multiple features may be selected by a particle. To achieve this, two new PPU algorithms are developed which can select zero or multiple features per cluster: the Multiple Feature PPU (MFPPU) and the Gaussian Multiple Feature PPU (GMFPPU). To achieve Objective 2, three novel feature selection algorithms are developed. Firstly, BPSO using the MFPPU algorithm to update particle positions. Secondly, two new Gaussian BPSO-based feature selection algorithms (GPSO1 and GPSO2) which use the GMFPPU algorithm to update particle positions. Each of the new feature selection approaches are designed to discover complex feature interactions, allowing the evolution of feature subsets that contain complimentary features. These feature subsets are expected to maintain or improve the classification accuracy over the feature selection approaches developed in Objective 1.

4.1.1 Multiple Feature PPU

The MFPPU algorithm uses the statistical information provided by the feature clusters to evolve feature subsets that contain no more than half of the features from each feature cluster. As most features in each cluster are relatively similar, selection of up to half of these features is expected to provide more than enough information about each of the clusters. Algorithm 8 (page 51 of Appendix 2) can be used to show the pseudo-code of the MFPPU algorithm by removing lines 11 - 18 and inserting a statement thereafter that includes at most $\lceil \frac{|cluster|}{2} \rceil$ features with the highest probabilities from the set of sorted features from each cluster in the position of $Particle_i$.

The MFPPU algorithm starts by erasing the given particle's position. It then iterates over each feature cluster in order to select zero or more features from each cluster. In this inner loop, a set is used to hold a subset of features from a given cluster that are considered for selection. This is known as the 'desired feature set'. Within the inner loop, the desired feature set for a given particle and cluster is filled with the features selected from the particle position update step in BPSO, using only the features held within the given feature cluster in Equation (2.3). The probability of selection for each feature is also included alongside the feature in the desired feature set which is calculated using the feature's associated velocity value in the sigmoid function shown in Equation (2.4). After establishing a set of desired features, half of the features from the given cluster ($\lceil \frac{|cluster|}{2} \rceil$) with the highest probabilities are included in the position of the given particle.

In most cases, selecting more than half of the features within a cluster that groups similar features is likely to introduce a number of redundant features, which could potentially harm the classification performance [3, 4]. MFPPU prevents this from happening by disallowing the selection of more than half of the features from each cluster. This allows the MFPPU algorithm to remove a number of redundant features that standard BPSO for feature selection (BPSOFS) would include in particle positions. With this constraint, the MFPPU algorithm is expected to evolve feature subsets that contain a smaller number of features than the feature subsets evolved by BPSOFS, whilst maintaining the classification accuracy.

4.1.2 Gaussian PSO for Feature Selection

The MFPPU algorithm disallows the selection of more than half of the features from each feature cluster, which can help eliminate a number of redundant features. However, this constraint only implicitly uses the statisistical information provided by a set of feature clusters and allows the MFPPU algorithm to select a number of redundant features that belong to the same cluster. Furthermore, maintaining a constant maximum number of features selected regardless of the feature cluster size has negative effects. For example, when selecting features from a cluster containing 2 features, a maximum of $\lceil \frac{1}{2} \rceil = 1$ feature may be chosen. However, these two features may be complimentary, providing a valuable improvement in classification performance when selected together [2]. On the other hand, a large feature cluster used within MFPPU containing 25 features limits the selection to $\lceil \frac{25}{2} \rceil = 13$ features from that cluster. However, the selection of up to 13 features from a single cluster without a penalty allows MFPPU to evolve feature subsets that contain a large number of redundant features that could harm the classification performance.

To solve these issues, two novel BPSO-based Gaussian feature selection approaches (GPSO1 and GPSO2) are developed to explicitly use the statisistical information provided by a set of feature clusters to evolve feature subsets that contain a minimal number of features from each cluster. GPSO1 uses the newly developed Gaussian Multiple Feature PPU (GMFPPU) algorithm in the particle position update step of BPSO, which aims to minimise the number of features selected per cluster. The GMFPPU algorithm is discussed in detail in Section 1 of Appendix B. The pseudo-code of GPSO1 is shown in Algorithm 3.

Algorithm 3: Pseudocode of GPSO1

```
Input : Training set, Test set, FC: a set of feature clusters, where each feature cluster is a set of feature indices
1  begin
2      randomly initialise the position and velocity of each particle in the swarm (Swarm);
3      while Maximum Iterations has not been met do
4          Evaluate the fitness of each particle according to its classification performance;
5          for i=1 to Swarm Size do
6              update pbest and gbest of particle i ;
7          end
8          for i=1 to Swarm Size do
9              update the velocity $v_i$ of particle i according to Equation (2.2);
10             update the position $p_i$ of particle i according to the GMFPPU algorithm shown in Algorithm 8 in
               Section 1 of Appendix B;
11         end
12         calculate the training and testing classification accuracy of the evolved feature subset return the position of
           gbest, the training and testing classification accuracies.
13     end
14 end
```

GPSO2 is a variant of GPSO1 that introduces the Gaussian fitness function, which is a newly developed fitness function that uses the feature cluster information as well as the classification performance. The Gaussian fitness function is discussed in detail in Section 2 of Appendix B. The pseudo-code of GPSO2 can be viewed in Algorithm 3 by replacing line 4, with the evaluation of a each particle's fitness according to the Gaussian fitness function given by Equation (B.4) in Section 2 of Appendix B. GPSO1 and GPSO2 make significant changes to standard BPSO for feature selection with the newly developed GMFPPU algorithm and Gaussian fitness function. The changes made help guide the swarm towards selecting a minimal number of features from each cluster that can achieve high classification performance and are discussed in depth in Appendix B.

4.2 Experimental Design

To examine the performance of the proposed algorithms, a set of experiments have been conducted. The experimental design is the same as in Objective 1 apart from the datasets used. Compared with Objective 1, more datasets that contain a large number of features are used in the experiments. These datasets are chosen as they resemble the types of problems that the feature selection approaches developed in Objective 2 can solve. The datasets are shown in Table 4.1.

Table 4.1: Datasets

Dataset	Number of features	Number of feature clusters	Number of classes	Number of instances
Wine	13	6	3	178
Vehicle	18	6	4	846
Ionosphere	34	11	2	351
Sonar	60	12	2	208
Musk1	166	14	2	476
Arrhythmia	279	15	16	452
Madelon	500	11	2	4400
Multiple Features	649	15	10	2000

4.3 Experimental Results

Experimental results of the proposed feature selection approaches on the eight datasets are shown in Table 4.2. “T1” represents the results of a T-Test with a 95% confidence interval between the testing classification accuracies achieved by each of the four feature selection algorithms (IMPPPU, MFPPU, GPSO1, GPSO2) and the classification accuracy of using all features. ”T2” represents the results of a T-Test between the *testing* classification accuracies achieved by each of the three feature selection approaches developed in Objective 2 (MF-PPU, GPSO1, GPSO2) and the IMPPPU algorithm developed in Objective 1.

Table 4.2: Experimental Results

Dataset	Method	Ave-Num-features	Ave-Train-Acc (Best-Train-Acc)	Std-Train-Acc	Ave-Test-Acc (Best-Test-Acc)	Std-Test-Acc	T1	T2
Wine	All	13	87.63		76.54			
	IMPPPU	6	95.13 (95.13)	7.77E-16	100 (100)	0E0	+	
	MFPPU	5.48	96.71 (96.71)	7.77E-16	95.8 (98.77)	3.08E-2	+	-
	GPSO1	5.4	96.71 (96.71)	7.77E-16	96.59 (98.77)	2.76E-2	+	-
	GPSO2	5.6	96.71 (96.71)	7.77E-16	97.04 (98.77)	2.77E-2	+	-
Vehicle	All	18	88.18		83.86			
	IMPPPU	6	84.45 (84.45)	2.22E-16	83.46 (83.46)	5.55E-16	-	
	MFPPU	8.74	85.96 (86.06)	13.1E-4	84.36 (85.83)	29E-4	+	+
	GPSO1	8.94	86.11 (86.31)	19.7E-4	84.3 (85.24)	62E-4	+	+
	GPSO2	8.56	86.15 (86.31)	18.8E-4	83.93 (85.83)	48.3E-4	=	+
Ionosphere	All	34	85.77		83.81			
	IMPPPU	11	90.16 (90.67)	36.1E-4	90 (92.38)	1.48E-2	+	
	MFPPU	9.56	90.64 (92.7)	78E-4	88.95 (93.33)	1.9E-2	+	-
	GPSO1	7.66	91.59 (92.35)	46.9E-4	89.5 (94.29)	1.68E-2	+	=
	GPSO2	8.6	91.46 (92.73)	49.9E-4	89.77 (92.38)	1.67E-2	+	=
Sonar	All	60	83.45		76.19			
	IMPPPU	12	85.22 (87)	74.2E-4	76.76 (85.71)	3.5E-2	=	
	MFPPU	23.7	86.06 (89.05)	92.9E-4	79.81 (87.3)	3.16E-2	+	+
	GPSO1	17.64	86.74 (89)	93.9E-4	78.19 (87.3)	4.14E-2	+	=
	GPSO2	16.22	86.94 (88.38)	86.3E-4	77.68 (82.54)	3.13E-2	+	=
Musk1	All	166	92.19		83.92			
	IMPPPU	14	86.18 (87.97)	62.8E-4	80.18 (86.01)	2.8E-2	-	
	MFPPU	76.3	90.12 (91.6)	64.2E-4	84.7 (90.91)	2.4E-2	+	+
	GPSO1	39.64	90.02 (91.29)	59.7E-4	84.95 (91.61)	2.73E-2	+	+
	GPSO2	35.82	89.88 (91.29)	64.9E-4	84.2 (88.81)	2.61E-2	=	+
Arrhythmia	All	278	94.79		94.46			
	IMPPPU	15	94.51 (94.69)	9.24E-4	94.51 (95.48)	43.1E-4	=	
	MFPPU	128.66	94.79 (94.98)	8.61E-4	94.58 (95.02)	27.7E-4	+	=
	GPSO1	45.5	94.87 (95.08)	9.14E-4	94.85 (95.7)	34.1E-4	+	+
	GPSO2	26.94	94.39 (94.89)	19.8E-4	94.51 (95.25)	41.2E-4	=	=
Madelon	All	500	83.24		70.9			
	IMPPPU	11	66.72 (73.46)	2.45E-2	66.77 (75.64)	3.28E-2	-	
	MFPPU	238.12	78.34 (79.78)	49E-4	77.35 (79.1)	1.02E-2	+	+
	GPSO1	36.08	85.45 (86.98)	72.6E-4	85.68 (87.82)	1.1E-2	+	+
	GPSO2	31.32	85.32 (87.03)	70.2E-4	85.91 (88.97)	1.14E-2	+	+
Multiple Features	All	649	99.36		98.63			
	IMPPPU	15	97.87 (98.37)	19.3E-4	97.52 (98.07)	27.8E-4	-	
	MFPPU	305.72	99.4 (99.46)	1.78E-4	99.01 (99.17)	7.99E-4	+	+
	GPSO1	91.4	99.38 (99.46)	3.75E-4	99.01 (99.27)	12.5E-4	+	+
	GPSO2	54.76	99.09 (99.4)	22.6E-4	98.75 (99.27)	26E-4	+	+

4.3.1 Results of MFPPU

The results in Table 4.2 show that the MFPPU algorithm evolved feature subsets that achieve higher classification accuracy than using all features on all of the datasets. Along with this, the evolved feature subsets are considerably smaller, containing less than half of the original number of features in all cases.

Compared with IMPPPU, MFPPU achieved a higher classification accuracy in five out of the eight datasets and has similar performance on one other. On the other two datasets, the MFPPU algorithm achieved a higher training accuracy but a lower testing accuracy compared to IMPPPU. As these two datasets contain a small number of features, the result is likely due to overfitting.

The results suggest that disallowing the selection of more than half of the features from each cluster allows MFPPU to discover complex feature interaction which results in feature subsets that are high in classification accuracy and of reduced size. In effect, the MFPPU algorithm achieved better classification performance with a smaller number of features than that of using all features and in most cases, it outperformed the IMPPPU algorithm in terms of the classification accuracy. Of course, the MFPPU algorithm selected more features than the IMPPPU algorithm which is the intention.

4.3.2 Results of GPSO1

Table 4.2 shows that the feature subsets selected by GPSO1 result in significantly higher classification accuracy than using all features on all datasets. Furthermore, on each dataset, GPSO1 selected fewer than half of the original features. In the datasets containing a larger number of features, GPSO1 selected a significantly small number of features, whilst making substantial improvements in classification performance. For example, on the Madelon dataset, GPSO1 selected on average only 47.62% of the original features (238.12 out of 500) and achieved an increase in classification accuracy of 6.44%.

In comparison with IMPPPU, GPSO1 achieved a higher classification performance in five out of the eight datasets and has similar performance on two others. The only dataset that the IMPPPU algorithm outperformed GPSO1 was the Wine dataset which contains a very small number of features. Even though GPSO1 achieved a higher classification accuracy than the IMPPPU algorithm, its testing accuracy was not as high as the IMPPPU algorithm which achieved a testing accuracy of 100%.

On all datasets except for the Vehicle dataset, GPSO1 evolved feature subsets that contain a smaller number of features than the MFPPU algorithm, whilst achieving higher testing accuracies on five out of the eight datasets. In all cases, the reduction in the number of features is significant, whilst in many cases, the classification performance increase is also significant. For example, on the Madelon dataset, GPSO1 selects on average 202.04 fewer features than the MFPPU algorithm, for a total of only 7.22% of the original features (36.08 features out of 500). With these small feature subsets, GPSO1 dramatically improved the classification accuracy over the MFPPU algorithm by 8.33%. In two of the remaining three datasets (Vehicle and Multiple Features), both approaches achieved a similar classification accuracy. However, in the Multiple Features dataset, GPSO1 selects on average less than one third of the features selected by the MFPPU algorithm (91.4 features compared to 305.72).

The results suggest that by introducing a heuristic that helps to minimise the number of features selected per cluster, GPSO1 can significantly reduce the number of redundant features introduced into solutions obtained by the MFPPU algorithm. This reduction in features leads to a much shorter classification time and reduced classifier complexity. Most importantly, the evolved feature subsets achieve higher classification accuracies than the feature subsets evolved by the MFPPU algorithm in most cases.

4.3.3 Results of GPSO2

Table 4.2 shows that the feature subsets selected by GPSO2 result in a higher classification accuracy than using all features on five out of the eight datasets and contain a significantly smaller number of features. On the three remaining datasets, GPSO2 evolves feature subsets that contain a significantly smaller number of features and achieved similar accuracies compared to using all features.

Compared to the IMPPPU algorithm, GPSO2 achieved higher classification accuracy on four out of the eight datasets and similar accuracy on 3 of the remaining datasets. Like GPSO1, GPSO2 achieved a higher training accuracy than the IMPPPU algorithm for the Wine dataset, but cannot achieve a testing accuracy as high as the 100% benchmark set by the IMPPPU algorithm. This is likely due to the small number of features selected by GPSO2 in the Wine dataset.

Like GPSO1, GPSO2 outperforms MFPPU in terms of the number of features selected in almost all datasets. This becomes a significant advantage of GPSO2 when selecting features from datasets containing a large number of features. This significant reduction in features lead to a slight trade off in classification accuracy, which resulted in GPSO2 achieving a slightly lower classification accuracy on half of the datasets. On the other half of the datasets, GPSO2 achieved a higher classification accuracy with much smaller feature subsets.

Compared to GPSO1, GPSO2 further reduced the number of features in most datasets. In particular, it makes a significant reduction of features on the larger datasets. For example, in the Multiple Features dataset, GPSO2 selected on average 36.64 features less than GPSO1. In most cases, the significant reduction in features leads to a small decrease in the classification accuracy achieved by GPSO2 compared to GPSO1. On two other datasets, both GPSO1 and GPSO2 achieve similar accuracies. However, on two of the remaining datasets, GPSO2 achieved a higher classification accuracy than GPSO1.

The results suggest that the addition of the Gaussian fitness function has helped guide the swarm in GPSO2 towards the selection of a minimal number of features per cluster. On three of the largest datasets, GPSO2 selects less than 10% of the total number of features which allows a shorter classification training time and reduced classifier complexity. However, the results suggest that on large datasets, the reduction in features comes at the cost of a small reduction in classification accuracy. Therefore, when choosing between GPSO1 and GPSO2 for feature selection, GPSO2 is preferred if a minimal number of features is desired for efficiency and reduced classifier complexibility, whilst GPSO1 is preferred if having the best classification performance is paramount.

4.3.4 Further Analysis

In order to examine the differences and similarities between the feature subsets evolved by IMPPPU, MFPPU, GPSO1 and GPSO2, we take a typical run from each of the four different feature selection approaches on the Ionosphere dataset and analyse the selected features. In the typical run, the number of features selected by IMPPPU, MFPPU, GPSO1 and GPSO2 are 11, 9, 7 and 8, respectively, out of a total of 34 features in the Ionosphere dataset. With Fi representing the ith feature, IMPPPU selects F1, F2, F5, F7, F8, F11, F15, F16, F18, F23 and F27. The features selected by MFPPU are F2, F5, F8, F11, F13, F16, F19, F27 and F29. The features selected by GPSO1 are F2, F5, F8, F11, F13, F19 and F27. The features selected by GPSO2 are F2, F5, F8, F11, F15, F23, F25 and F27. With these feature subsets, it can be observed that approximately half of the features selected by MFPPU and IMPPU are the same. Most of the features selected by GPSO1 and GPSO2 are also selected by IMPPPU, with a few different features selected by each algorithm. This suggests that by allowing the selection of multiple features per cluster, the algorithms in Objective 2 select some alternative features that can achieve a higher training accuracy than the features selected by IMPPPU. We can see that GPSO1 selectes 2 fewer features than MFPPU and all of the features selected by GPSO1 are contained within the features selected by MFPPU. In this example, GPSO1 has successfully reduced the number of redundant features which has lead to higher training and testing accuracies as can be seen in the results. Out of the eight features selected by GPSO2, seven are contained within the 11 features selected by IMPPPU and 5 are contained within the 7 features selected by GPSO1. This suggests that GPSO2 evolves similar feature subsets to GPSO1, whilst reducing the number of redundant features selected by IMPPPU.

4.4 Summary

The goal of Objective 2 was to develop a single-objective PSO-based feature selection approach that allows the selection of multiple features from each feature cluster. This goal was achieved through the development of the MFPPU algorithm, GPSO1 and GPSO2, which are three new BPSO-based feature selection algorithms that allow the selection of multiple features from each cluster.

The results from the three new feature selection algorithms (MFPPU, GPSO1, GPSO2) suggest that through the selection of multiple features from each cluster, a number of complimentary features can be discovered and included in the evolved solutions throughout the evolutionary process. In effect, the evolved feature subsets achieve a higher classification accuracy with a smaller number of features than using all features. Furthermore, MFPPU, GPSO1 and GPSO2 all achieved higher classification accuracies than the IMPPPU algorithm which represents the best feature selection approach from Objective 1.

By introducing a heuristic that aims to minimise the number of features selected per cluster, GPSO1 dramatically reduces the number of redundant features selected over the MFPPU algorithm on datasets containing a large number of features. This resulted in a significant increase in classification performance, outperforming the MFPPU algorithm in most cases.

GPSO2 further minimises the number of features selected per cluster by introducing the gaussian fitness function which penalises feature subsets that contain more than 1 feature per cluster. The addition of the gaussian fitness function reduced the classification performance by a small amount, providing a clear distinction between the benefits of the GPSO1 and GPSO2 approaches for feature selection. GPSO2 was found to be better at selecting a minimal number of features which reduces the classifier training time and improves the classifier understandability. On the other hand, GPSO1 is better at achieving a very high classification accuracy which is of utmost importance for accurately identifying class labels of instances in classification tasks.

Chapter 5

A Multi-Objective PSO-based Approach to Feature Selection

Feature selection is a multi-objective problem, where the two objectives are to minimise the classification error rate and the dimensionality of the solution (the number of features selected). However, these two objectives often conflict with each other as the classification error rate is likely to increase when fewer features can be used to distinguish the correct class label of a given instance. Multi-objective feature selection can obtain a set of non-dominated feature subsets rather than a single feature subset. An appropriate feature subset can then be selected by a user in terms of their desired trade off between the classification error rate and the dimensionality of the feature subset. This chapter addresses Objective 3, building upon the single-objective algorithms developed in Objective 2 and existing work in the area of multi-objective feature selection with PSO [42, 47, 41] by developing a new multi-objective PSO-based approach that uses statistical clustering information. By doing so, a new area of research is explored in which the statistical information obtained from clustering is used to solve feature selection as a multi-objective task with PSO. Objective 3 is broken down into two key sub-objectives:

- Sub-Objective A is to develop a multi-objective PSO-based approach that uses non-dominated sorting and statistical clustering information to evolve a Pareto front of non-dominated solutions from which the user can select an optimal feature subset.
- Sub-objective B is to develop a new multi-objective PSO-based approach that includes an external leader set, binary tournament selection of a particle's *gbest* and mutation operators in PSO to further minimise the dimensionality and/or classification error rate of the evolved solutions.

5.1 Development of PSO-based Feature Selection Approaches

The PSO-based feature selection approaches developed in Objective 3 use the binary representation for particles, the velocity update equation shown in Equation (2.2) for updating particle velocities and the GMFPPU algorithm to update particle positions using a set of feature clusters. The GMFPPU algorithm is chosen for use within the new multi-objective feature selection approaches as it was successfully used in GPSO1 and GPSO2 to evolve feature subsets that achieve high classification accuracy with a small number of features.

The feature selection approaches developed in Objective 3 must minimise two objectives: the number of features selected and the classification error rate. To achieve this, two new multi-objective PSO-based feature selection approaches are developed that maintain a

Pareto front of non-dominated solutions: Non-dominated Sorting Gaussian Particle Swarm Optimisation for Feature Selection (NSGPSO) and NSGPSO2 which introduces an external leader set, a binary tournament selection of a particle's *gbest* and mutation operators to the NSGPSO feature selection approach. By maintaining a Pareto front of non-dominated solutions, the two new approaches are expected to provide a range of optimal solutions that improve the classification accuracy and/or reduce the number of features over the feature selection approaches developed in Objective 2.

5.1.1 NSGPSO

As standard PSO was originally developed as a single objective algorithm, it cannot be directly applied to the exploration of the Pareto front of feature subsets in terms of classification accuracy and dimensionality. In order to achieve this, NSGPSO is developed as a multi-objective PSO-based feature selection approach that is based on the idea of non-dominated sorting in NSGA-II [17]. Algorithm 4 shows the pseudo-code of NSGPSO.

Algorithm 4: Pseudocode of NSGPSO

```
Input : Training set, Test set, FC: a set of feature clusters, where each feature cluster is a set of feature
        indices
1  begin
2      randomly initialise the position and velocity of each particle in the swarm (Swarm);
3      while Maximum Iterations has not been met do
4          evaluate the two objective values of each particle (classification error rate on the training set and
           number of features);
5          nonDomParticles ← identify the particles in Swarm that have non-dominated solutions;
6          calculate the crowding distance of each particle in nonDomParticles;
7          sortedParticles ← sort the particles in nonDomParticles by their crowding distance ;
8          archive ← initialise an archive to contain all particles in Swarm ;
9          for i=1 to Swarm Size do
10             update the pbest of particle i ;
11             randomly select a gbest for particle i from the first half of sortedParticles (the top 50% of the
               least crowded solutions in the Pareto front) ;
12             update the velocity $v_i$ of particle i according to Equation (2.2);
13             update the position $p_i$ of particle i according to the GMFPPU algorithm shown in
               Algorithm 8 in Section 1 of Appendix B;
14             add particle i to archive
15         end
16         empty the current Swarm for the next iteration ;
17         paretoFronts ← identify the set of non-dominated fronts in archive;
18         $i \leftarrow 1$ ;
19         while $|Swarm| <$ populationSize do
20             if ($|Swarm| + |\text{paretoFronts}_i| <$populationSize) then
21                 add the particles in $|\text{paretoFronts}_i|$ to Swarm
22             else
23                 add the (populationSize $-|Swarm|$) least crowded particles from $|\text{paretoFronts}_i|$ to
                   Swarm
24             end
25             $i \leftarrow i + 1$ ;
26         end
27         calculate the training and testing classification accuracy of the evolved feature subsets in
           $\text{paretoFronts}_1$ (the non-dominated solutions);
28         return the feature subsets in $\text{paretoFronts}_1$, the training and testing classification accuracies.
29     end
30 end
```

NSGPSO starts by randomly initialising the velocities and positions of each particle in the swarm. After this, each iteration of NSGPSO starts by evaluating the two objectives for each particle: the classification error rate on the training set and the number of selected features. NSGPSO uses this information to identify the non-dominated particles in the swarm, forming the set of potential *gbest* solutions for each particle in the swarm. The crowding distance for each non-dominated particle in this set is then calculated with the crowded comparison operator described in NSGA-II [17], which is then used to sort the particles by their crowding distance such that the least crowded solution is at the start of the list. An archive is then initialised to contain all particles in the swarm. This archive is used to store the particles from the previous and current iteration of PSO from which only the non-dominated particles will be included in the swarm for the next iteration. Following this, each particle updates its *pbest*. Then for each particle, a *gbest* is selected randomly from the top 50% of the least crowded particles in the sorted non-dominated solutions. The velocity of each particle is then updated according to the original velocity update Equation (2.2). Using the GMF-PPU algorithm, the position of each particle is then updated and the new particle is added to the archive as to have a chance to be included in the swarm for the next iteration. At this stage, the swarm is emptied and the archive which contains the particles from the current and previous iterations is then used to identify the set of Pareto fronts. This starts from the first Pareto front and iterates over the Pareto fronts until the swarm has reached the predetermined population size. All particles contained in the current Pareto front are added to the swarm if by doing so, the swarm size does not exceed the population size. Otherwise, the particles in the current Pareto front are sorted via their crowding distance and the least crowded particles are included in the swarm such that the number of particles in the new swarm equals the desired population size. The new swarm is then used in the next iteration, continuing the process described above until the maximum number of iterations is met. Lastly, the training and testing classification accuracy is calculated for the non-dominated feature subsets contained in the first Pareto front.

In the original single-objective PSO, the *pbest* of each particle in the swarm is updated if and only if the new position dominates the old position. i.e the fitness of the new position is better than the fitness of the old position. However, in order to increase the diversity of the search within the new multi-objective PSO approach (NSGPSO), the *pbest* updating rule described in [27] is used in which the *pbest* of a particle is updated if either the new position dominates the old position or if both positions are non-dominated with respect to each other.

When determining the crowding distance of solutions in a particular Pareto front, the crowded comparison operator described in NSGA-II [17] is used. This operator evaluates the crowding distance of a solution based on the sum of its normalised distance between each objective value of its two nearest neighbours. The implementation of this operator can be viewed in detail in [17]. The crowded comparison operator is used in NSGPSO to evaluate the goodness of each particle within the same Pareto front such that particle i is evaluated as better than particle j if the crowded comparison operator evaluates particle i to be less crowded than particle j. This information is then used for determining the least crowded particles in the Pareto front to obtain a set of potential *gbest* particles during each iteration of the evolutionary process. By randomly selecting a *gbest* for each particle from the top 50% of these least crowded solutions, half of the most crowded solutions along the non-dominated Pareto front are discarded, whilst the random *gbest* selection from the remaining non-dominated solutions for each particle helps guide the search towards different areas of the non-dominated Pareto front where the non-dominated solutions are sparse. The crowding distance operator is also used in the process of selecting particles to include in the swarm for the next iteration as to determine the least crowded particles to select from a Pareto front that contains too many particles to include as a whole in the swarm. Both of these uses of the crowding comparison operator help to increase the spread of the final non-dominated

Pareto front, which is desired as it gives the user more choice in selecting an appropriate non-dominated feature subset.

For the particle position update step in NSGPSO, the GMFPPU algorithm is used which allows the use of the statistical information provided by the feature clusters to evolve feature subsets containing complimentary features within the Pareto front of non-dominated solutions. Combined with the non-dominated sorting mechanism, *pbest* and *gbest* update rules, NSGPSO is expected to be able to search the solution space and establish a Pareto front of non-dominated feature subsets that achieve high classification accuracy with a relatively small number of features.

5.1.2 NSGPSO2

By sorting the particles in the swarm via their values for each objective (classification error rate and number of features), a set of non-dominated feature subsets is obtained. This allows NSGPSO to be applied to multi-objective feature selection. However, with the direct application of the ideas from NSGA-II to PSO, maintaining an appropriate degree of diversity in the swarm becomes an issue. The lack of swarm diversity in NSGPSO arises from its use of elitism in which the swarm's particles for the next iteration are selected from the current particle' positions and the updated particle' positions such that the non-dominated particles are selected first. This means that any non-dominated particles from the last iteration that are not dominated by any particle in the current iteration are selected to remain in the swarm for the next iteration. As the chances of a particle becoming a non-dominated solution in its next position update is small, only a small number of particles and their positions will be updated throughout the swarm. This issue of low swarm diversity might harm the search in PSO as particles are likely to become stuck in local optima. NSGPSO2 is developed to overcome the lack of swarm diversity found in NSGPSO by introducing an external leader set that holds the non-dominated particles, a binary tournament selection of a particle's gbest and mutation operators which are applied to particles throughout the evolutionary process. Each of these changes aim to diversify the swarm in PSO and hence, aims to improve the search abilities of PSO for multi-objective feature selection using a set of feature clusters. The pseudo-code of NSGPSO2 is shown in Algorithm 5.

NSGPSO2 starts by randomly initialising the velocities and positions of each particle in the swarm. At the start of each iteration of NSGPSO2, the crowding distance for each non-dominated particle in the *LeaderSet* is calculated and used in the binary tournament selection for the selection of a *gbest* for each particle. Following this, for each particle in the swarm, the *pbest* is updated if either the new position dominates the old position or if both positions are non-dominated with respect to each other. The *gbest* is then selected from the *LeaderSet* by using a binary tournament selection based on the crowding distance of the non-dominated particles. Like NSGPSO, the velocity and position of each particle are then updated according to the original velocity update Equation (2.2) and the GMFPPU algorithm, respectively. The uniform and non-uniform mutation operators are then applied to two sets of particles in the swarm with each set containing a third of the swarm's particles. The remaining third of particles in the swarm do not have any mutation operator applied to them. This process is repeated until the maximum number of iterations is met. Lastly, the training and testing classification accuracy is calculated for the final non-dominated feature suets from the *LeaderSet*.

The external leader set is a major change in NSGPSO2 as it uncouples the elitism aspect from the search process, holding and updating up to n of the best non-dominated particles throughout the evolutionary process, where n is the number of particles in the swarm. This is in contrast to NSGPSO which halts the positions of the non-dominated particles in the

Algorithm 5: Pseudocode of NSGPSO2

```
Input : Training set, Test set, FC: a set of feature clusters, where each feature cluster is a set of feature
        indices
1  begin
2      randomly initialise the position and velocity of each particle in the swarm (Swarm);
3      initialise the set of leaders (LeaderSet);
4      while Maximum Iterations has not been met do
5          calculate the crowding distance of each particle in LeaderSet ;
6          swarmSize ← the number of particles in the swarm;
7          for i=1 to swarmSize do
8              update the pbest of particle i ;
9              select a gbest for particle i from LeaderSet by using a binary tournament selection based on
               the crowding distance ;
10             update the velocity $v_i$ of particle i according to Equation (2.2);
11             update the position $p_i$ of particle i according to the GMFPPU algorithm shown in
               Algorithm 8 in Section 1 of Appendix B;
12             if ($i > \frac{2\times \text{swarmSize}}{3}$) then
13                 apply uniform mutation operator to particle i
14             end
15             else if ($i > \frac{\text{swarmSize}}{3}$) then
16                 apply non-uniform mutation operator to particle i
17             end
18         end
19         nonDomParticles ← identify the particles contained in LeaderSet and Swarm that have
           non-dominated solutions;
20         calculate the crowding distance of each particle in nonDomParticles;
21         Update the LeaderSet with up to swarmSize of the least crowded particles in nonDomParticles
22     end
23     calculate the training and testing classification accuracy of the evolved feature subsets in LeaderSet
       (the non-dominated solutions);
24     return the feature subsets in LeaderSet, the training and testing classification accuracies.
25 end
```

swarm until a better solution is obtained, disallowing the movement of these particles in PSO and hence, coupling the elitism aspect with the search process. The particles held in the leader set belong to the non-dominated Pareto front where a binary tournament selection and the crowding distance are used to select the *gbest* for each particle in the swarm. This external leader set eliminates the need for maintaining the best non-dominated particles in the swarm for subsequent iterations as these particles are accessed and updated in the external leader set. Hence, NSGPSO2 provides better search abilities than NSGPSO as it allows each particle to search the solution space over a number of iterations, heuristically guided by its corresponding *pbest* and a chosen *gbest* from the external leader set.

Two mutation operators are introduced to further diversify the search process in PSO. Firstly, the uniform mutation operator is used in which the variability range for each feature is kept constant at $\frac{1}{f}$, where f represents the number of features. Secondly, the non-uniform mutation operator is used in which the variability range for each feature decreases over the evolutionary process linearly from $\frac{1}{f}$ to 0. NSGPSO2 splits the particles in the swarm evenly into three distinct sets in which the first set has no mutation applied to it, the second set has the uniform mutation operator applied to it and the third set has the non-uniform mutation operator applied to it. By doing this, three sets of particles with different search abilities are used to search the solution space. The first set represents the normal particles found in PSO, the second set which contains exploratory particles, providing better global search ability through the uniform chance of mutation, and the the third set which can provide better local search ability as the evolutionary process continues through the decreasing chance of mutation.

As the major changes in the NSGPSO2 algorithm are aimed at increasing the diversity and search ability of the swarm, NSGPSO2 is expected to discover a wider range of non-dominated solutions in the search space, further minimising both the dimensionality and classification error rate of the evolved solutions over NSGPSO.

5.2 Experimental Design

To examine the performance of the proposed algorithms, a set of experiments have been conducted. The experimental design is the same as in Objectives 1 and 2 apart from minor changes to the dataset. Seven of the datasets have been reused from Objective 2, whilst the WBCD dataset has been introduced, replacing the Ionosphere dataset used in Objective 2 as it was found to provide more distinctive results when used in Objective 1. The datasets are shown in Table 5.1.

Table 5.1: Datasets

Dataset	Number of features	Number of feature clusters	Number of classes	Number of instances
Wine	13	6	3	178
Vehicle	18	6	4	846
WBCD	30	6	2	569
Sonar	60	12	2	208
Musk1	166	14	2	476
Arrhythmia	279	15	16	452
Madelon	500	11	2	4400
Multiple Features	649	15	10	2000

5.3 Experimental Results

For each dataset, BPSO and GPSO1 obtain a single solution in each of the 50 independent runs. The multi-objective algorithms, NSGPSO and NSGPSO2 obtain a set of non-dominated solutions in each run. In order to compare these two kinds of results, for each multi-objective algorithm, 50 sets of feature subsets from the 50 independent runs are combined into one union set. From this union set, the non-dominated solutions are identified and presented on the charts to compare with the solutions achieved by the two single objective algorithms, BPSO and GPSO1.

The experimental results for NSGPSO, NSGPSO2, BPSO and GPSO1 are shown in Figure 5.1, where each chart corresponds to one of the datasets used in the experiments. On the top of each chart, the total number of features are shown for the given dataset as well as the classification error rate achieved by using all features. In each chart, the classification error rate is shown along the vertical axis, whilst the number of features selected are shown along the horizontal axis. The legend of each chart shows the color used to represent the solutions for a particular algorithm. "NSGPSO" and "NSGPSO2" show the non-dominated solutions in the Pareto front evolved by NSGPSO and NSGPSO2 over the 50 independent runs, respectively. "GPSO1" and "BPSO" show the 50 solutions evolved by GPSO1 and BPSO respectively. It is to be noted that the same feature subset can be obtained on different runs on a single dataset and these are shown as the same point in the associated chart. This explains why, in most cases there are fewer than 50 data points plotted for GPSO1 and BPSO.

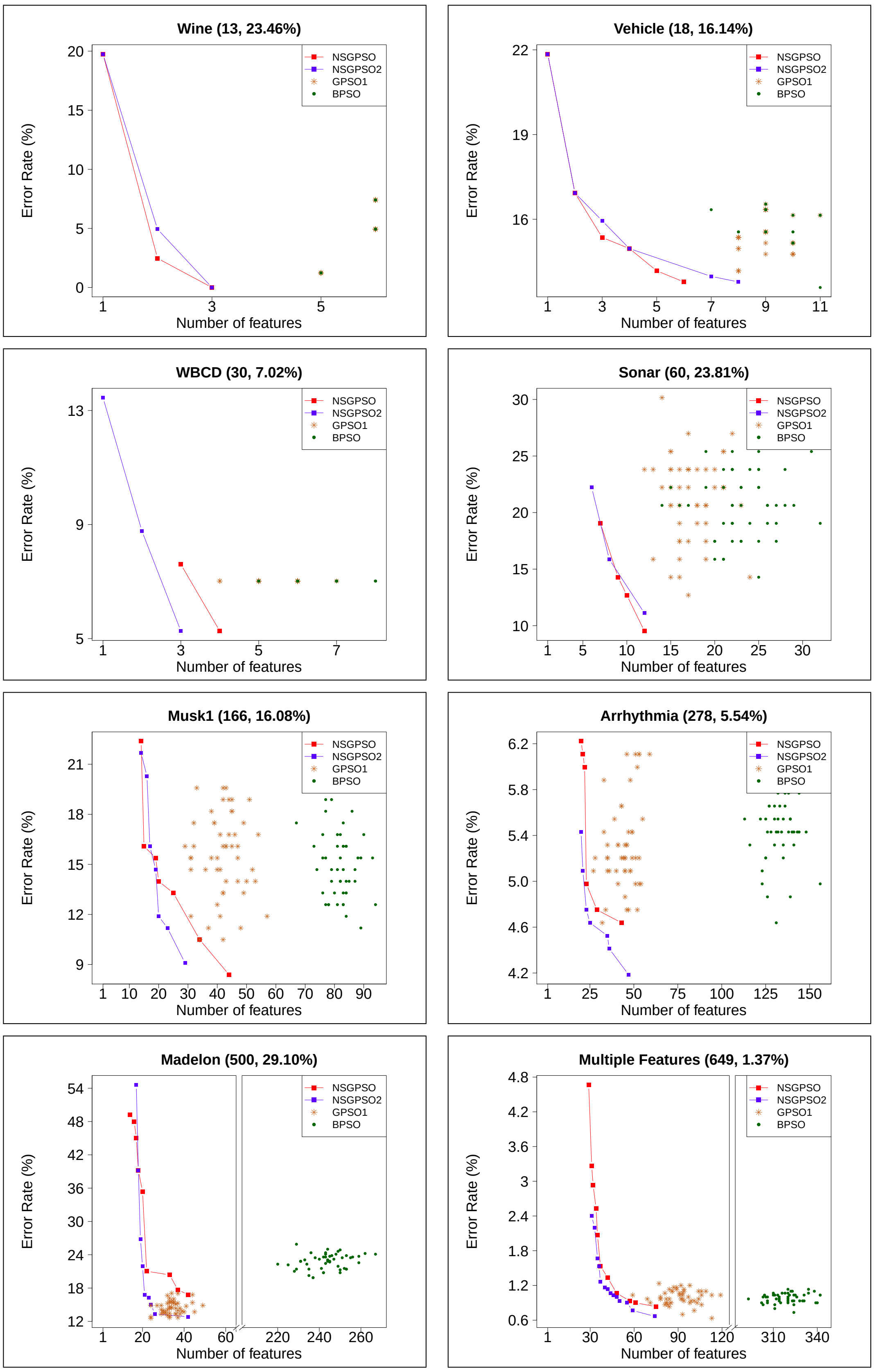


Figure 5.1: Experimental Results of NSGPSO, NSGPSO2, GPSO1 and BPSO

5.3.1 Results of NSGPSO

As can be seen in Figure 5.1, in all datasets, NSGPSO evolves at least two non-dominated solutions that contain a significantly small number of features and achieve a lower classification error rate than using all features. For example, on the Musk1 dataset, a non-dominated solution evolved in NSGPSO contains only 26.5% of the available features (44 from 166) and reduces the classification error rate more than twofold (from 16.08% to 7.69%). In seven out of the eight datasets, there exists at least one non-dominated solution that contains less than 10% of the the available features. In most cases, the non-dominated solution(s) that contain less than 10% of the available features also achieve a lower classification rate than using all features.

Compared to BPSO, the classification error rates of the evolved solutions in BPSO and in the non-dominated Pareto front in NSGPSO are similar in most cases. However, the number of features contained in feature subsets evolved by NSGPSO is significantly smaller. In six out of the eight datasets, NSGPSO includes at least one feature subset that has fewer features and achieves a lower classification error rate than any of the feature subsets evolved by BPSO. In the two remaining datasets, the feature subset containing the lowest classification error rate evolved in BPSO achieved a classification error rate which was less than 0.25% lower than the feature subset containing the lowest classification error rate evolved by NSGSPO. For example, in the MultipleFeatures dataset, BPSO evolved a feature subset that achieved a classification error rate of 0.733% with 324 features, whilst in NSGPSO, the non-dominated solution with the lowest classification error rate contained only 75 features and achieved a classification error rate of 0.833% which is only 0.1% worse than that of the best solution evolved by BPSO.

Compared to GPSO1, most of the non-dominated solutions in NSGPSO contain a smaller number of features in all datasets. In most of the datasets containing a smaller number of features, the classification error rates of GPSO1 and NSGPSO are similar. However, in two of the datasets containing a larger number of features, GPSO1 evolves feature subsets that achieve lower classification error rates than NSGPSO. This is likely due to the swarm diversity in NSGPSO which limits the algorithms search ability on datasets containing a large number of features. In most cases, NSGSPO evolves at least one solution that achieves a lower classification error rate with a smaller number of features than GPSO1.

The results suggest that by using non-dominated sorting and statistical clustering information within a PSO-based approach, NSGPSO can effectively explore the Pareto front and obtain a range of feature subsets that achieve better classification performance with a smaller number of features than BPSO for feature selection. Furthermore, as a multi-objective feature selection approach, NSGPSO can effectively evolve feature subsets that contain a smaller number of features than GPSO1 and is found to be competitive with GPSO1 in terms of classification performance, despite GPSO1 being a single-objective algorithm that maximises the classification performance of its evolved feature subsets.

5.3.2 Results of NSGPSO2

As can be seen in Figure 5.1, the majority of the evolved non-dominated solutions in NSGPSO2 achieve a lower classification error rate than using all features and contain a significantly smaller number of features. For example, on the Madelon dataset, a non-dominated solution evolved by NSGPSO2 contains only 5.2% of the available features (26 from 500) and reduces the classification error rate significantly, by 15.67%. (from 29.10% to 13.33%). In all datasets, there exists at least one non-dominated solution that contains less than 10% of the the available features and in most cases, these non-dominated solution achieve a lower classification rate than using all features. On some datasets, NSGPSO2 evolves feature subsets that contain *significantly* small numbers of features and achieve *significantly* better classification performance than using all features. For example, on the Wine dataset, NSGPSO2 evolves a feature subset containing only 3 out of 13 features that can accurately classify every instance of the dataset with 100% accuracy. This is in comparison to the accuracy of 76.54% when using all 13 features.

Comparing NSGPSO2 to BPSO, the classification error rates of the evolved solutions in BPSO and in the non-dominated Pareto front in NSGPSO2 are similar in most datasets. However, the non-dominated solutions in NSGPSO2 contain a significantly smaller number of features than in BPSO. In most cases, NSGPSO2 includes at least one feature subset that contains a significantly smaller number of features and a lower classification error rate than any of the solutions evolved by BPSO. For example, in the Musk1 dataset, NSGPSO2 contains a feature subset that achieves a classification error rate of 9.09% with only 29 out of the 166 available features. In comparison, in BPSO, the best feature subset in terms of classification error rate is 11.19% with 89 features, whilst the best feature subset in terms of the number of selected features is 67, which achieved a classification error rate of 18.58%.

Compared to GPSO1, most of the non-dominated solutions in NSGPSO2 contain a smaller number of features in all datasets. In most cases, the non-dominated solutions evolved by NSGPSO2 achieve a lower classification error rate than GPSO1 with less features. The two exceptions are in the Madelon and Multiple Features dataset, where GPSO1 evolves a feature subset that contains a slightly lower classification error rate than in NSGPSO2.

In comparison to NSGPSO, the classification error rates and the numbers of features in the evolved solutions of NSGPSO2 are similar to NSGSPO in three of the datasets (Wine, Vehicle, Sonar). However, in most of the remaining datasets, NSGPSO2 evolves a set of non-dominated solutions that achieve a lower classification error rate and a smaller number of features than that of the non-dominated solutions evolved by NSGPSO. For example, for every non-dominated solution evolved by NSGPSO in the Multiple Features dataset, NSGPSO2 evolved a non-dominated solution that achieves a lower classification error rate with the same or smaller number of features than the feature subset evolved by NSGPSO. Furthermore, on most of the datasets containing a larger number of features (Arrhythmia and Multiple Features), NSGPSO2 achieves a greater number of solutions and a better spread than NSGPSO along its Pareto front of non-dominated solutions. This is best seen on the Multiple Features dataset where NSGPSO2 evolves 14 non-dominated feature subsets compared to the 11 non-dominated feature subsets evolved by NSGPSO.

The results suggest that with an external leader set, binary tournament selection of a particle's gbest and mutation operators, the diversity of the swarm in NSGPSO2 is increased which leads to a more efficient and effective search process. In turn, NSGPSO2 outperforms BPSO in all cases and further lowers the classification error rate of its evolved feature subsets over NSGPSO. NSGPSO2 also outperforms GPSO1 in all cases in terms of the number of features selected and in most cases in terms of classification performance. For these reasons, NSGPSO2 is found to be an effective method that can evolve a range of non-dominated feature subsets that achieve high classification performance with a significantly small number of features.

5.4 Summary

The goal of this chapter was to develop a multi-objective PSO-based feature selection approach that can explore the Pareto front of non-dominated feature subsets. This goal was achieved by developing NSGPSO and NSGPSO2, which are two BPSO-based multi-objective feature selection approaches that can explore the Pareto front of non-dominated feature subsets.

By using non-dominated sorting and statistical clustering information together within a multi-objective PSO-based approach, NSGPSO is able to effectively explore the Pareto front, evolving a range of feature subsets that achieve high classification performance with a small number of features. The results show that NSGPSO outperforms standard BPSO for feature selection and GPSO1 in terms of the dimensionality of solutions. NSGPSO was also found to outperform BPSO and perform similarly to GPSO1 in terms of the classification performance.

NSGPSO2 further increases the classification performance of its evolved feature subsets by introducing an external leader set, binary tournament selection of a particle's *gbest* and mutation operators into a multi-objective PSO-based feature selection approach that uses statistical clustering information. These additions were found to improve swarm diversity, leading to a significant increase in the search ability of NSGPSO2 over NSGPSO. In effect, NSGPSO2 outperformed both BPSO and GPSO1, proving to be an effective multi-objective feature selection approach that can explore the Pareto front and find a range of non-dominated feature subsets that achieve high classification performance with a signficantly small number of features.

Chapter 6

Conclusions and Future Work

The work conducted in this project is the *first* research that investigates the use of a set of statistically clustered features in PSO-based feature selection approaches for general classification problems. Specifically, three main objectives were focused upon. Firstly, the development of a single-objective PSO-based approach that selects a subset of features from a set of feature clusters with the constraint of selecting a single feature from each cluster. Secondly, the development of a single-objective PSO-based feature selection approach that allows the selection of multiple features from each feature cluster. Thirdly, the development of a multi-objective PSO-based feature selection approach that can explore the Pareto front of non-dominated feature subsets in terms of their classification accuracy and dimensionality.

The first Objective was achieved by developing the MPPPU, MPTSPPU, RWSPPU and IMPPPU algorithms, which are used within BPSO to select a single feature from each feature cluster during the particle position update step. Experimental results show that by using the statistical information to select a single feature from each feature cluster, the newly developed algorithms can significantly reduce the number of features in the datasets. In most cases, this led to either maintaining or improving the classification performance over using all features. By introducing a greater amount of stochasticity in the MPTSPPU, RWSPPU and IMPPPU algorithms, the swarm diversity was greatly improved compared to the MPPPU algorithm, further improving the classification performance on most datasets with feature subsets of equal size.

To achieve the second Objective, three new PSO-based feature selection algorithms were developed (MFPPU, GPSO1 and GPSO2) that can select multiple features from each feature cluster. The experimental results show that by allowing the selection of multiple features from each cluster, a number of complimentary features can be discovered and included in the evolved solutions. In effect, the evolved feature subsets achieve a higher classification accuracy with a smaller number of features than using all features. By introducing a heuristic that aims to minimise the number of features selected per cluster, GPSO1 dramatically reduces the number of redundant features selected over the MFPPU algorithm, resulting in a significant increase in classification performance. The newly developed Gaussian fitness function which is used in GPSO2 is found to trade a small amount of classification accuracy for a smaller number of features. In turn, GPSO2 was found to be better at selecting a minimal number of features, providing shorter classifier training time and better classifier understandability. On the other hand, GPSO1 was found to be better at achieving high classification performance which is of utmost importance in classification tasks.

For the third objective, two new multi-objective PSO-based feature selection algorithms were developed (NSGPSO and NSGPSO2). By using non-dominated sorting and statistical clustering information together within a multi-objective PSO-based approach, NSGPSO and

NSGPSO2 are found to be able to effectively explore the Pareto front, evolving a range of feature subsets that achieve high classification performance with a small number of features. NSGPSO2 introduces an external leader set, binary tournament selection of a particle's *gbest* and mutation operators into a multi-objective PSO-based feature selection approach that uses statistical clustering information. These additions were found to improve swarm diversity over NSGPSO, leading to a significant increase in the search ability of NSGPSO2 over NSGPSO. In effect, NSGPSO2 outperformed NSGPSO in terms of the classification performance and dimensionality of the evolved non-dominated feature subsets. Both NSGPSO and NSGPSO2 were found to outperform standard BPSO for feature selection, evolving a range of non-dominated feature subsets that achieve high classification performance with a significantly small number of features.

6.1 Future Work

The feature selection algorithms of objective 2 (MFPPU, GPSO1 and GPSO2) automatically evolve solutions that contain a range of different features. Some of these features are complimentary to one another which can greatly improve the classification accuracy when included into a solution [2]. However, unless we have sufficient domain knowledge, it is particularly challenging to pinpoint which features in a dataset are complimentary to one another due to the complex interactions between features [2]. By analyzing the evolution of the selected features within MFPPU, GPSO1 and GPSO2 over a series of iterations, it may be possible to uncover information about the interactions between features. This could help in discovering heuristics that could lead the swarm in PSO towards selecting complimentary features, whilst avoiding conflicting features. Due to the scope of this project, the analysis of feature interaction within the MFPPU, GPSO1 and GPSO2 algorithms is left for future work.

The Gaussian fitness function is developed and used in GPSO2 to penalise solutions that select a large number of features per cluster. The results show that the Gaussian fitness function guides GPSO2 towards evolving feature subsets that have fewer features, but this achieves a slightly lower classification performance. Although the resulting effects of the Gaussian fitness function on the feature selection approach are known, the effects upon the swarm diversity and search abilities within PSO due to the parameter setup of the Gaussian fitness function are unknown and need to be explored in detail. By doing so, a relationship between the parameters of the Gaussian fitness function and the trade-off between the dimensionality of the feature subset and its classification performance may be able to be discovered. From this, an optimal set of parameters could be revealed which maximize classification performance and minimise the number of features selected. The analysis of the effects of the Gaussian fitness function parameters are left for future work.

NSGPSO2 introduces three new aspects to the multi-objective PSO-based feature selection approach. These additions include an external leader set, binary tournament selection of a particle's *gbest* and mutation operators. Together, these additions are found to improve the swarm diversity over NSGPSO and result in the evolution of feature subsets that achieve higher classification performance with a smaller number of features. However, the individual effects of each addition on the search abilities of the swarm are unknown. In further research, the individual impact of the three new additions in NSGPSO2 needs to be explored as to ascertain which additions are necessary/unnecessary and which additions can be removed or improved.

Bibliography

[1] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern Recogn.*, vol. 43, pp. 5–13, Jan. 2010.

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

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

[4] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1, pp. 245–271, 1997.

[5] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[6] A. W. Whitney, "A direct method of nonparametric measurement selection," *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 1100–1103, 1971.

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

[8] H. Yuan, S.-S. Tseng, W. Gangshan, and Z. Fuyan, "A two-phase feature selection method using both filter and wrapper," in *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 132–136, IEEE, 1999.

[9] K. Neshatian and M. Zhang, "Dimensionality reduction in face detection: A genetic programming approach," in *Image and Vision Computing New Zealand, 2009. IVCNZ'09. 24th International Conference*, pp. 391–396, IEEE, 2009.

[10] H. Ming, "A rough set based hybrid method to feature selection," in *Knowledge Acquisition and Modeling, 2008. KAM'08. International Symposium on*, pp. 585–588, IEEE, 2008.

[11] 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.

[12] C.-S. Yang, L.-Y. Chuang, J.-C. Li, and C.-H. Yang, "Chaotic maps in binary particle swarm optimization for feature selection," in *Soft Computing in Industrial Applications, 2008. SMCia'08. IEEE Conference on*, pp. 107–112, IEEE, 2008.

[13] 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. 1, pp. 29–38, 2008.

[14] C.-S. Yang, L.-Y. Chuang, C.-H. Ke, and C.-H. Yang, "Boolean binary particle swarm optimization for feature selection," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pp. 2093–2098, IEEE, 2008.

[15] 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, no. 10, pp. 12699–12707, 2011.

[16] J. Kennedy and W. M. Spears, "Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence.*, pp. 78–83, IEEE, 1998.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197, 2002.

[18] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Parallel Problem Solving from Nature PPSN VI*, pp. 839–848, Springer, 2000.

[19] E. Zitzler, M. Laumanns, L. Thiele, E. Zitzler, E. Zitzler, L. Thiele, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," 2001.

[20] T. M. Hamdani, J.-M. Won, A. M. Alimi, and F. Karray, "Multi-objective feature selection with NSGA II," in *Adaptive and Natural Computing Algorithms*, pp. 240–247, Springer, 2007.

[21] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Genetic and Evolutionary Computation—GECCO 2003*, pp. 37–48, Springer, 2003.

[22] F. R. Bach and M. I. Jordan, "A probabilistic interpretation of canonical correlation analysis," 2005.

[23] S. Pledger and R. Arnold, "Multivariate methods using mixtures: Correspondence analysis, scaling and pattern-detection," *Computational Statistics & Data Analysis*, 2013.

[24] E. Matechou, I. Liu, S. Pledger, and R. Arnold, "Biclustering models for ordinal data," Presentation at the NZ Statistical Assn. Annual Conference, University of Auckland (2011).

[25] B. Sahu and D. Mishra, "A Novel Feature Selection Algorithm using Particle Swarm Optimization for Cancer Microarray Data," *Procedia Engineering*, vol. 38, pp. 27–31, 2012.

[26] "26th Australasian Joint Conference on Artificial Intelligence," 2013. http://ai2013.otago.ac.nz/accepted-papers/.

[27] M. R. Sierra and C. A. C. Coello, "Improving PSO-Based multi-objective optimization using crowding, mutation and-dominance," in *Evolutionary Multi-Criterion Optimization*, pp. 505–519, Springer, 2005.

[28] "Evo* 2014." http://www.evostar.org/.

[29] R. D. King, C. Feng, and A. Sutherland, "Statlog: comparison of classification algorithms on large real-world problems," *Applied Artificial Intelligence an International Journal*, vol. 9, no. 3, pp. 289–333, 1995.

[30] T.-S. Lim, W.-Y. Loh, and Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

[31] P. L. Lanzi, W. Stolzmann, and S. W. Wilson, *Learning classifier systems: from foundations to applications*, vol. 1813. Springer, 2000.

[32] B. Baharudin, L. H. Lee, and K. Khan, "A review of machine learning algorithms for text-documents classification," *Journal of advances in information technology*, vol. 1, no. 1, pp. 4–20, 2010.

[33] V. C. Gandhi and J. A. Prajapati, "Review on Comparison between Text Classification Algorithms," *International Journal*, 2012.

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

[35] M. M. Millonas, "Swarms, phase transitions, and collective intelligence," tech. rep., Los Alamos National Lab., NM (United States), 1992.

[36] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, vol. 5, pp. 4104–4108, IEEE, 1997.

[37] H. Almuallim and T. G. Dietterich, "Learning boolean concepts in the presence of many irrelevant features," *Artificial Intelligence*, vol. 69, no. 1, pp. 279–305, 1994.

[38] S. C. Yusta, "Different metaheuristic strategies to solve the feature selection problem," *Pattern Recognition Letters*, vol. 30, no. 5, pp. 525–534, 2009.

[39] S. D. Stearns, "On selecting features for pattern classifiers," in *Proceedings of the 3rd International Joint Conference on Pattern Recognition*, pp. 71–75, 1976.

[40] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern recognition letters*, vol. 15, no. 11, pp. 1119–1125, 1994.

[41] B. Xue, L. Cervante, L. Shang, W. N. Browne, and M. Zhang, "A multi-objective particle swarm optimisation for filter-based feature selection in classification problems," *Connection Science*, vol. 24, no. 2-3, pp. 91–116, 2012.

[42] B. Xue, M. Zhang, and W. N. Brown, "New fitness functions in binary particle swarm optimization for feature selection," *IEEE Congress on Evolutionary Computation(CEC'2012)*, pp. 2145–2152, 2012.

[43] 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.

[44] K. Bache and M. Lichman, "UCI Machine Learning Repository," 2013.

[45] T. Abeel, Y. Van de Peer, and Y. Saeys, "Java-ML: A machine learning library," *The Journal of Machine Learning Research*, vol. 10, pp. 931–934, 2009.

[46] F. Van Den Bergh, *An analysis of particle swarm optimizers*. PhD thesis, University of Pretoria, 2006.

[47] B. Xue, M. Zhang, and W. N. Browne, “Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach,” *IEEE Transactions on Systems, Man, and Cybernetics (Part B)*, 2012.

[48] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, “Design patterns: Abstraction and reuse of object-oriented design,” in *ECOOP '93*, pp. 406–431, Springer-Verlag, 1993.

Appendix A

A.1 Roulette Wheel Feature Selection PPU

The Roulette Wheel Feature Selection (RWSPPU) algorithm selects features from a set of feature clusters proportional to their probability within their associated clusters, providing features with low probabilities a chance for selection. As each feature within a cluster has at least a small chance for selection, particles are unlikely to prematurely converge on a local optimum, maintaining swarm diversity throughout the evolutionary process. The pseudocode of the RWSPPU algorithm is shown in Algorithm 6.

Algorithm 6: Pseudocode of RWSPPU

```
Input : $Particle_i$: the $ith$ Particle; $FC$: a set of feature clusters, where each feature cluster is a set of
        feature indices
begin
    set all elements in the position vector of $Particle_i$ equal to 0;
    foreach cluster in $FC$ do
        numFeatures ← $|FC[\text{ cluster }]|$ ;
        probabilitySum ← 0 ;
        unNormalisedProbabilities ← $\{P_1^*, P_2^*, ..., P_{\text{numFeatures}}^*\}$ ;
        normalisedProbabilities ← $\{P_1, P_2, ..., P_{\text{numFeatures}}\}$ ;
        foreach feature in $FC$[cluster] do
            unNormalisedProbabilities [ feature ] ← the unnormalised probability of the current feature
            using Equation (A.1) ;
            probabilitySum ← probabilitySum + unNormalisedProbabilities [ feature ] ;
        end
        foreach feature in $FC$[cluster] do
            normalisedProbabilities [ feature ] ← $\frac{\text{unNormalisedProbabilities[feature]}}{\text{probabilitySum}}$ ;
        end
        featureIndexRanges ← divide the interval $[0, 1]$ into ranges proportional to
        normalisedProbabilities ;      /* each feature index is associated to a single range */
        randomValue ← draw a random value uniformly from the interval $[0, 1]$ ;
        featureIndex ← get the feature index that is associated with the range in featureIndexRanges that
        contains randomValue;
        $Particle_i$.position[ featureIndex ] = 1 ;   /* Include the selected feature in the position
        of $Particle_i$ */
    end
end
```

The RWSPPU algorithm starts by erasing the given particle's position. Following this, the algorithm iterates over each feature cluster in order to select a single feature from each cluster. The first step of the inner loop calculates the unnormalised feature probabilities for all features within the given feaure cluster. The unnormalised probability ϕ_i of feature i is obtained through pushing its corresponding velocity value v_i through the following sigmoid function.

$$\phi_i = \frac{1}{1 + 4e^{-2v_i}} \tag{A.1}$$

The unnormalised feature probabilities are then normalised over all features in the given feature cluster into the range between 0 and 1 that represents its probability of being selected. A value is then randomly chosen from a uniform distribution between 0 and 1, which is found to match a particular feature that is then included as part of the given particle's position.

The sigmoid function shown in Equation (2.4) was originally used for transforming the velocity vector to a probability vector. However, the gradient of the sigmoid function was found not to be steep enough through the range of given particle velocities. The effect of this is a small difference in probability between features significantly differing in velocity. In order to increase the gradient of the sigmoid function, the velocity value used in the exponential factor is doubled in Equation (A.1). This provided a greater difference in probability between features differing in velocity. Another issue in the standard sigmoid function is that it calculates an unnormalised feature selection probability of 50% for features with neutral velocity ($v_i = 0$). The combination of many of these values significantly disadvantage the chance of selection for features with higher velocities. As to reduce the chances of selection for features with smaller velocity values, the entire exponential value is quadrupled in Equation (A.1). This decreases the chances for selection of features with lower velocities, whilst increasing the chances for selection of features with higher velocities.

The RWSPPU algorithm stochastically guides particles towards promising solutions with the consideration of features proportional to their probability within their associated feature cluster. This gives features with low probabilities a small chance of selection, allowing the exploration of unique regions in the search space that are likely to go ignored using the MPTSPPU and MPPPU algorithms.

A.2 Incremental Maximum Probability PPU

The increased stochasticity in the MPTSPPU and RWSPPU algorithms allow particles to effectively search a wide region of the search space. However, as the amount of stochasticity remains relatively consistent throughout the evolutionary process, particles in the swarm do not converge to a final solution. Contrastingly, due to the small amount of stochasticity in the MPPPU algorithm, particles in the swarm quickly converge to a nonoptimal final solution. The Incremental Maximum Probability PPU (IMPPPU) combines the advantages of the MPPPU, MPTSPPU and RWSPPU algorithms with a stochastic maximum probability approach that selects a feature from a given feature cluster either randomly or with the highest probability (highest velocity). At iteration one, the chances of selecting the feature with the highest probability from a given feature cluster is very small. This chance increases linearly throughout the evolutionary process untill the very last iteration in which there is a 100% chance that the feature from each feature cluster with the highest probability (highest velocity) will be chosen. The IMPPPU algorithm allows the swarm's particles to initially search a wide region of the search space in order to find many different optimas. As the evolutionary process continues, the stochasticity decreases which narrows the search towards features with maximum probability, allowing the convergence of the swarm to an optimal solution. The pseudo-code of the IMPPPU algorithm is shown in Algorithm 7.

The IMPPPU algorithm starts by calculating the chance of selection for the feature with the highest probability (highest velocity) from the given feature cluster in the current iteration. This becomes the Temperature (T) and is calculated using the following equation where $T_{min} = 0.1$ and $T_{max} = 1$.

Algorithm 7: Incremental Maximum Probability PPU

```
Input : $Particle_i$: the $ith$ Particle; $FC$: a set of feature clusters, where each feature cluster is a set of
        feature indices; $T_{min} = 0.1$; $T_{max} = 1$; $Iteration_{current} = thecurretiteration$
begin
    set all elements in the position vector of $Particle_i$ equal to 0;
    temperature ← the temperature of $Iteration_{current}$ using Equation (A.2);
    foreach cluster in $FC$ do
        numFeatures ← $|FC$[ cluster ]$|$ ;
        featureIndex ← -1 ;
        randomValue ← draw a random value uniformly from the interval $[0, 1]$ ;
        if randomValue $>$ temperature then
            feature ← get a random discrete feature index from the interval $[0,$ numFeatures$]$ ;
            featureIndex ← $FC$[ cluster ][ feature ] ;
        else
            maxVelocity ← $-\infty$ ;
            for feature = 0 to numFeatures do
                featureVelocity ← velocity of feature at index $FC$[ cluster ][ feature ] ;
                if featureVelocity $>$ maxVelocity then
                    maxVelocity ← featureVelocity;
                    featureIndex ← $FC$[ cluster ][ feature ] ;
                end
            end
        end
        $Particle_i$.position[ featureIndex ] = 1 ;    /* Include the selected feature in the position
        of $Particle_i$ */
    end
end
```

$$T = T_{min} + (T_{max} - T_{min}) * \frac{Iteration_{current}}{Iteration_{total}} \tag{A.2}$$

Secondly, the given particle's position is erased. Following this, the algorithm iterates over each feature cluster in order to select a feature from each cluster. Within this inner loop, the feature with maximum velocity within the given feature cluster is selected and included in the particle's position with a chance proportional to the calculated Temperature. Otherwise, a random feature is selected and included in the particle's position. The IMPPPU algorithm starts off with a selection of features that are almost completely random, allowing the algorithm to discover a wide range of good starting positions. Throughout the search, the IMPPPU algorithm stochastically guides particles towards optimal solutions and as the iterations increase, the stochasticity decreases. This allows the continuous narrowing of the search throughout the evolutionary process, ruling out less optimal solutions in favour of better ones.

A.3 PPU Design and Implementation

Each PPU algorithm was first implemented as a separate method incorporated within the Particle class. For each PPU algorithm, there existed a unique constant that represented it. A variable is then assigned to match one of these constants depending on the PPU algorithm desired for use. At each iteration, the PPU algorithm found to match the variable would then be invoked on each particle during the position update step. Although this implementation worked fine in practice, it presented a number of undesirable software characteristics as discussed next.

A.3.1 Reuse and Understandability Issues

Each particle contained multiple PPU algorithms defined in methods within its class definition. These methods are similar to one another, resulting in a failure to reuse code effectively. Furthermore, as the number of different PPU methods increased, the Particle class became more cluttered. This baggage introduces understandability issues within the Particle class as there becomes multiple methods to choose from for the particle position update step. The placement of methods in the particle class implies that any PPU algorithm could be used at any iteration during a run in PSO. However, only a single PPU algorithm should be used to update a particle's position during an entire run as to maintain consistency when testing.

A.3.2 Coupling and Maintenance Issues

At each iteration, a PPU algorithm is invoked on each particle in order to update its position. The algorithm invoked is determined by a method in the Swarm class that uses a set of unique PPU algorithm constants in a large messy if/else statement. This increases the coupling between the Swarm and Particle classes as the Swarm class has to maintain knowledge of all PPU algorithms available in the Particle class. Through this coupling, maintenance issues arise in which every time a new PPU algorithm is created, multiple areas of the code have to be updated. Firstly, the new PPU algorithm is constructed as a method in the Particle class. Secondly, a constant corresponding to the algorithm has to be produced and assigned a value. Lastly, the if/else statement has to be updated in order to be able to invoke the new PPU algorithm.

A.3.3 Improving Software Design with the Strategy Pattern

As each PPU method accepts the same parameters and performs a similar job, an opportunity arose to use the strategy pattern [48] to improve upon the current software design. Firstly, a PPU interface is designed consisting of a single position update method that requires a particle and a set of feature clusters as parameters. Secondly, each PPU algorithm is extracted from the Particle class into a single concrete implementation of the PPU interface. This reduces the amount of unused code within the Particle class which helps resolve understandability issues. Furthermore, an improved reuse of code is obtained by factoring out common code within different methods. Thirdly, the if/else statements used for determining which PPU method to invoke are collapsed into a single statement that asks the particle to update its position. The corresponding method in the particle class then simply uses its associated concrete PPU strategy object to update its position. This reduces the coupling observed between the Swarm and Particle classes. It also improves the maintenance of software as any problems encountered during the particle position update process can be traced to a single method inside a single strategy class.

Refactoring using the Strategy pattern significantly improves software design and provides substantial benefits for further development. Firstly, the construction of new PPU's are kept simple through a standard contract defined in the PPU interface. After construction, each PPU can easily evolve through subclassing in order to effectively reuse code, whilst providing additional functionality. Testing also becomes easier as different PPU algorithms can be tested with a simple switch of a particle's Strategy object used in position updating.

Appendix B

B.1 Gaussian Multiple Feature PPU

GPSO1 and GPSO2 are BPSO-based feature selection approaches which differ from BPSO in the particle position update process. Both algorithms use the newly developed GMFPPU algorithm. GMFPPU uses a Gaussian distribution over each feature cluster when updating particle positions to determine the chances of selecting a certain number of desired features from a given cluster. The parameters of the Gaussian distribution are defined to penalise the selection of a large number of features from each cluster which helps guide the GMFPPU algorithm towards selecting a minimal number of features from each cluster. As shown in Objective 1, a single feature can be used to represent a given cluster. Therefore, a mean of 1 is chosen for the Gaussian distribution which ensures the selection of a single feature from a cluster is given a higher probability than the selection of more than one feature. The standard deviation of the Gaussian distribution is calculated using the following equation:

$$Log(10 \times |cluster|) \tag{B.1}$$

As each cluster must contain at least one feature and $Log(10 \times 1) = 2.303$, the equation evaluates the standard deviation to be greater than 2 for feature clusters of any size. This enforces a very small penalty for selecting 2 or 3 features from a cluster which allows the GMFPPU algorithm to easily select more than 1 feature from a given cluster, regardless of its size. In turn, this allows the GMFPPU algorithm to discover complex feature interaction, even within feature clusters that contain a small number of features.

The standard deviation of the Gaussian distribution increases with the feature cluster size at a logarithmic scale which can be seen from the blue line in Figure B.1. This is preferred over a constant value shown as the yellow line in Figure B.1 as it provides a greater chance for selecting more features from a cluster that contains a large number of features. The logarithmic scaling is also used as opposed to a linear scaling shown as the red line in Figure B.1 as it guides the GMFPPU algorithm towards selecting fewer features from large feature clusters as can be seen from Figure B.1. This is beneficial as it minimises the chance of introducing redundant features into solutions. The effects of the standard deviation functions upon a Gaussian distribution of a typical feature cluster containing 5 features and a large feature cluster containing 30 features can be seen in Figures B.2(a) and B.2(b). Figure B.2(a) shows that the logarithmic standard deviation function (shown as the blue line) provides a chance of selecting 1, 2, 3 or 4 features that is more even than the linear standard deviation function (shown as the yellow line) which favors selecting 1 or 2 features from small clusters. Hence, when used for the Gaussian distribution within the GMFPPU algorithm, the logarithmic standard deviation function is likely to discover more feature interaction within small feature clusters. In Figure B.2(b), the logarithmic standard deviation function (shown as the blue line) provides a much smaller chance for selecting more than 10 features from a large feature cluster than the other two standard deviation functions. Hence,

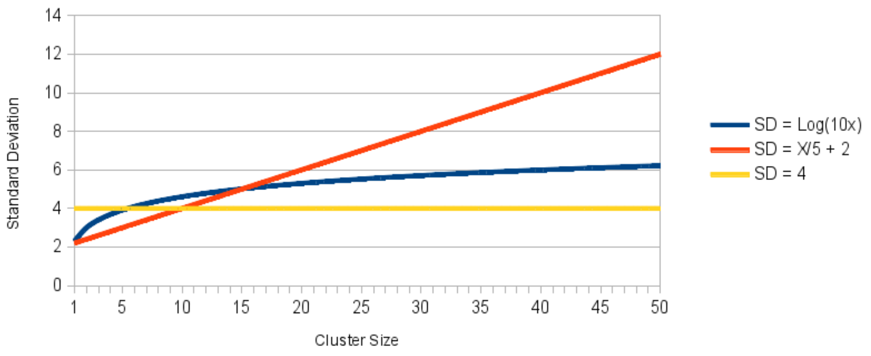
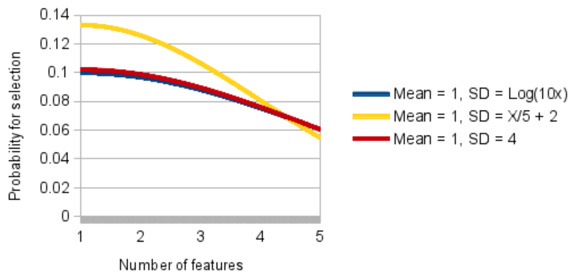
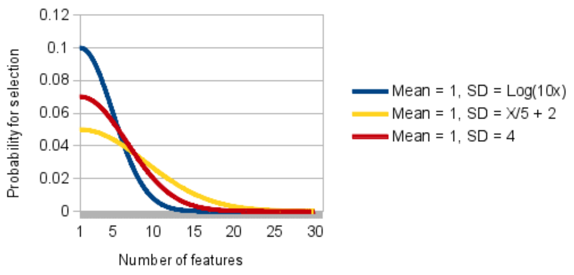


Figure B.1: Standard deviation functions for feature clusters containing up to 50 features

as many features are similar within a cluster, less redundant features will be introduced into solutions when the logarithmic standard deviation function is used for the Gaussian distribution within the GMFPPU algorithm.



(a) Gaussian distributions for a feature cluster with 5 features



(b) Gaussian distributions for a feature cluster with 30 features

Figure B.2: The effects of the standard deviation functions upon two Gaussian distributions

The Gaussian distribution function used to calculate the probabilty of selecting x features from a given cluster is shown in Equation (B.2) where the mean = 1, x is the number of features selected for a given cluster and the standard deviation is calculated according to Equation (B.1). The pseudo-code for the GMFPPU algorithm is shown in Algorithm 8.

$$g(x) = \frac{\exp(-\frac{(x-1)^2}{2log^2(10\times|cluster|)})}{\sqrt{2\pi}\log(10\times|cluster|)} \tag{B.2}$$

Like the MFPPU algorithm, the GMFPPU algorithm first erases a particle's position and then for each feature cluster, a set of desired features is constructed and includes the features selected by the particle in the BPSO position update process using only the features held within the given feature cluster in Equation (2.3). These features are then sorted by their associated probabilities for selection (given by Equation 2.4) so that the first feature has the highest probability. Only a subset of the features contained in the desired set will be selected by the given particle in the next part of the algorithm which determines which subset of features will be selected from the features contained in the desired set. The GMFPPU algorithm allows the next part of the algorithm to be skipped if no features are desired within a given cluster. This allows a particle's position to contain zero features from a cluster that contains only useless features. The next part of the algorithm which focuses on selecting a subset of the features from the set of desired features starts by calculating the standard deviation of the Gaussian distribution for the given feature cluster according to Equation

Algorithm 8: Pseudocode of GMFPPU

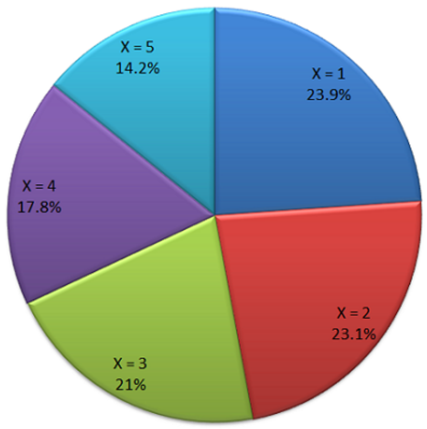
```
Input : $Particle_i$: the $ith$ Particle; $FC$: a set of feature clusters, where each feature cluster is a set of feature indices
1  begin
2      set all elements in the position vector of $Particle_i$ equal to 0;
3      foreach cluster in $FC$ do
4          desiredFeatures ← initialise an empty map of desired features and their probabilities ;
5          foreach feature in $FC$[cluster] do
6              featureProbability ← the probability of feature using the feature's velocity in Equation (2.4)
7              include feature and featureProbability into desiredFeatures if the BPSO position update equation given
               by Equation (2.3) evaluates to 1 given the feature ;
8          end
9          if desiredFeatures is not empty then
10             sortedFeatures ← sort desiredFeatures by ascending probability ;
11             gaussianDistribution ← construct a Gaussian distribution with mean = 1 and standardDeviation
               calculated using cluster in Equation (B.1)
12             numDesiredFeatures ← |desiredFeatures| ;
13             normalisedProbabilities ← $\{P_1, P_2, ..., P_{\text{numDesiredFeatures}}\}$ ;        /* Entry $i$ holds the normalised
               probability of selecting $i$ features from cluster  */
14             for numFeatures = 1 to numDesiredFeatures do
15                 normalisedProbabilities [ numFeatures ] ← calculate the normalised probability of selecting
                   numFeatures features from cluster using gaussianDistribution
16             end
17             numFeatures ← Select a number of features for selection proportional to their associated probabilities
               given by normalisedProbabilities
18             include numFeatures sortedFeatures in the position of $Particle_i$ ;
19         end
20     end
21 end
```

(B.1). The Gaussian distribution $g(x)$ which is shown in Equation (B.2) is then constructed with the calculated standard deviation and mean of 1. $g(x)$ is used to individually calculate the Gaussian score of selecting a single feature, two features, three features and so on up to the number of features contained in the desired feature set for the given cluster. These Gaussian scores are then normalised and resemble a set of probabilities that sum to 1. In this set, entry i is the normalised probability of selecting i features from the given cluster. Examples of the sets of normalised probabilities for two different feature clusters can be seen in Figures B.3(a) and B.3(b). Each segment of each pie chart resembles the probability of a certain number of features being selected from the given cluster. For example, Figure B.3(a) shows that there is a 14.2% chance of selecting five features when five features are desired from a feature cluster containing five features. A number of features X is then chosen proportional to the set of normalised probabilities. After this, the X features with the highest probabilities are included in the position of $Particle_i$. It is to be noted, that if the desired feature set contains a single feature, it is automatically included in the particle position as its normalised probability is 1.0 regardless of its unnormalised probability given by the Gaussian distribution.

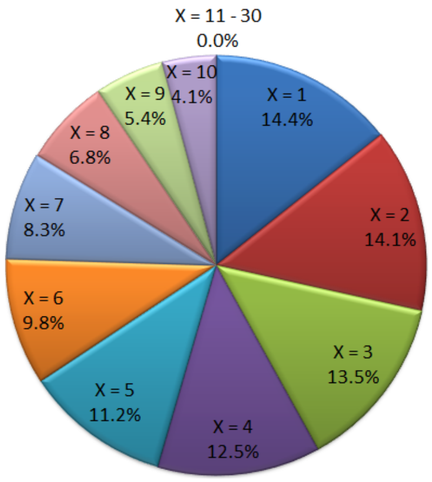
An example of a typical run-through of feature selection from a single feature cluster with the GMFPPU algorithm is outlined below. In this example, we focus only on the inner loop of the GMFPPU algorithm where we select a subset of features from a single feature cluster that contains five features.

- Step 1: The desired set of features is constructed and includes each of the five features if and only if Equation (2.4) in the particle position update of BPSO evaluates to 1 for the given feature using its associated velocity. In this example, we assume Equation (2.4) evaluates each feature is equal to 1 and therefore the desired feature set contains all five features.

- Step 2: A Gaussian distribution $g(x)$ is constructed using Equation (B.2), with mean =



(a) The probability of selecting X features from a feature cluster containing 5 features and a desired feature set containing 5 features



(b) The probability of selecting X features from a feature cluster containing 30 features and a desired feature set containing 10 features

Figure B.3: Normalised probabilities for selecting X features from two feature clusters of differing size

1 and standard deviation according to Equation (B.1).

- Step 3: The Gaussian scores of selecting x features are calculated using $g(x)$ from $x = 1$ to $x = 5$.
- Step 4: The Gaussian scores are normalised as to sum to 1. The probabilities of selecting x features can be seen in Figure B.3(a). e.g. The chances of selecting three features is 21%.
- Step 5: A number of features X is chosen via a roulette wheel selection. In this example, we assume that the roulette wheel selection determines that 2 features (X = 2) should be selected in the next step. (the chances of this happening is proportional to 23.1%).
- Step 6: The X features (2, in this case) with the highest probabilities are included in the position of $Particle_i$.

Used within BPSO, the GMFPPU algorithm can select a minimal number of features that are representatives of the features from each cluster. In turn, the GMFPPU algorithm is expected to select fewer features than the MFPPU algorithm on datasets containing a large number of features, whilst maintaining or improving the classification accuracy.

B.2 Gaussian Fitness Function

GPSO2 is a variant of GPSO1 which introduces the Gaussian fitness function. The Gaussian fitness function evaluates the fitness of a particle by its classification performance and a Gaussian distance measure between the number of features selected by the particle and the total number of feature clusters. This new fitness function aims to further minimise the number of features selected per cluster by evaluating the fitness of a solution based in part on how close the solution is to a single feature per cluster.

The classification accuracy of the feature subset is a common performance measure to use as the fitness of a particle in PSO for feature selection. Choosing the classification accuracy as the fitness of a particle leads the swarm towards a *pbest* and *gbest* that have high training classification accuracies, which in general, implies a high classification accuracy on the unseen data. However, as the fitness function evaluates a feature subset solely on its classification performance, a feature subset with high fitness may contain a number of redundant features. These unneeded features increase the amount of time needed to construct classification algorithms, whilst they may even decrease the classification accuracy on the unseeen data [2]. The Gaussian fitness function addresses this issue by including a Gaussian distance measure in the fitness function which evaluates how close the feature subset is to containing a single feature from each cluster with a score of 1 representing the best score and 0 representing the worst score. For example, a feature subet that contains a single feature per cluster has a perfect Gaussian distance measure score, evaluating the solution as equal to 1. Contrastingly, a feature cluster that contains many features per cluster has a bad Gaussian distance measure score, evaluating the solution as close to 0. The reasoning behind the construction and inclusion of the Gaussian distance measure in the fitness function is because the results of Objective 1 suggest that a single feature from each cluster provides most of the information needed to describe the entire dataset with high classification performance and low redundancy. Therefore, the Gaussian distance measure aims to promote the selection of a minimal number of features per cluster.

To calculate the Gaussian distance of a solution, a multivariate Gaussian distribution is constructed and used. As a desired solution contains a single feature per cluster, a mean of 1 is chosen on every dimension of the multivariate Gaussian distribution. In order to be consistent with the GMFPPU algorithm, the logarithmic standard deviation function defined in Equation (B.1) is used to define the standard deviation along each dimension. This provides a small penalty for selecting a small number of features from any cluster and a large penalty for selecting a large number of features from large clusters. As the Gaussian distribution is multivariate, a covariance matrix is needed to establish the variances along each dimensions. To achieve this, the standard deviation values along each dimension are squared as to equal the variances and are placed on the diagonal entries of a $N \times N$ covariance matrix where N is the number of feature clusters. By using the logarithmic standard devation function to calculate the variance values used in the covariance matrix, we ensure the variance along each dimension scales with the size of the particular feature cluster. By assigning all non-diagonal entries the value of 0, we also ensure that the selection of features from each cluster is independent from one another. An example of the transformation between a set of feature clusters for the Wine dataset: $\{1,4,9,12\},\{0,3,11\},\{5,7\},\{6,8\},\{10\},\{2\}$ to the covariance matrix of the multivariate Gaussian distribution used in the Gaussian fitness function is shown below.

$$\begin{pmatrix} 13.61 & 0 & 0 & 0 & 0 & 0 \\ 0 & 11.57 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8.97 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.30 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5.30 \end{pmatrix}$$

As we can see, the feature clusters containing more features have a higher variance and hence, allow a greater amount of variation in the number of features selected from those clusters compared to smaller feature clusters. In effect, there is a greater chance of selecting more features from large feature clusters than from smaller feature clusters.

After determaining the set of features selected by a given particle, a vector $\mathbf{x}$ can be constructed which denotes the number of features selected per cluster. This process is achieved by counting the number of features selected within each of the feature clusters and for each feature cluster i, assigning the ith entry in $\mathbf{x}$ to the number of features contained in feature cluster i. The vector $\mathbf{x}$ is then used within the multivariate Gaussian distribution to calculate the Guassian distance measure. This function is shown in Equation (B.3) where $\mathbf{x}$ is the vector with each entry i corresponding to the number of features selected from the ith feature cluster, $\mathbf{m}$ is the mean vector in which the ith dimension represents the optimal number of features to be selected from the ith feature cluster which is 1 along each dimension, Cov is the multidimensional covariance matrix for the feature cluster, Cov^{-1} is its inverse and $|Cov|$ is its determinant. A value of 1 along each dimension in the mean vector $\mathbf{m}$ ensures that a solution containing 1 feature per cluster is evaluated by the multivariate Gaussian distribution to be equal to 1, a perfect solution. On the other hand, a solution that contains many features per cluster will have a Gaussian distance measure score close to 0 which is the worst score. In effect, the higher the Gaussian distance measure score, the greater the fitness of the solution.

$$Gaussian\,Distance(\mathbf{x},\mathbf{m}) = \frac{\exp(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T Cov^{-1}(\mathbf{x}-\mathbf{m}))}{2\pi^{\frac{|cluster|}{2}}\sqrt{|Cov|}} \tag{B.3}$$

The fitness of a particle using the Gaussian fitness function is given in Equation (4.4). As the classification performance is assumed to be much more important than the number of features selected per cluster, the classification accuracy determines 98% of a particle's fitness. The other 2% is evaluated by the Gaussian distance measure. Even though this is a relatively small percentage, it can make a dramatic difference as it allows 2% of the classification accuracy to be sacrificed for the minimisation of the number of features selected per cluster. Like the classification accuracy, a value close to 1.0 for the Gaussian distance measure represents a near perfect fit in which a single feature is chosen from each cluster.

$$Fitness = (0.98 \times Classification\ Accuracy) + (0.02 \times Gaussian\,Distance) \tag{B.4}$$

Due to the Gaussian fitness function evaluating a particle's fitness based partly on the distance between the number of selected features per cluster and a single feature per cluster, the swarm's *pbest* and *gbest* have a higher chance of containing a smaller number of features per cluster. As the *pbest* and *gbest* are used to guide the swarm towards solutions, the Gaussian fitness function is expected to guide the swarm towards selecting solutions with a minimal number of features per cluster. In turn, this is expected to allow GPSO2 to remove a small number of redundant features from solutions obtained in GPSO1 on large feature clusters.