Evolutionary Computation for Feature Selection in Classification

by

Bach Hoai Nguyen

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

Victoria University of Wellington
2018

Abstract

Classification aims to identify a class label of an instance according to the information from its characteristics or features. Unfortunately, many classification problems have a large feature set containing irrelevant and redundant features, which reduce the classification performance. In order to address the above problem, *feature selection* is proposed to select a small subset of relevant features. There are three main types of feature selection methods, i.e. *wrapper, embedded* and *filter* approaches. Wrappers use a classification algorithm to evaluate candidate feature subsets. In embedded approaches, the selection process is embedded in the training process of a classification algorithm. Different from the other two approaches, filters do not involve any classification algorithm during the selection process. Feature selection is an important process but it is not an easy task due to its large search space and complex feature interactions. Because of the potential global search ability, Evolutionary Computation (EC), especially Particle Swarm Optimization (PSO), has been widely and successfully applied to feature selection. However, there is potential to improve the effectiveness and efficiency of EC-based feature selection.

The overall goal of this thesis is to investigate and improve the capability of EC for feature selection to select small feature subsets while maintaining or even improving the classification performance compared to using all features. Different aspects of feature selection are considered in this thesis such as the number of objectives (single-objective/multi-objective), the fitness function (filter/wrapper), and the searching mechanism.

This thesis introduces a new fitness function based on mutual information which is calculated by an estimation approach instead of the traditional counting approach. Results show that the estimation approach

works well on both continuous and discrete data. More importantly, mutual information calculated by the estimation approach can capture feature interactions better than the traditional counting approach.

This thesis develops a novel binary PSO algorithm, which is the first work to redefine some core concepts of PSO such as velocity and momentum to suit the characteristics of binary search spaces. Experimental results show that the proposed binary PSO algorithm evolve better solutions than other binary EC algorithms when the search spaces are large and complex. Specifically, on feature selection, the proposed binary PSO algorithm can select smaller feature subsets with similar or better classification accuracies, especially when there are a large number of features.

This thesis proposes surrogate models for wrapper-based feature selection. The surrogate models use surrogate training sets which are subsets of informative instances selected from the training set. Experimental results show that the proposed surrogate models assist PSO to reduce the computational cost while maintaining or even improving the classification performance compared to using only the original training set.

The thesis develops the first wrapper-based multi-objective feature selection algorithm using MOEA/D. A new decomposition strategy using multiple reference points for MOEA/D is designed, which can deal with different characteristics of multi-objective feature selection such as highly discontinuous Pareto fronts and complex relationships between objectives. The experimental results show that the proposed algorithm can evolve more diverse non-dominated sets than other multi-objective algorithms.

This thesis introduces the first PSO-based feature selection algorithm for transfer learning. In the proposed algorithm, the fitness function uses classification performance to reduce the differences between domains while maintaining the discriminative ability on the target domain. The experimental results show that the proposed algorithm can select feature subsets which achieve better classification performance than four state-of-the-art feature-based transfer learning algorithms.

List of Publications

- Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "PSO with Surrogate Models for Feature Selection: Static and Dynamic Clustering-based Methods", Memetic Computing, 07 March 2018 (Online). https://doi.org/10.1007/s12293-018-0254-9

- Hoai Bach Nguyen, Bing Xue, Peter Andreae. "Mutual Information for Feature Selection: Estimation or Counting?", Evolutionary Intelligence, vol. 9, no. 3, pp. 95-110, 2016.

- Hoai Bach Nguyen, Bing Xue, Peter Andreae and Mengjie Zhang. "A New Binary Particle Swarm Optimization Approach: Momentum and Dynamic Balance Between Exploration and Exploitation". Submitted to IEEE Transactions on Cybernetics (under revise and resubmit).

- Hoai Bach Nguyen, Bing Xue, Hisao Ishibuchi, Peter Andreae, and Mengjie Zhang. "Multiple Reference Points based Decomposition for Multi-objective Feature Selection in Classification: Static and Dynamic Mechanisms". Submitted to IEEE Transactions on Evolutionary Computation (under revise and resubmit).

- Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "A Particle Swarm Optimization based Feature Selection Approach to Transfer Learning in Classification". Proceedings of 2018 Genetic and Evolutionary

Computation Conference (GECCO 2018). ACM Press. Kyoto, Japan, July 15th-19th 2018. 8pp. (to appear).

- Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "A Hybrid GA-GP Method for Feature Reduction in Classification". Proceedings of the 11th International Conference on Simulated Evolution and Learning (SEAL 2017). Lecture Notes in Computer Science. Vol. 10593. Shenzhen, China. November 10-13, 2017. pp. 591-604.

- Hoai Bach Nguyen, Bing Xue, Peter Andreae and Mengjie Zhang. "Particle Swarm Optimisation with Genetic Operators for Feature Selection". Proceedings of 2017 IEEE Congress on Evolutionary Computation (CEC 2017). Donostia - San Sebastian, Spain, 5-8 June, 2017. pp. 286-293.

- Hoai Bach Nguyen, Bing Xue, Hisao Ishibuchi, Peter Andreae, and Mengjie Zhang. "Multiple Reference Points MOEA/D for Feature Selection". Proceedings of 2017 Genetic and Evolutionary Computation Conference (GECCO 2017) Companion. ACM Press. Berlin, German, 15 - 19 July 2017. pp 157-158.

- Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "Surrogate-model based Particle Swarm Optimisation with Local Search for Feature Selection in Classification". Proceeding of the 20th European Conference on Applications of Evolutionary Computation (EvoApplications 2017) Part I, Lecture Notes in Computer Science. Vol. 10199. Amsterdam. 18-21 April 2017. pp. 487-505. **(Nominated as Best Paper)**

- Bach Hoai Nguyen, Bing Xue and Peter Andreae. "A Novel Binary Particle Swarm Optimisation Algorithm and its Applications on Knapsack and Feature Selection Problems". Asia-Pacific Symposium on Intelligent and Evolutionary Systems (IES2016), Canberra, Australia, 16-18 November 2016. pp. 319-332

- Hoai Bach Nguyen, Bing Xue and Peter Andreae. "Mutual Information Estimation for Filter Based Feature Selection Using Particle Swarm Optimization". Proceedings of the 19th European Conference on the Applications of Evolutionary Computation (EvoApplications 2016, EvoIASP 2016). Lecture Notes in Computer Science. Vol. 9597. Porto, Portugal, March 30 - April 1, 2016. pp. 719-736

Acknowledgments

I would like to express my sincere thanks to all those who gave me the assistance and support during my PhD study. It would not be possible to complete this thesis without your help.

First and foremost, I would like to express my deepest gratitude to my supervisors Dr. Bing Xue, A/Prof. Peter "Pondy" Andreae, and Prof. Mengjie Zhang for the support, entrenchment and guidance. Dr. Bing Xue spends dedicated time and efforts to help me improve my research skill. She always encourages and guides me through all the hard times during my study. A/Prof. Peter Andreae is always nice to talk. All the discussions with him always lead me to think deeper and have better ideas. He also helps me a lot to improve my English. Prof. Mengjie Zhang not only helps me in my research but also assists me in my life. He is always the person I am seeking for help when I face any problem in my life.

I wish to thank all my friends in School of Engineering and Computer Science, especially members of Evolutionary Computation Research Group (ECRG). Special thanks to my officemates Qurrat Ul Ain (Annie) and Fangfang Zhang for their jokes and tasty food.

I wish to thank my parents (Thanh Nguyen and Bon Nguyen) and my little sister (Thao Nguyen) for their supports and encouragements. Last but not least, I would like to thank my girlfriend Nga Kieu, for all her love and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces the problem that this thesis addresses, then describes the motivations, the research goals, and the major contributions of the thesis.

1.1 Problem Statement

Classification is an important task in machine learning, which aims to assign a class label to an instance based on the characteristics or features of the instance. In classification, a classification algorithm is trained on a set of labeled instances, called a training set. The learned classifier is then used to predict the class label of the unlabeled future instances. The performance of a learned classifier depends heavily on the quality of the feature set describing the instances.

Nowadays, with advancements in technology, real-world problems are often described by a large number of features. Due to the "curse of dimensionality" [1], it is difficult to train a classification algorithm efficiently and effectively on a large number of features. Particularly, the increment in dimensionality enlarges the number of possible instances in the instance space, which makes the available data become sparse [2]. In order to achieve reliable results in such high-dimensional problems, classification

algorithms require a large amount of data which usually grows exponentially with respect to the number of features. Therefore, high dimensionality is problematic to any classification algorithm. Fortunately, not all features are necessary or useful.

In contrast to relevant features which provide useful information about the learning task, irrelevant features provide misleading information leading to deterioration in the classification performance [3]. For example, in k-nearest neighbor (KNN), the irrelevant or noisy features may increase the distances between instances from the same class, which makes KNN more difficult to correctly classify instances. In some other classification algorithms such as decision trees (DT) or support vector machines (SVM), the learned model may have to overfit the irrelevant features to cope with the data; in which case it will not work well on unseen/future instances.

Redundant features provide the same or similar information about the learning task as other features. In respect of classification algorithms which directly use training instances in the classification process such as Naive Bayes or KNN, redundant features add unnecessary weights which can reduce the classification performance. For classification algorithms which explicitly build a classification model such as DT or SVM, the redundant features can be removed during the training process. However, the redundant features cause extra complexity which increases the training time.

In order to solve the high dimensionality problem, *feature selection* [4] is proposed to select a small and more informative feature subset from the original features. Feature selection reduces the number of features by removing irrelevant or redundant features, which results in a subset of the original features. The benefits of doing feature selection include improving the learning performance, saving the cost of measuring unused features, and making the learned classifier simpler and easier to understand. A key element of any feature selection algorithm is its evaluation criterion for choosing between alternative subsets of features.

Depending on the evaluation criterion, existing feature selection meth-

ods can fall into three categories: embedded, wrapper and filter approaches [5]. In embedded approaches, the selection process is a part of the training procedure of a classification algorithm; so embedded approaches consider the interactions between selected features and the classification algorithm. However, embedded approaches are only applicable to some specific algorithms such as DT, SVM. In wrappers, a classification algorithm is used to calculate fitness values of feature subsets. In contrast, filters use data characteristics to evaluate feature subsets, which do not involve any classification algorithm. Compared with wrappers, it is more difficult for filters to consider multi-way feature interactions. Therefore, wrappers usually achieve better classification accuracies than filters. However, wrappers produce feature subsets with poorer generality to other classification algorithms (other than the wrapped classification method) than filters. In addition, wrappers are typically more computationally intensive than filters. Given the pros and cons of the filter and wrapper approaches, one of the objectives of this thesis is to improve their effectiveness and/or efficiency, which contributes to the evaluation part of feature selection.

Feature selection has been proven to be an NP-hard combinatorial problem [6], which makes an exhaustive search impractical. This thesis mainly uses Particle Swarm Optimization (PSO) [7] as the searching mechanism to achieve feature selection. As a member of evolutionary computation (EC) family, PSO has a potential global search ability which is suitable to feature selection. In comparison with other (EC) techniques, such as genetic programming (GP) [8], genetic algorithms (GAs) [9], PSO gains more attention since it is simpler, has fewer parameters and converges quicker [10]. Furthermore, although both GAs and PSO have vector-based representations which is suitable to feature selection, PSO is applied more widely. The main reason is that without a careful design, the crossover and mutation operators of GAs might break good groups of complementary features when solving feature selection problems.

Feature selection can be considered a multi-objective problem since

its two main objectives, reducing the number of features and improving the classification performance, are usually in conflict. As a family of population-based optimization techniques, EC can be naturally applied to evolve a set of trade-off solutions for multi-objective problems, including feature selection. The evolutionary multi-objective methods (EMO) fall into three main categories: Pareto dominance-based algorithms, indicator-based algorithm, and decomposition-based algorithm. Most existing multi-objective feature selection work directly apply the Pareto dominance-based algorithms without considering the characteristics of multi-objective feature selection such as its highly discontinuous Pareto front and its partially conflicting objectives.

Transfer learning is an important task in machine learning, which uses gained knowledge from a problem (source domain) to improve learning performance on a different but related problem (target domain). The source and target domains can have different feature spaces and/or data distributions. Transfer learning is useful in many real-world cases, for example when it is difficult or expensive to collect labeled data in a domain, the data or extracted knowledge from related domains can be utilized to assist learning in the concerned domain. The knowledge can be transferred by different approaches such as instance-based [11, 12], parameter-based [13, 14] or feature-based approaches [15, 16, 17]. This thesis targets to achieve transfer learning through feature-based approaches. The main goal is to find a good feature representation to minimize the differences between two domains while maintaining discriminating abilities on both domains.

1.2 Challenges

1.2.1 Fundamental Challenges of Feature Selection

Feature selection is a difficult task because of the following reasons:

- There can be **complex interactions between features**. For example, two weakly relevant features can significantly improve the classification performance when they are selected together. These features are usually called complementary features [10]. In contrast, selecting two relevant features may result in redundancy. In order to produce an optimal subset of features in terms of size and classification performance, the searching algorithm and fitness evaluation have to consider the interactions between features, which is difficult to achieve.

- The **search is expensive**: Given n original features, the total number of possible feature subsets is 2^n. Thus, in feature selection, the search space size grows exponentially with respect to the number of original features. Due to the complex interactions between features, the combination of m individually best features is not necessary the best m features. Therefore, greedy searches generally unable to provide the best results for feature selection. In order to achieve good performance in feature selection, global heuristic searches are required to generate promising feature subset candidates.

1.2.2 Limitations of Filter-based Feature Selection

Nowadays, there are many filter measures for feature selection, for example Fisher score [18, 19], consistency measure [20], correlation measures [21, 22] and information-based measures [23, 24, 25]. Among these measures, information-based measures, specifically mutual information, gain more attention because they are easier to detect non-linear relationship between two random variables [26, 27]. It also has been shown that mutual information achieves better classification performance than other filter measures when they are applied to feature selection [28]. However, in most current approaches, to calculate the mutual information, the probability distribution is achieved by counting the number of instances with

each possible feature value. Therefore, mutual information can only be applied to discrete datasets. In addition, when the number of possible feature values is large, this requires a large number of instances to calculate mutual information accurately. Therefore, how to design a new mutual information-based fitness function, which is efficient and cooperative with EC techniques, can deal with different kinds of datasets, is an open issue.

1.2.3 Limitations of Wrapper-based Feature Selection

In comparison with filter-based feature selection approaches, the wrapper-based approaches usually achieve better classification performance since the evaluation process explicitly considers the interactions between selected features and the classification algorithm. However, the effectiveness of wrappers comes along with their expensive computational cost since it involves a learning process during each evaluation step. The question of how to reduce the computational cost while still maintaining or even improving the feature selection performance is still an open issue.

1.2.4 Limitations of PSO-based Feature Selection

Most of the current PSO-based feature selection algorithms apply continuous PSO, in which each position entry corresponds to an original feature and its value is a real number in the range [0,1]. A threshold is used to determine whether the corresponding feature is selected. It would be more natural to represent each position entry as a binary value i.e. 0 or 1, which is known as binary PSO [29]. However, the performance of binary PSO is limited compared with continuous PSO [30]. The main reason is that binary PSO directly applies the updating equations from continuous PSO without considering the characteristics of binary search spaces. For example, in the binary search space, there is no direction in the movement of a particle; so it is not appropriate to directly apply directed velocity and momentum from the continuous space to the binary space. It is important

to design new updating mechanisms for binary PSO, which follows the main idea of PSO and consider the characteristics of binary search spaces.

1.2.5 Limitations of Multi-objective Feature Selection

Feature selection has two main objectives, which are maximizing the classification performance and minimizing the number of selected features. Most of the recent studies consider feature selection as a multi-objective problem. However, feature selection has its own characteristics which should be considered to evolve better non-dominated feature subsets. Firstly, feature selection has a highly discontinuous Pareto front. Secondly, the two objectives are *not always in conflict* with each other. For example, in some cases removing irrelevant features may improve the classification performance. By detecting conflicting regions, the performance of a multi-objective feature selection algorithm can be improved by focusing more on the detected regions. Many multi-objective algorithms have been applied to feature selection but none of them can address the above characteristics of multi-objective feature selection.

1.2.6 Limitations of Feature-based Approaches for Transfer Learning

In feature-based transfer learning, the goal is to search for a good feature representation, which not only reduces the differences between domains but also maintains as much useful information for classification as possible. Most of the existing feature-based transfer learning approaches aim mainly to achieve the first objective while ignoring the second one. In addition, feature interactions are difficult to be considered in the existing approaches. Although EC techniques have been widely applied to feature selection and achieved promising results, they have not been used to achieve feature-based transfer learning.

1.3 Research Goals

The overall goal of this thesis is to develop new methods to improve the capability of EC-based feature selection in terms of both the evaluations and searching mechanisms, which is expected to efficiently and effectively reduce the number of features while improving the classification performance over using all features. In achieving this overall goal, this thesis is structured around the following five objectives:

1. Develop a new fitness function based on mutual information for feature selection. The proposed fitness function is expected to consider interactions between features, which results in a smaller set of features with better classification performance than using all features and other traditional mutual information-based feature selection algorithms.

2. Develop a surrogate model for wrapper-based feature selection, which is expected to reduce the computational cost while achieving similar or better performance than traditional wrapper-based feature selection algorithms. In wrappers, the main computational cost is from the evaluation process, so the proposed surrogate model focuses on speeding up the evaluation process while maintaining or even improving the evolved feature subsets.

3. Develop a new binary PSO algorithm which can reflect movements in binary search spaces more accurately than the standard one. The proposed binary PSO algorithm is expected to solve feature selection as well as other binary problems better than the standard binary PSO algorithm and other well-known binary EC algorithms.

4. Develop a new multi-objective feature selection approach, which can cope with the characteristics of multi-objective feature selection. The proposed algorithm is expected to evolve a set of non-dominated feature subsets with a wider range of numbers of features and better

classification performance than other well-known EC-based multi-objective feature selection algorithms.

5. Develop an EC-based feature selection approach to transfer learning, which not only minimizes the differences between different domains but also selects the most informative features for classification. The proposed algorithm is expected to evolve a small feature subset with better classification performance than other well-known feature-based transfer learning methods.

1.4 Major Contributions

This thesis makes the following contributions:

1. This thesis proposes a new mutual information-based fitness function in PSO for feature selection, in which kernel density estimation (KDE) is used to calculate the mutual information. The estimator allows mutual information to work directly on continuous datasets, which is the limitation of many mutual information based feature selection methods. The experimental results show that the proposed fitness function achieves similar or better classification algorithm performance than the traditional counting approach on both continuous and numeric discrete datasets.

   Part of this contribution has been published in:

   Hoai Bach Nguyen, Bing Xue and Peter Andreae. "Mutual Information Estimation for Filter Based Feature Selection Using Particle Swarm Optimization". Proceedings of the 19th European Conference on the Applications of Evolutionary Computation (EvoApplications 2016, EvoIASP 2016). Lecture Notes in Computer Science. Vol. 9597. Porto, Portugal, March 30 - April 1, 2016. pp. 719-736

Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "Mutual Information for Feature Selection: Estimation or Counting?", Evolutionary Intelligence, vol. 9, no. 3, pp. 95-110, 2016.

2. This thesis proposes a novel binary PSO (BPSO) algorithm, which can be applied to not only feature selection but other binary optimization problems. In the proposed BPSO, the velocity and momentum are reformulated as a flipping probability and a stickiness property, respectively. The exploration and exploitation of the proposed BPSO are investigated to develop a dynamic mechanism which updates the algorithm's parameters. The experimental results show that the new velocity and momentum concepts assist BPSO to evolve better solutions than the standard BPSO algorithm and other EC algorithms. This is the first time in BPSO, the four important concepts, i.e. velocity, momentum, exploration and exploitation, are investigated systematically.

   Part of this contribution has been published/submitted to:

   Hoai Bach Nguyen, Bing Xue and Peter Andreae. "A Novel Binary Particle Swarm Optimization Algorithm and its Applications on Knapsack and Feature Selection Problems". Asia-Pacific Symposium on Intelligent and Evolutionary Systems (IES2016), Canberra, Australia, 16-18 November 2016.pp 319-332.

   Hoai Bach Nguyen, Bing Xue, Peter Andreae and Mengjie Zhang. "A New Binary Particle Swarm Optimization Approach: Momentum and Dynamic Balance Between Exploration and Exploitation". submitted to IEEE Transactions on Cybernetics (under revise and resubmit).

3. This thesis proposes a surrogate model for wrapper-based feature selection which reduces the computational cost by selecting a small set of informative instances (surrogate set) from the original training set to quickly locate promising search regions. The surrogate set is

built by firstly applying a hierarchical clustering method to the original training set and then selecting an instance as a representative for each instance cluster. The experimental results show that the surrogate model successfully reduces the computational time, improves the feature selection performance, and partially avoids overfitting.

Part of this contribution has been published in:

Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "Surrogate-model based Particle Swarm Optimization with Local Search for Feature Selection in Classification". Proceeding of the 20th European Conference on Applications of Evolutionary Computation (EvoApplications 2017) Part I, Lecture Notes in Computer Science. Vol. 10199. Amsterdam. 18-21 April 2017. pp. 487–505 **(Nominated as Best Paper)**

Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "PSO with Surrogate Models for Feature Selection: Static and Dynamic Clustering-based Methods", Memetic Computing, 07 March 2018 (Online). https://doi.org/10.1007/s12293-018-0254-9

4. This thesis proposes a novel decomposition-based multi-objective feature selection algorithm, in which two decomposition mechanisms (static and dynamic) based on multiple reference points are developed to address the characteristics of feature selection. The static multiple reference point-based mechanism helps to reduce the dependence of the decomposition on the true Pareto front shape and effect of the discontinuity; the dynamic one is able to detect regions in which the objectives are in conflict, and allocates more computation resources to the detected regions. In comparison with other EMO algorithms, the proposed decomposition approach evolves more diverse Pareto fronts with better performance. The dynamic mechanism successfully identifies conflicting regions and further improves the approximation quality for the Pareto fronts.

Part of this contribution has been published/submitted to:

Hoai Bach Nguyen, Bing Xue, Hisao Ishibuchi, Peter Andreae, and Mengjie Zhang. "Multiple Reference Points MOEA/D for Feature Selection". Proceedings of 2017 Genetic and Evolutionary Computation Conference (GECCO 2017). ACM Press. Berlin, German, 15 - 19 July 2017.pp 157-158

Hoai Bach Nguyen, Bing Xue, Hisao Ishibuchi, Peter Andreae, and Mengjie Zhang. "Multiple Reference Points based Decomposition for Multi-objective Feature Selection in Classification: Static and Dynamic Mechanisms". submitted to IEEE Transactions on Evolutionary Computation (under revise and resubmit).

5. This thesis proposes a new feature-based transfer learning method, where a new fitness function is developed to select a number of original features and shift source and target domains to be closer. Classification performance is used in the proposed fitness function to maintain the discriminative ability of the selected features in both domains and minimize the number of model assumptions. The results show that the proposed algorithm is able to select less than half of the original features, achieve better classification performance than using all features, and outperform the four state-of-the-art feature-based transfer learning algorithms. This is the first time EC is utilized to achieve feature selection-based transfer learning.

   Part of this contribution has been published in:

   Hoai Bach Nguyen, Bing Xue, and Peter Andreae. "A Particle Swarm Optimization based Feature Selection Approach to Transfer Learning in Classification". Proceedings of 2018 Genetic and Evolutionary Computation Conference (GECCO 2018). ACM Press. Kyoto, Japan, 15-19 July 2018. 8pp. (to appear)

1.5 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 presents the literature review. The six research objectives are presented in five contribution chapters from Chapter 3 to Chapter 7. Chapter 8 concludes the thesis.

Chapter 2 firstly introduces basic concepts of machine learning, classification, feature selection, information theory, evolutionary computation, multi-objective algorithms and transfer learning. Based on the discussion about the related feature selection work, open issues are identified, which form the thesis's motivation.

Chapter 3 presents a new filter-based feature selection algorithm based on mutual information estimation. The chapter firstly introduces more details about mutual information with its current challenges when applying to feature selection. The chapter then presents how to use mutual information estimation and PSO to address the challenges. A set of experiments are conducted on both synthetic and real-world datasets, and then the results are discussed and analyzed.

Chapter 4 presents a novel binary PSO, which can be applied to binary problems including feature selection. The chapter starts with a short introduction about binary PSO and its current limitations. The chapter then proposes the novel binary PSO and shows how the reformulated concepts can address the limitations. Experiments on two well-known traditional binary problems, i.e. knapsack and feature selection, are performed to compare the proposed binary PSO algorithm with other EC algorithms.

Chapter 5 presents a surrogate model for wrapper-based feature selection algorithm. The chapter shows how to build a small surrogate training set from the original one using a hierarchical clustering algorithm. A local search based on the surrogate training set is then described. Experiments are conducted on different datasets to examine the performance of the proposed surrogate model and compare it with using the original training set.

Chapter 6 presents a novel multi-objective feature selection algorithm, which is based on the MOEA/D (Multi-objective Evolutionary Algorithm based on Decomposition) framework [31]. The new decomposition mechanism using multiple reference points is presented, which is then followed by a dynamic reference point allocating mechanism to detect the conflicting regions. Experiments are conducted to compare between the proposed algorithm and other well-known EMO algorithms such as NSGAII (Non-dominated Sorting Genetic Algorithm II), SPEA2 (Improving the Strength Pareto Evolutionary Algorithm), OMOPSO and the standard MOEA/D algorithm.

Chapter 7 presents the first feature-based transfer learning algorithm using PSO. The chapter describes a new fitness function which mainly uses the classification accuracies to simultaneously reduce the differences between domain and maintain the discriminative ability. The fitness function is used with PSO to form a transfer learning algorithm, which is then examined and compared with other state-of-the-art transfer learning algorithms on different real-world problems.

Chapter 8 summarizes the work in the thesis and draws overall conclusions. Some possible future research directions are suggested in the chapter.

1.6 Benchmark Datasets

In this thesis, the proposed algorithms are evaluated on 12 datasets selected from the UCI Repository of Machine Learning Databases [32]. The selected datasets have different numbers of features (from 13 to 649), different numbers of classes (from 2 to 16), and different numbers of instances (from 178 to 2600). The selected datasets also have different feature types, i.e., real-value features, integer-value features, or mixture between both types. They are also from different real-world areas such as physics, finance, or medical area. The datasets are selected with an expectation

Table 1.1: Feature selection datasets.

Dataset	#Features	#Classes	#Instances	Feature Type	Area
Wine	13	3	178	Mixed	Physical
Australian	14	2	178	Mixed	Financial
Vehicle	18	4	846	Integer	Image
German	24	2	1000	Mixed	Financial
WBCD	30	2	569	Real	Medical
Ionosphere	34	2	351	Mixed	Physical
Sonar	60	2	208	Real	Physical
Hillvalley	100	2	606	Real	Graph
Musk1	166	2	476	Integer	Physical
Arrhythmia	279	16	452	Mixed	Medical
Madelon	500	2	4400	Integer	Artificial
Multiple Features	649	10	2000	Mixed	Digit Recognition

that they can be well representatives of real-world problems. More details about the selected datasets can be seen in Table 1.1.

Chapter 2

Literature Review

This chapter firstly provides background and important concepts of machine learning, feature selection, evolutionary computation (mainly PSO), multi-objective optimization (mainly MOEA/D) and information theory. It then reviews related work on traditional (non-EC) feature selection and EC-based feature selection. The review also presents limitations of existing work which form the motivations of this thesis.

2.1 Machine Learning

Machine learning is a major field of Artificial Intelligence, which aims to construct a system that is capable to learn from data rather than explicitly follows programmed instructions [33, 34]. By observing more examples/instances, a machine learning system is expected to automatically improve its performance for a certain task. There are different types of machine learning tasks: supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and transfer learning [35, 36]. The differences between these machine learning types are given as below:

- **Supervised learning**: In supervised learning, all instances or examples given to a machine learning system are labeled by desired out-

puts, which are known in advance. The task of supervised learning is to generate a function, which maps from an input to one of the known outputs [37]. Two well-known supervised learning tasks are classification and regression.

- **Unsupervised learning**: In unsupervised learning, the desired outputs are not known, which means that the instances are not labeled. Therefore, the goal of unsupervised learning algorithms is to extract a common pattern from the instances, which can be used to group similar instances together. Clustering is probably the most common task in unsupervised learning.

- **Semi-supervised learning**: Semi-supervised learning falls between supervised learning and unsupervised learning, where only a few labeled instances are provided while most instances are not labeled. The aim of semi-supervised learning is to extract useful information from both labeled and unlabeled instances, which is used to either correctly predict the class labels of unlabeled instances (transductive learning) or infer a mapping function from the inputs to the outputs (inductive learning).

- **Reinforcement learning**: In reinforcement learning, a machine learning system directly interacts with an environment via a sequence of actions. Each action will result in a reward or punishment based on feedback from the environment. The task is to achieve a certain goal by learning a sequence of actions with the best goodness.

- **Transfer learning**: In transfer learning, the main task is to reuse knowledge obtained from a source problem to improve the learning performance on a different but related problem (target problems). The two problems can have different learning tasks, different feature spaces, and/or different data distributions.

This thesis focuses on Supervised learning, particularly Classification, and Transfer learning.

2.1.1 Classification

Classification is one of the most important tasks in supervised learning, which aims to assign a given category (a class label) to an instance [38]. In a classification process, a classifier is needed to predict the class label of unseen instances. The classifier makes decisions based on values of features that describe the instances. The classifier is obtained by training a classification algorithm on a set of labeled instances. A classification problem is called a binary classification if there are only two class labels. When the number of class labels is more than two, the classification problem is known as a multi-class problem. An example of classification application is to predict breast cancer for a patient [39]. Based on the symptoms of people who are "negative" or "positive" in the cancer test, the classification algorithm is trained to capture the characteristics of patients having cancer. After that, the learned classifier can take the symptoms of a patient as an input to predict whether the patient has a breast cancer or not.

Training and testing

There are two main processes in a classification system: training and testing processes. During the training process, a classification algorithm is learned by using a set of instances, which is called a *training set*. The learned classifier is then evaluated on another set of instances which are unseen during the training process. The set of instances used in the testing process is called a *test set*. Each instance is described by a vector of feature values which can be numeric or categorical. The features have a significant effect on the learning time and the classification performance of the learned classifier.

In order to examine a classification algorithm, real-world datasets are

used as the benchmark problems. These datasets can be collected from public source for researches such as UCI Machine Learning Repository [32]. Each dataset is divided into two parts for training and testing purposes. The division should ensure that there is no overlapping between training and test sets.

Holdout [40] is one of the division methods, which divides the dataset into two disjoint subsets for training and testing processes. The sizes of these subsets follow an user-predefined proportion. However, in this method, all instances in the test set are not used for training the classification algorithm. It is not an efficient usage of datasets, especially when the number of instances in the dataset is too small. To overcome the limitation of holdout method, n-fold cross-validation was proposed [41]. In n-fold cross-validation, a dataset is randomly divided into n subsets (called "folds") with near-equal sizes. Note that the partitioning process ensures that the class distribution in each fold roughly remains the same as in the whole dataset. After that, each fold is then used for testing process exactly once while the rest of the dataset is used for training the classification algorithm. Consequently, the classification algorithm is trained n times, which results in n experiments with n different accuracies. The overall performance of the classifier is the average of the above n accuracies. In general, the larger the number of folds (n), the less bias the estimation of the classification performance, because a large number of instances are used in the training process. However, a large n might result in a high variance. Leave-one-out cross-validation (LOOCV) is a special case of n-folds cross-validation when n is set to the number of training examples. It means that, in LOOCV, each instance will be used as a test case exactly once and the number of times that the classification algorithm is trained, is equal to the number of instances in a dataset. Generally, by using n-fold cross-validation, all instances are used for both training and testing processes while the mutual exclusive condition between training and test set still holds. n-fold cross-validation is usually used when there are a small

number of instances in the whole dataset. One possible problem of n-fold cross-validation is the n repetition are not independent which might introduce bias. To address this problem, repeated n-fold cross-validation can be used. Particularly, the data is shuffled every repetition so that each fold has different data for different repetitions.

In classification, overfitting [42] is a common problem. The goal of classification is to achieve as high testing accuracy as possible. However, when there are too many parameters and the training phase does not involve any regularization pressure, the learned classifier starts remembering all characteristics of the training data. This memorization leads to very high training accuracy. However, there will be a risk that the learned model also fits with noisy instances in the training set. Consequently, due to the lack of generality, the learned model has a poor prediction ability, which results in a low testing accuracy. The phenomenon, in which the learned classifier performs well on the training set and badly in predicting the testing instances, is called overfitting. In contrast to overfitting, underfitting is another problem where the model to learn is too simple with too few degrees of freedom. Consequently, the learned model does not fit the data well enough which also leads to a poor performance on unseen/testing data.

Classification algorithms

- **K-nearest Neighbor Classifier (KNN)**

  KNN [43] is a type of instance-based learning approach, which simply remembers all the training instances instead of inducing any classification rule. The new instance is compared with all training instances to determine its class label. Firstly the distances between the new instance and every training instance are calculated. After that the K nearest training instances (neighbors) of the new instance are identified, where K is a user-predefined small integer number. The most popular class label among the K nearest neighbors is assigned

to the new instance. Many distances measures have been used in KNN, for example, Euclidean distance (continuous data), Manhattan (discrete data). KNN is considered a lazy-learning algorithm since its learning phase is very minimal. Although KNN is a simple learning algorithm, it performs well on many real-world problems. In addition, KNN is a nonparametric learning algorithm because it does not require any assumption about the probability distribution of the dataset.

A limitation of KNN is it is slow and requires a large memory, especially when the training set has a large number of features or instances. In addition, due to inducing no learning rule, KNN is sensitive to noise especially when K is small.

- **Decision Tree (DT)**

  DT [44] is one of the nonparametric classification algorithms in data mining, which can be applied to numeric, categorical or mixture data types. DT maps instances to class labels by building a tree-based prediction model. In a tree, each inner node, called a decision stump, corresponds to a single feature of the instance. The arc from an inner node is usually labeled by a value of the feature in the inner node. Each leaf of the tree is a class label. To classify an instance, its feature values will be compared with the decision stump in each inner node until reaching a leaf node. In order to build a tree, the most important step is to determine which feature and the feature's value (splitting point) should be used at each inner node. The most common strategy is a top-down greedy search to select the best feature for each inner node, which can split the source set into subsets with smallest impurities. Different DT algorithms use different metrics to measure the subset's impurity, for instance, C4.5 [45] uses information gain, CART [46] uses Gini Index, and CHAID [47] uses Chi-squared test.

  Each decision tree model can be seen as a set of "if-then-else" de-

cision rules, which makes it simple to understand and interpret. In addition, it is able to handle both numeric and categorical data. DT is also robust to noise and scale well with large datasets. However, due to having only one feature in each inner node, DT usually does not perform well when there are complex interactions between features [45].

- **Random Forest**

  In DT algorithms, when a tree grows too deep, it tends to overfit the training set. In this case, the tree simply models the noise in the training set rather than represents the relationship between inputs (features) and output (class label). Breiman [48] proposed a new classification algorithm, called random forest, which averages multiple decision trees to reduce the variance. Each random forest contains a set of decision trees, which are learned from different parts of the training set. In particular, for each decision tree, its own training set is constructed by randomly selecting instances with replacements from the original training set. In addition, the tree learning algorithm is also modified where at each inner node, a random subset of features is used instead of the original feature set. This process is also known as "feature bagging". After a number of decision trees are learned by using their own training sets, they are combined to form a random forest classifier. The classifier is then classified new/unseen instances by applying a voting scheme.

- **Support Vector Machines (SVMs)**

  In 1963, SVMs were originally proposed by Vapnik to solve binary classification problems. Recently, SVMs have been extended to adapt to multi-class problems [49]. The main goal of SVMs is to build or construct one or more hyperplanes to split a given dataset into multiple subsets corresponding to different class labels. There might be many hyperplanes that can split the data but the selected hyperplane

should maximize the distances with its nearest training instances (called functional margin).

In many practical problems, SVMs achieve good performance [50]. However, users usually have to provide a good kernel function for SVMs for non-linear cases. In addition, in terms of efficiency, SVMs have a high computational cost and require a large memory in the training phase when there is a high number of dimensions [51].

- **Bayesian Classifiers**

  In Bayesian classifiers, a probabilistic model is learned, which is then used to predict the class label of unseen data. Bayesian classifiers assume that relationships between features and the class label can be described in terms of probability distributions and the features are conditionally independent given the class label [34]. Based on the training set, Bayesian classifiers induce the conditional probability distribution of each feature given the class label (called likelihood) as well as the probability distribution of the class label (called prior). The two distributions are then used to calculate how likely an unseen instance belongs to each class [52]. One of the most common and straightforward Bayesian classifiers is Naive Bayes (NB). Although NB is an efficient classification algorithm, in many real-world problems, its assumption is violated due to the complex interactions between features.

2.1.2 Transfer Learning

Machine learning has witnessed a significant improvement during the past decades, which contributes to many areas such as classification, regression, and clustering. However, most learning algorithms assumes that the training set and the test set have an identical feature space with the same distribution. Therefore, any change in the feature space or the distribution leads to a requirement of rebuilding or retraining a learning model

using the newly collected training data, which might be difficult or even impossible task in many real-world applications. It would be desirable to reuse useful knowledge from an existing similar but not identical source domain to improve the performance of learning model in the target domain, which is the main motivation of *transfer learning* [36].

Transfer learning is inspired by the fact that human can utilize acquired knowledge learned in a specific problem to solve a similar problem in a faster and more efficient way [53]. For example, a person, who already learned to ride bicycles, can easily learn how to ride a motorbike. Transfer learning has been studied since 1995, which is also known by many different names such as learning to learn, knowledge transfer, inductive transfer or multi-task learning [36].

In transfer learning, there are three main questions to answer:

- **What to transfer?** Each domain has its own knowledge. On the one hand, some knowledge, which belongs to a specific domain, will not be useful in the other domain. On the other hand, some knowledge, which is common between different domains, is transferable to improve the performance on the target domain. Only the common knowledge should be transferred.

- **How to transfer?** After the common knowledge is discovered, a method needs to be developed to transfer this knowledge from the source domain to the target domain. There are many ways for transferring knowledge, such as instance-based transfer, feature-based transfer, which will be reviewed later.

- **When to transfer?** This is a difficult question which asks for in which situation knowledge should be transferred. If two domains have no relationship, knowledge from the source domain might not help or even cause worse performance in the target domain. However, discovering the relationship between two domains is usually a challenging problem, which is still an open issue. Currently, most trans-

fer learning algorithms assume that there should be a relationship between the domains.

Transfer learning approaches

In transfer learning, the task is to optimize performance on one domain (target domain), given extracted knowledge from a different domain (source domain), which is related to the target domain. The existing methods to achieve transfer learning problems can be categorized into four main types [36]:

- **Instance-based transfer learning**

  These approaches assume that there are some parts of source data that are usable with a few labeled target instances to improve the target performance. Most works in this category aim to assign weights to source instances so that they can match the target domain [54, 55].

- **Clustering-based transfer learning**

  These approaches achieve transfer learning by building a similarity graph between all instances and the weight on each edge represents the similarity between two instances [56, 57].

- **Parameter-based transfer learning**

  Parameter-based transfer learning methods can be applied when the models to learn in the source and target domains have some parameters in common, such as SVMs. The learned parameters from the source domain can be transferred to improve the learning performance on the target domain [58].

- **Feature-based transfer learning**

  Feature-based transfer learning approaches aim to find a good common feature representation, which simultaneously reduces the difference between the distributions on the source and target domains,

and maintains important information of the source and target data. The common feature representation may contain either some original features on both domains [59, 60] or newly built high-level features [61, 62].

2.2 Feature Selection

Feature reduction is an important preprocessing technique for classification. Feature reduction aims to reduce the dimensionality of a dataset while maintaining or even improving classification performance over using all features. There are two main feature reduction approaches which are feature selection and feature construction. This thesis focuses on feature selection.

2.2.1 Feature Selection

There are more than one feature selection definitions proposed by researchers, but most of them are similar in intuition and/or content [63]. These definitions are illustrated as below:

- Improving predictive accuracy: the aims of feature selection is to select a small number of features without leading to a significant reduction of prediction ability of the classifier learned from the selected features [63].
- Preserving conditional distribution: feature selection should find a subset of features such that the conditional distribution given the selected features is similar to the conditional distribution given all the features [63].
- Classical: Given n original features, the task of feature selection is to select a set of m features ($m < n$), which has the most optimal fitness with respect to an evaluation function, over all subsets of size m [64].

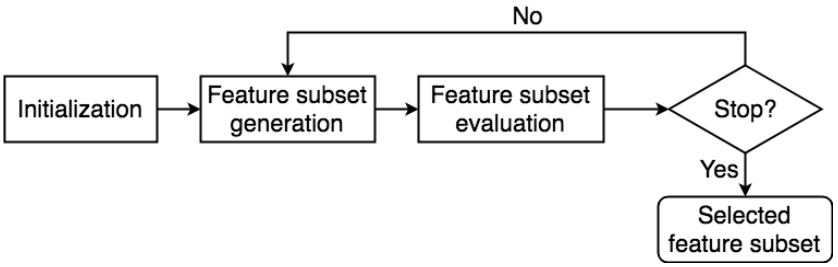


Figure 2.1: Feature Selection Process

- Idealized: feature selection aims to find the smallest subset of features, which is necessary and sufficient to describe the target concept [65].

In general, the goal of feature selection is to find a minimal subset of features which is necessary and sufficient to solve classification problems. This task is achieved by removing irrelevant and redundant features from the original feature set. By applying feature selection as a data preprocessing step, it is expected that the less complex dataset will help efficiently train a classifier, which is simpler, more efficient and accurate than using all features. As can be seen from the above definitions, feature selection has two main objectives: maximizing the classification performance and minimizing the number of selected features.

General feature selection process

In general, each feature selection algorithm has four main processes as shown in Figure 2.1 [66], which are illustrated as below:

- Initialization: In this phase, the feature selection algorithm is initialized according to the original features.

- Subset generation: In this step, one or several candidate feature subsets are generated using a searching mechanism. The starting point of the searching mechanism can be any feature subsets which can contain none of the original features, all of the original features or

some randomly selected original features. Based on that, the searching mechanism is expected to generate more promising candidate feature subsets in the following steps.

- Subset evaluation: The goodness of each generated feature subset is measured by an evaluation criterion, also known as fitness function. The fitness function guides the searching mechanism to explore promising regions in the search space. Therefore, designing a good fitness function is an essential task to any feature selection algorithm.

- Stopping criteria: In general, there are two main stopping criteria, which are based on generation procedure or the evaluation function. For instance, the feature selection process can stop when a predefined number of features are selected or the maximum number of iterations is reached. These are examples of generation procedure-based criteria. An example of evaluation function stopping criteria is when the best fitness value is not improved for a finite number of steps. When the stopping condition is satisfied, the best feature subset is returned as the final feature subset. Otherwise, the two phases: subset generation and subset evaluation are repeated.

After the final feature subset is generalized, the original dataset is transformed to the new dataset by removing unselected features. New training and test sets, which are generated from the new dataset, are fed into a learning algorithm to obtain training and testing accuracies, respectively. In a feature selection algorithm, the searching mechanism and the evaluation criteria are the most important components which significantly affect the quality of final feature subsets. The two components are discussed in more details below.

Searching mechanisms

Feature selection can be seen as an optimization problem, in which the search space is 2^n possible feature subsets where n is the total number of

features. More importantly, the interactions between features are complex. The two characteristics make feature selection a challenging problem; and it is important to develop an efficient search mechanism for feature selection. Currently, many searching mechanisms have been applied to find an optimal or near-optimal feature subset. These search techniques can be divided into the following categories [67]:

- Exhaustive search: Exhaustive searches consider all possible feature subsets, which ensures that an optimal subset will be discovered. An example of this exhaustive search space is FOCUS [68]. However, these search techniques are impractical when there are a large number of features due to their expensive computational costs.

- Sequential search: Sequential searches are heuristic searches, in which a finite steps of adding or removing features are performed to find an optimal feature subset. At each step, one or more features are either added or removed from the current feature subsets to form new subsets with better fitness values. These greedy searches perform more efficiently than the exhaustive searches but they do not guarantee to output an optimal subset. More importantly, sequential searches usually get stuck at local optima. Some examples of sequential searches are sequential forward selection [69], sequential backward selection [70].

- Evolutionary search: Evolutionary searches are also a kind of heuristic search. However, not like sequential searches, evolutionary searches have potential global search abilities which can cope with the large and complex search spaces of feature selection. Some examples of the evolutionary search are GP, PSO and GAs.

Evaluation criteria

Beside search strategies, evaluation criteria also play an important role in a feature selection algorithm. The searching for an optimal subset is to find

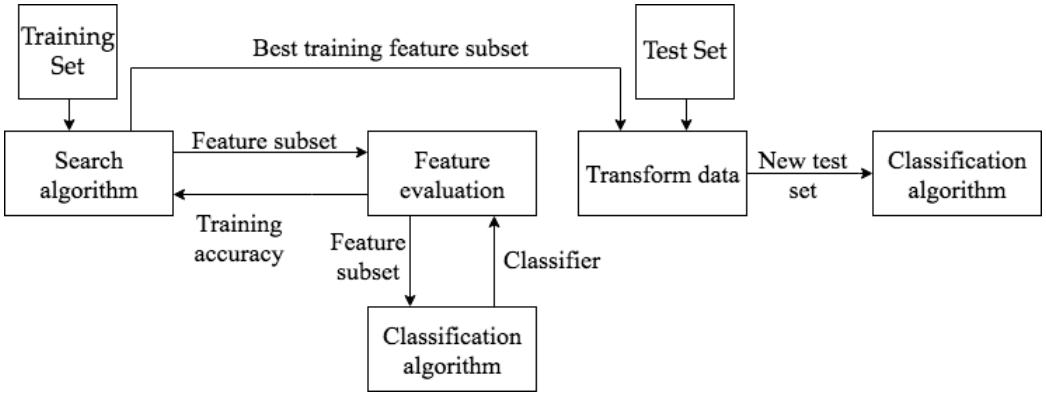
a subset with the best fitness value with respect to an evaluation function. In other words, fitness functions guide the search strategy to generate a new feature subset with better fitness value than the old one. Based on the evaluation approach, current feature selection algorithms can be divided into three types: filter, wrapper and embedded approaches [71].

- **Wrapper approach**

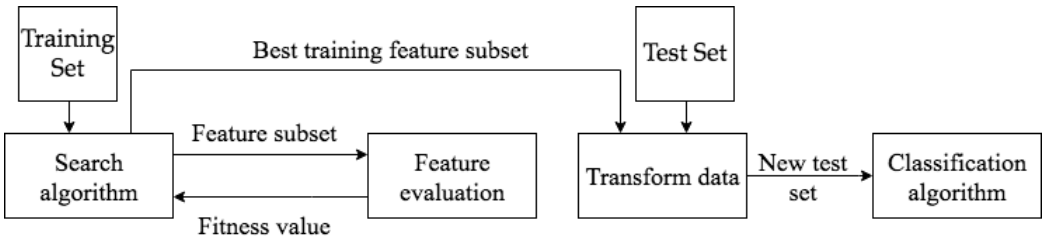
  Figure 2.2(a) [72] shows the diagram of a wrapper-based feature selection approach. In a wrapper approach, a classification algorithm is used to evaluate a candidate feature subset. The most discriminative feature subset is searching by maximizing the classification performance. Some common learning algorithms, used as a black box to evaluate the feature set, are KNN, DT, and SVM. In comparison with other methods, wrappers usually achieve better classification performance because they consider the direct interactions between the feature subset, the class, and the wrapped classification algorithm. However, the selected feature subset is optimized specifically for a learning algorithm. Therefore, a wrapper approach is less general than a filter approach. In addition, during the searching process, a classification algorithm is repeatedly trained to evaluate feature subsets, which usually causes a long computational time. Therefore, wrapper approaches are usually more expensive than other approaches.

- **Filter approach**

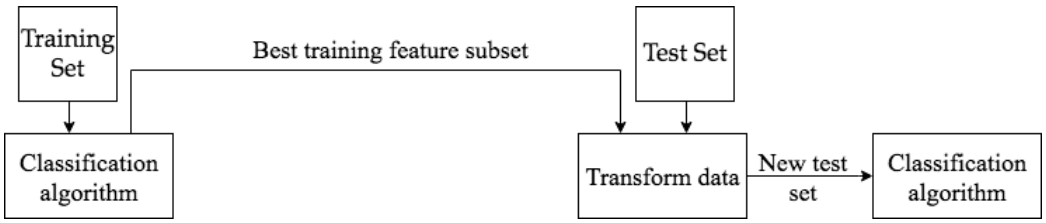
  Figure 2.2(b) [72] shows the diagram of a filter-based feature selection approach. In a filter-based method, instead of using the classification performance, a feature subset is evaluated based on intrinsic characteristics of a dataset. Because the feature subset is evaluated in an independent way of any learning algorithm, filters provide more general solutions than wrappers [73]. In addition, filters are also less computationally intensive than wrappers. However,



(a) Wrapper



(b) Filter



(c) Embedded

Figure 2.2: Feature Selection Categories

since these methods do not take into account the interaction between the selected feature subset and the learning algorithm, they usually achieve lower classification accuracies than wrappers [74]. In the past decades, different types of filter measures have been applied to feature selection.

- **Distance measure:** According to this measure, the more different between the conditional probabilities that instances belong to different class labels, the better the set of features. An example of this measure is ReliefF [75].
- **Correlation measure:** A correlation measure is used to evaluate how dependent two random variables are. A feature subset is good if it contains features highly correlated to the class, but uncorrelated to each other. This measure has been applied to achieve feature selection problems [21].
- **Consistency measure:** Consistency measure [20] attempts to find a minimum number of features that can separate classes as consistently as the full set of features. An inconsistency is defined when two instances having the same feature values but different class labels. The task is to find a minimal size subset of features that satisfies an acceptable inconsistency rate.
- **Information measure:** Mutual information is a well-known concept in information theory [76], which is used to measure the relevance between two or more random variables. Feature selection aims to select a feature subset, which is highly relevant to the class label and contains no redundant features. Therefore, a good feature subset should have high mutual information with the class label and low mutual information between inner features. Because mutual information can detect non-linear feature interactions easier than other measures, it is gaining more attention in the feature selection community recently.

- **Embedded approach**

  In an embedded approach, feature selection is embedded into the training process of a classification algorithm. Once the classification algorithm is trained, the features used in the classifier are selected. In comparison with wrappers, embedded approaches are more efficient and may still maintain a good classification performance. DT is an example of the embedded-based feature selection approach. Particularly, features used in the final tree are considered a good feature set. SVM can also be used in an embedded-based feature selection approach where each weight in the learned SVM model can be considered as the importance of the corresponding feature. Figure 2.2(c) shows the diagram of an embedded-based feature selection approach.

2.2.2 Feature Construction

In many difficult real-world problems, classification algorithms, especially symbolic algorithms such as decision trees, cannot be trained to be a good classifier using the set of individual features [77]. Feature selection might help to induce a good feature subset, which contains only relevant features without any redundancy. However, inducing the relationship between single features to help improve the classification performance is also a challenging problem. This problem can be partially solved by feature construction. Feature construction aims to create new high-level features, which can be seen as a function of some original features; the new features are expected to improve the classification performance [78]. A feature construction system is similar to a feature selection system, which usually aims to transform the original representation space into a lower-dimension one to improve the classification performance. In general, a feature construction process also has four steps:

- Initialization: In this phase, the feature construction algorithm is ini-

tialized.

- High-level features generation: In this step, a number of new high-level features are constructed based on the original feature space. The key challenge is to design which original features and which operators are used in the constructing process. It is important to select a good set of operators which are suitable for the feature set. In addition, the number of new-high level features is also an important parameter.

- Feature evaluation: Similar to feature selection, the new high-level features are evaluated to guide the searching process. In the evaluation process, it is important to consider the interactions between new features, which ensures that the new features can cooperate to produce a good classification performance without any redundancy.

- Stopping criteria: The feature construction process output the final set of new high-level features when the stopping condition is met.

Both feature selection and feature construction can reduce the number of features. However, feature selection preserves meanings of the original features, while feature construction builds new high-level features which might be difficult to interpret. Therefore, this thesis focuses on feature selection.

2.3 Evolutionary Computation

Evolutionary Computation (EC) is a sub-field of artificial intelligence, which is inspired by principles of biological evolution. Most EC algorithms maintain a population, which consists of a set of candidate solutions. The population is maintained and evolved by an updating mechanism to search for better solutions according to a predefined fitness function. In general,

EC methods can be divided into two main categories: evolutionary algorithms and swarm intelligence.

2.3.1 Evolutionary Algorithms (EAs)

EAs refer to the evolutionary algorithms which follow Darwinian principles. In particular, these algorithms apply genetic operators such as mutation, crossover, reproduction, and selection to evolve a population of individuals. The individuals compete to survive based on their fitness values. There are four main evolutionary algorithms:

- **Genetic Algorithms (GAs) [79]**

  GAs were probably the first EA. In GAs, each individual is represented by a fixed-length vector, which is usually called a chromosome. The vector entries can be either real numbers, integers or bit values. The evolution process is performed by four genetic operators: selection, mutation, crossover, and reproduction.

- **Evolutionary Strategies (ES) [80]**

  In terms of representation, ES is similar to GAs where each individual is represented by a fixed-length vector. The main difference is ES mainly relies on the mutation operator to evolve its population. In addition, the selection in ES is deterministic.

- **Genetic Programming (GP) [81]**

  GP can automatically evolve a computer program to solve a specific task. In GP, each candidate program is represented by a variable-length tree-based representation. Because of the flexible representation, GP has been widely applied to a wide range of real-world problems. The evolution process of a population in GP is similar to GAs, which are mainly based on mutation and crossover operators.

- **Evolutionary Programming (EP) [82]**

EP is used mainly to evolve finite state machines (FSM) where each FSM is a program. Therefore, the representation of an individual in EP is similar to GP except for the program's structure is fixed. The main genetic operator in EP is the mutation operator.

2.3.2 Swarm Intelligence (SI)

SI algorithms are inspired by the behaviors of social insects. In these algorithms, a population consists of a set of individuals, which parallelly explore the search space. The individuals then share their knowledge about the search space to other members. The sharing mechanism helps the whole swarm move toward the better positions in the search space, which eventually converges to an optimum [83]. Particle Swarm Optimization, Ant Colony Optimization, and Differential Evolution are three well-known SI algorithms.

- **Particle Swarm Optimization (PSO) [7]**

  In PSO, the set of individuals is called a swarm. Each individual maintains its best position and its neighbors' best position. The two bests are expected to guide each individual towards better positions.

- **Ant Colony Optimization (ACO) [84]**

  ACO is inspired by the foraging behavior of ant colonies which aims to find the shortest path from their colony to a food source. The candidate solutions are called ants in ACO. During the searching process, each ant deposits an amount of pheromone on its favorite path which attracts other ants to follow this path. The "best" solution is the path has the largest amount of pheromone.

- **Differential Evolution (DE) [85]**

  DE was proposed by Storn and Price in 1995 [85]. DE is different from PSO and GAs regarding its mechanism to generate new candidate solutions. In DE, the population evolves by three operators:

mutation, crossover, and selection. Although the three operators have the same names as those in GAs, their functionalities are very different. Not like GAs, all DE individuals or vectors are involved in the evolutionary process. The mutation operator builds a new vector, called mutant, by combining three random vectors (excluding the target one) that are not necessarily the best ones as in PSO. The crossover operator then builds a trial vector by setting each vector entry to an entry value selected from either the mutant vector or the target vector. The selection process selects the better one among the trial vector and the target vector.

Apart from the mentioned EC methods, there are also other popular EC methods such as Learning Classifier System (LCS) [86], and Artificial Immune Systems (AIS) [87].

2.3.3 Particle Swarm Optimization

PSO [30] is an evolutionary computation method, which is inspired by the social behaviors of bird flocking. In PSO, a problem is optimized by using a population of particles, called a swarm. In order to find the optimal solution, each particle moves around the search space by updating its position and velocity. Particularly, the current position of particle is represented by a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimensionality of the search space. These positions are updated by using another vector, called velocity $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$, which is limited by a predefined maximum velocity, v_{max} and $v_{id} \in [-v_{max}, v_{max}]$. During the searching process, each particle maintains a record of its best position, called $pbest$, and its neighbors' best position, called $nbest$. If each particle shares its information with all other particles, all particles have the same $nbest$, which is usually called $gbest$. The position and velocity of each particle are updated according to the following equations:

$$v_{id}^{t+1} = w \times v_{id}^{t} + c_1 \times r_{i1} \times (pbest_{id} - x_{id}^{t}) + c_2 \times r_{i2} \times (gbest_{id} - x_{id}^{t}) \quad (2.1)$$

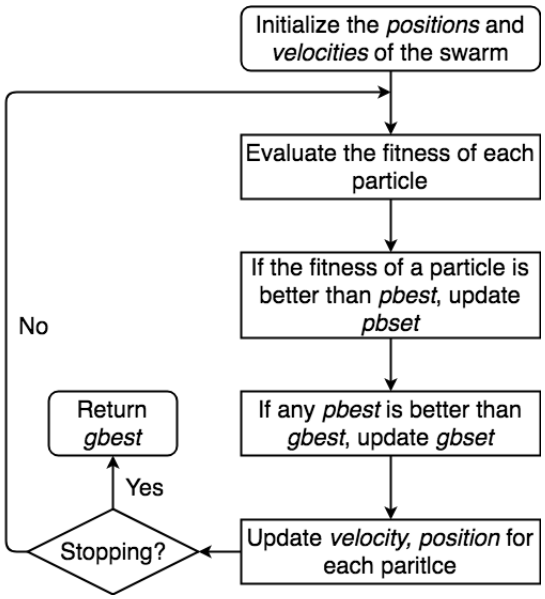


Figure 2.3: Flowchart of PSO

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \tag{2.2}$$

where t denotes the t^{th} iteration in the search process, d is the d^{th} dimension in the search space, i is the index of particle, w is inertia weight, c_1 and c_2 are acceleration constants, r_{i1} and r_{i2} are random values uniformly distributed in [0,1], $pbest_{id}$ and $gbest_{id}$ represent the position entry of $pbest$ and $gbest$ in the d^{th} dimension, respectively. The general process of PSO is given in Figure 2.3.

Binary PSO (BPSO)

PSO was originally developed to address continuous optimization problems. Therefore, both position and velocity are vectors of real numbers. However, this representation is not suitable for problems having discrete search spaces. For example, a feature selection problem has a discrete search space, more specifically a binary search space, where a feature is either included or excluded in a feature subset. In 1997, Kennedy and Eberhart [88] proposed binary particle swarm optimization (BPSO). In BPSO,

the position of each particle is a vector of binary numbers, which are restricted to either 0 or 1. Each entry v_i of the velocity is used to calculate the probability that the corresponding element in the position vector, x_i, taking value 1. The velocity of each particle is still updated by using Eq. (2.1). A new equation (2.3), which uses a sigmoid function, is applied to update the particle position.

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

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

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

BPSO has been applied to many real-world problems. Sarath et al. [89] applied BPSO to generate association rules from transactional datasets. The results on a dataset from an Indian commercial bank showed that applying BPSO not only provided higher quality rules but also avoided redundant rules. Taha et al. [90] used BPSO to detect available frequencies in cognitive radio, which allowed to utilize system resources. The BPSO-based dynamic allocation achieved higher detection rates and lower false alarm rates with different noise ratios. Lin et al. [91] used BPSO to search for highly profitable item sets instead of frequent item sets in transactional databases. It was shown that the BPSO-based algorithm was more efficient, more effective and converged faster than GAs.

In many other works, BPSO has also been modified to improve its performance. In the original BPSO [29], a sigmoid-function was used as a transfer function known as a S-shaped function. In [92], V-shaped transfer functions were proposed for BPSO. The position updating equation also considered the previous location. The experimental results showed that applying V-shaped functions while considering the previous location improved the performance of BPSO. However, it was not clear that which

modification contributed more to the improvement. The performance of BPSO on different datasets heavily depended on the specific transfer functions, even when they were from the same family (V-shaped or S-shaped). Hence, it was not an easy task to select an appropriate transfer function for a particular task or dataset.

Zhang et al. [93] used BPSO for feature selection, which aimed to improve the performance in spam detection problems. A mutation operator was used to increase the diversity of the swarm and avoid premature convergence. The experimental results indicated that the proposed algorithm achieved better results than GAs and the standard PSO algorithm. Yang et al. [94] attempted to allocate workload to sensors in a network so that the system was more energy-efficient and the communication volume was reduced. The BPSO algorithm with a V-shape function, a new updating equation, and mutation operators, was proposed. The BPSO algorithm outperformed GAs and standard BPSO. However, the modifications in the two proposed BPSO algorithms still ignored the previous location. Aiman et al. [95] modified BPSO to achieve a good state assignment on a finite state machine, which aimed to minimize the area of sequential circuits. The authors proposed an updating equation for velocity, which considered the correlation between $pbest$ and $gbest$. Although the experimental results showed that the modified BPSO algorithm was more effective than standard BPSO and GAs, it has potential that the swarm would be stuck at local optima when $pbest$ and $gbest$ had the same position. Zhai et al. [96] improved a BPSO-based instance selection algorithm by utilizing an immune mechanism. Before updating the swarm, some particles were modified based on the numbers of their good neighbors (vaccine value) and current positions. The experimental results on datasets with different numbers of instances showed that the immune BPSO outperformed both standard BPSO and INSIGHT, which was a deterministic instance selection algorithm, in terms of the classification accuracy and the size of instance subsets. It could be seen that considering the previous position

in BPSO has a positive effect. However, the proposed algorithm was designed specifically for instance selection.

In 2016, Liu et al. [97] provided an analysis about the effect of the inertia weight parameter on the searching ability of BPSO. Particularly, it was shown that when the two bests were not changed, a larger inertia weight tended to enhance the exploitation ability, which was opposite to continuous PSO (CPSO). Based on this observation, an incremental strategy for the inertia weight was defined as follows:

$$w = \begin{cases} \underline{w} + \frac{\pi \times (\overline{w} - \underline{w})}{\rho \times \overline{\pi}}, \textbf{ if } \pi \leq \rho\overline{\pi} \\ \overline{w}, \textbf{ if } \rho\overline{\pi} < \pi \leq \overline{\pi} \end{cases} \tag{2.5}$$

where π and $\overline{\pi}$ were the current iteration and the maximum number of iterations, $\overline{w}$ and $\underline{w}$ were the upper and lower boundaries of w, ρ was a parameter in the interval [0,1], which was used to determine the number of iterations to increase w from $\underline{w}$ to $\overline{w}$. In the first $\rho \times \pi$ iterations, the inertia weight was linearly increased from $\underline{w}$ to $\overline{w}$. The inertia weight w was not changed in the following iterations.

Based on the analysis, Liu et al. [97] proposed a BPSO algorithm named Up BPSO, which outperformed the standard BPSO algorithm with a constant and a linearly decreasing inertia weight. Up BPSO was already applied to achieve web-service allocation [98], which aimed to minimize network latency and cost to deploy servers.

Existing works demonstrate the need of BPSO to solve binary problems. However, BPSO still has limited performance in comparison with continuous PSO [99], which requires additional modifications for BPSO to improve its performance. Despite the fact that many modifications have been made, not much research considers the core reason which is the inappropriate application of continuous velocity and momentum concepts to BPSO.

2.4 Multi-objective Optimization

In a multi-objective problem, two or more conflicting objectives are optimized simultaneously. An o-objective minimization problem can be written as follows.

$$\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_o(\mathbf{x})) \tag{2.6}$$

$$\text{subject to } g_i(\mathbf{x}) \leq 0, i = 1, 2, ..., k$$
$$h_i(\mathbf{x}) = 0, i = 1, 2, ..., l$$

where **f(x)** is a vector of objectives, f_i(**x**) is the i^{th} objective, **x** is a decision vector, $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are the constraint functions of the problem.

The quality of a solution is determined by the trade-off between the objectives. Specifically, a solution **y** is better than a solution **z** if and only if:

$$\forall i\text{: } f_i(\mathbf{y}) \leq f_i(\mathbf{z}) \text{ and } \exists j\text{: } f_j(\mathbf{y}) < f_j(\mathbf{z}) \tag{2.7}$$

It can be said that **y** dominates **z**. If a solution is not dominated by any other feasible solutions, the solution is called a Pareto optimal solution. The set of all Pareto optimal solutions form a trade-off surface in the objective space, which is called the true Pareto front. The task of any multi-objective algorithm is to evolve a set of well-distributed non-dominated solutions, which is a good approximation of the true Pareto front. Feature selection can be considered a two-objective minimization problem, in which the number of features and the classification error rate need to be minimized.

2.4.1 Evolutionary Multi-objective Algorithm (EMO)

As a family of population-based optimization algorithms, EC is widely applied to solve multi-objective problems since it can naturally evolve a set of non-dominated solutions. The EC-based multi-objective algorithms are

called Evolutionary Multi-objective Optimization Algorithms (EMO). Basically, EMO can be divided into three main categories: Pareto dominance-based algorithms, indicator-based algorithms, and decomposition-based algorithms [100].

Pareto dominance-based algorithms

The main idea of these algorithms is to sort their populations based on Pareto dominance. The obtained ranking is used as the main part of a fitness function. Typical Pareto dominance-based algorithms can be seen as below:

- **Non-dominated Sorting based Multi-objective Genetic Algorithm II (NSGA-II) [101]**

  NSGA-II follows the same principle as GAs, which applies genetic operators to generate offspring from parents. NSGA-II uses a fast non-dominated sorting technique to rank the union of generated offsprings and parents to different levels of non-dominated solutions. The non-dominated solutions in the same level are then further ranked according to their crowding distances, which is expected to maintain the diversity of the population. The crowding distance of a point is the average distance between two closest points on both sides of the point along each objective. The next population is then filled with the top-ranked solutions.

- **Strength Pareto Evolutionary Algorithm 2 (SPEA2) [102]**

  Different from NSGA-II, SPEA2 maintains a set of non-dominated solutions called an archive set. SPEA2 assigns fitness values to each individual in both archive set and the population. The fitness function contains two parts: *raw fitness* (the strength of a candidate solution) and *a density measure* (distance from the candidate solution to its k^{th} nearest neighbor). During the evolutionary process, one or more

candidate solutions can be added or removed from the archive set according to the fitness values or the density measure, respectively.

- **Multi-objective PSO (MOPSO) [103]**

  MOPSO follows the main ideas of PSO except for how to determine $gbest$. The reason is that in a multi-objective problem, instead of a single best solution there is a set of non-dominated solutions. OMOPSO [104], one of the most well-known MOPSO algorithms, maintains an archive to store the non-dominated solutions. A crowding distance is used to select an archive member as a $gbest$ for each particle.

Decomposition-based algorithms

These algorithms transform a multi-objective problem into several single-objective sub-problems. The final set of non-dominated solutions can be obtained by combining solutions from all sub-problems. MOEA/D (Multi-objective Evolutionary Algorithm based on Decomposition) [31] is a representative of decomposition-based EMO algorithms. Standard MOEA/D decomposes a multi-objective algorithm using a set of weight vectors. Each single-objective sub-problem has its corresponding weight vector **w**, which is used to build a fitness function. The task in the sub-problem is to find an optimal solution with respect to the fitness function. Each sub-problem has one candidate solution, so the number of decomposed single-objective sub-problems is equal to the population size. Each sub-problem has other T sub-problems as its neighbors. The neighborhood relations are defined by the distance between the weight vectors. It is expected that the solutions of neighboring sub-problems should be similar so that each sub-problem can improve its solution by using information from its neighbors.

Given an o-objectives problem, each weight vectors has o elements, **w** = $(w_1, w_2, ..., w_o)$, which satisfies the following conditions

$$\sum_{i=1}^{o} w_i = 1 \text{ and } w_i \in \left\{0, \frac{1}{M}, \frac{2}{M}, ..., \frac{M}{M}\right\} \tag{2.8}$$

where M is a predefined positive integer. For two-objective problems, M is the number of sub-problems, which is also the population size [31]. There are various ways to aggregate multiple objectives into a single scalar function, in which Weighted Sum, Tchebycheff, and Penalty-based Boundary Intersection are three common approaches [105].

A good set of weight vectors can generate a good approximation of the Pareto front, but defining an appropriate set of weight vectors is a difficult task in MOEA/D since it depends strongly on the shape of the true Pareto front [106]. Many attempts have been made to adjust weight vectors dynamically during the evolutionary process to cope with different complicated Pareto front shapes [107, 108]. However, most approaches depended on the true Pareto front consisting of continuous regions (if not being fully continuous). It is not clear how they can be made to work on problems with highly discontinuous or discrete Pareto fronts, which is a characteristic of multi-objective feature selection.

Indicator-based algorithms

The idea of these algorithms is to use an indicator measure to evaluate the quality of a set of non-dominated solutions. One of the most well-known indicators is hypervolume [109], which is used in the S Metric Selection Evolutionary Multi-objective Optimization Algorithm (SMS-EMOA) [110]. However, indicator values are usually computationally intensive, which makes indicator-based algorithms less popular in comparison with the other two EMO categories.

2.5 Information Theory

2.5.1 Basic Concepts

Entropy, one of the core concepts in information theory [111], is used to measure the uncertainty or the amount of information of a random vari-

able. Given X is a discrete variable, its entropy can be calculated by the following formula:

$$H(X) = -\sum_{x \in X} P(X = x) \times \log_2 P(X = x) \tag{2.9}$$

Entropy can be extended to measure the uncertainty of a joint variable, which consists of more than one random variable. The joint entropy can be defined as:

$$H(X_1, \ldots, X_n) = -\sum_{\substack{x_i \in X_i \\ i=1 \ldots n}} p(x_1, \ldots, x_n) \times \log_2 p(x_1, \ldots, x_n) \tag{2.10}$$

where $p(x_1, \ldots, x_n) = P(X_1 = x_1, \ldots, X_n = x_n)$

Mutual information is another important concept in information theory. Mutual information is used to calculate the common information between two random variables. Mutual information is a symmetric measure, which is defined by the following formula:

$$\begin{aligned} MI(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= \sum_{x \in X, y \in Y} p(x,y) \times \log_2 \frac{p(x,y)}{p(x) \times p(y)} \end{aligned} \tag{2.11}$$

where $p(x, y)$ is the joint *probability distribution* function. According to Eq. (2.11), if X and Y are totally independent, which means $p(x, y) = p(x) \times p(y)$, then the mutual information between X and Y becomes 0. On the other hand, if there is a strong relationship between X and Y then $MI(X;Y)$ will be large. If X and Y are two continuous variables, mutual information is extended by replacing the summation by a definite double integral as below:

$$MI(X;Y) = \int_X \int_Y p(x,y) \times \log_2 \frac{p(x,y)}{p(x) \times p(y)} \, \mathrm{d}x \, \mathrm{d}y \tag{2.12}$$

where $p(x), p(y)$ and $p(x, y)$ are *probability density* functions.

Mutual information is also extended in many ways to measure the common information between more than two random variables. Suppose

that S is a joint variable, which consists of m single variables. Multi-variate information (MvI) or interaction information is used to measure the common between all variables' information. The interaction information of a joint variable S = $\{s_1, \ldots, s_m\}$ is calculated using Eq. (2.13).

$$MvI(S) = -\sum_{U \subseteq S} (-1)^{|S|-|U|} H(U) \tag{2.13}$$

There is also another extension of MI, called "total correlation information" (TCI) [112], which measures the common information between any variable subsets of S. TCI can be computed by using the following equation:

$$TCI(S) = \sum_{s_i \in S} H(s_i) - H(s_1, s_2, \ldots, s_m) \tag{2.14}$$

where m is the total number of single feature/variable (s_i) in the joint feature/variable (S).

2.5.2 Mutual Information Calculation

In order to calculate the mutual information between two or more variables/features, it is necessary to know the probability distribution of each variable as well as the joint probability distribution. The trivial way is to derive the distributions by counting instances with each possible value of the features, which is known as the counting approach. Although this approach is efficient, it only works on discrete datasets.

Mutual information estimation is another approach to calculate mutual information. The oldest and simplest estimator is the "basic histogram" [113], in which each dimension corresponding to one variable is divided into many non-overlapping bins with a fixed size. The probability distribution of each "bin" is calculated as a ratio between the number of observations falling into the bin and the total number of observations. Therefore, each bin is considered a possible value of a single variable or a joint

variable. The entropy of each single/joint variable can be calculated by applying the discretized version given in Eq. (2.9) and then the mutual information can be acquired according to the formula Eq. (2.11). In this approach, there are two most important parameters, which are the number of bins and the bin's size.

The basic histogram is sensitive to the parameter selections. In addition, histogram approaches have sharp boundaries, which means that two similar instances on different sides of the boundary are considered different values. To avoid this discontinuity, Parzen et al. [114] proposed kernel density estimation (KDE). This approach estimated the probability density of each instance with a kernel function Θ, which is shown in Eq. (2.15).

$$\hat{p}(S_i) = \frac{1}{N} \times \sum_{i'=1}^{N} \Theta\left(|S_i - S_{i'}| - r\right) \tag{2.15}$$

where Θ is the kernel function and r is the kernel width, $|.|$ is a norm and N is the total number of instances.

The kernel function Θ measures the similarity between two instances of feature set S, S_i, and $S_{i'}$. Normally, the Θ is a step function, which means that $\Theta(X > 0) = 0$ and $\Theta(X \leq 0) = 1$. The norm $|.|$ is the maximum norm. Therefore, the probability estimated by Eq. (2.15) is the proportion of the N instances, whose distances to the instance S_i are less than r. The entropy of the joint variable or feature subset S is then achieved by averaging the local entropy of all instances, which can be seen in Eq. (2.16). The calculated entropies are plugged in Eq. (2.11) to derive the mutual information estimation.

$$\hat{H}(S) = \frac{1}{N} \times \sum_{i=1}^{N} -\hat{p}(S_i) \times \log \hat{p}(S_i) \tag{2.16}$$

Besides KDE, recently Kraskov et al. [115] proposed another estimation approach, called Nearest Neighbor Estimation (NNE). Similar to KDE, NNE also works on each instance. The main idea of NNE is if neighbors

of an instance on two dimensions X and Y are similar, then there must be a strong relationship between X and Y. Particularly, for each instance i, K nearest neighbors of the instance are found to derive the distance $\epsilon(i)$, which is then used as a boundary to define the neighbors of the instance on each dimension (feature). The boundary $\epsilon(i)$ is twice the distance from the i^{th} instance to its K^{th} nearest neighbor. On each dimension j, the number of instances whose distances to the i^{th} instance is smaller than $\epsilon(i)$ is counted. The resulting number is denoted as n_{ij} which is used to estimate the mutual information using Eq. (2.17).

$$\hat{MI}(S) = \psi(k) - \frac{m-1}{K} + (m-1) \times \psi(N) - \frac{1}{N} \times \sum_{i=1}^{N} \sum_{j=1}^{m} n_{ij} \qquad (2.17)$$

where ψ is a digamma function, m is the number of single variables (features) in the variable (feature) set S.

Therefore, NNE can be seen as an improvement of KDE, where the boundary *r* is dynamically determined by the number of nearest neighbors *K*. Both estimators are implemented in Java Information Dynamics Toolkit (JIDT), an information-theoretic toolkit developed by Lizier et al. [116].

2.6 Traditional Techniques (non-EC) for Feature Selection

A basic version of feature selection is feature ranking [66], where a score is assigned to each feature according to an evaluation criterion. Feature selection can be achieved by selecting the features with the highest scores. However, this type of algorithm ignores the interactions between features. Additionally, the features with the highest scores are usually similar. Therefore, these algorithms tend to select redundant features.

Sequential search techniques have been applied to solve feature selection problems. In particular, sequential forward selection (SFS) [69] and

sequential backward selection (SBS) [70] are proposed. At each step of the selection process, SFS (or SBS) adds (or removes) a feature from an empty (full) feature set. Although these local search techniques often achieve better performance than the feature ranking method, they might suffer from the "nesting" problem, in which once a feature is added (or removed) from the feature set, it cannot be removed (or added) later. In order to avoid "nesting" effect, Stearns et al. [117] proposed a "plus-l-takeaway-r" method in which SFS was applied l times forward and then SBS was applied for r backtracking steps. However, it is challenging to determine the best values of (l,r). This problem is addressed by sequential backward floating selection (SBFS) and sequential forward floating selection (SFFS), proposed by Pudil et al.[118]. In SBFS ad SFFS, the values (l, r) are dynamically determined rather than being fixed in the "plus-l-takeaway-r" method. Later, Nakariyakul et al. [119] propose a method named IFFS which further improves the sequential searches using an additional step replacing weak features in the current feature subset by an unselected feature. The experimental results show that IFFS achieves better performance than other sequential search algorithms. Mutual information is also used with sequential searches to achieve feature selection [120, 121, 122].

Another approach is to use regularization models for feature selection. This can be considered an embedded-based feature selection approach since when the model is trained as a classifier to minimize the classification error, some of its coefficients are forced to be very small or to be exactly zero. Feature selection can be achieved by selecting features with nonzero coefficients [123, 124]. Particularly, given a dataset contains N instances {**X,Y**}, where **X** = {$x_1, x_2, ..., x_N$} is the feature values of the N instances and **Y** = {$y_1, y_2, ..., y_N$} is the class labels of the N instances. The task of regularization-based feature selection is to find the optimal weight vector **w** which minimizes the following fitness function:

$$Fitness(\mathbf{w}) = loss(\mathbf{w}, \mathbf{X}) + \alpha \times penalty(\mathbf{w}) \tag{2.18}$$

where $loss(.)$ is the classification error and $penalty()$ is the regularization

term which forces some weights in $\mathbf{w}$ to be close to 0. Some common choices of $loss(.)$ are *quadratic loss*, *hinge loss*, and *logistic loss*. The $penalty(\mathbf{w})$ can use the ℓ_1 form, which is $penalty(\mathbf{w}) = \sum_{i=1}^{N} |\mathbf{w}_i|$ [125]. These methods gain increasing attention because of its good performance [126, 127]. However, since these methods select features based on the feature's weight, they are likely to miss the interactions between features.

2.7 EC-based Feature Selection

EC techniques have been widely applied to feature selection because of their global search ability. More importantly, EC algorithms have an advantage of coping with either single-objective problems or multi-objective problems, which is the case for feature selection.

2.7.1 Single-objective Feature Selection

Single-objective feature selection is achieved by many EC algorithms such as GAs, PSO, GP or DE. Since this thesis focuses on applying PSO to achieve feature selection, the PSO-based feature selections are reviewed first, then other EC-based feature selection algorithms are reviewed later.

PSO for feature selection

Both continuous and binary PSO have been widely applied to wrapper-based or filter-based feature selection. In general, each particle's position is represented by a vector, in which each entry corresponds to an original feature and indicates whether that feature is selected or not. In continuous PSO, each entry is a real number, which falls in the range [0,1]. A threshold θ is used to determine whether or not a feature is selected. In particular, if the entry's value is greater than θ, the corresponding feature is selected. Otherwise, the corresponding feature is not selected. Binary PSO has more

natural representation, in which each entry is either 0 or 1. A feature is selected if and only if its corresponding bit value is 1.

- **PSO for wrapper-based feature selection**

  In wrapper-based feature selection approaches, different classification algorithms were used to evaluate the solution's fitness, for instance SVMs, KNN, ANNs. Many ideas have been proposed to improve the performance of PSO-based feature selection algorithms. These ideas included modifications in the initialization strategy, representation, fitness function or searching mechanisms. In [128], Xue et al. proposed three new initialization mechanisms, which mimic the sequential feature selection approach. While the small initialization used about 10% of the original features to initialize the particles, particles in the large initialization were built based on 50% of original features. These two initialization mechanisms were combined in the mixed initialization, which used the small initialization for most of particles and the large initialization for the rest. In addition, three new updating mechanisms for $pbest$ and $gbest$ were proposed in the paper. The experimental results showed that the new initialization and updating mechanisms led to smaller feature subsets with better classification performance than the standard PSO and the two-stage binary PSO algorithm [129, 130].

  In [131], Vieira et al. proposed a new representation for binary PSO, which simultaneously performed feature selection and optimized the SVM kernel parameters. Particularly, each entry corresponded to an original feature or a kernel parameter, which resulted in the length of the new representation was equal to the total number of features and kernel parameters. Experimental results showed that the proposed algorithm achieved better classification performance than other binary PSO-based feature selection algorithms [132, 133] and selected smaller feature subsets than a GA-based feature selec-

tion algorithm [134]. This representation was also applied in continuous encoding [135] and a mixture of binary and continuous encoding [136]. Lane et al. [137] applied statistical clustering, which grouped similar features into one cluster. In particular, the proposed method arranged the features in the same cluster together and selected a single feature from each cluster. The results showed that the proposed algorithms significantly reduced the number of selected features, which was equal to the number of clusters. Lane et al. [138] further improved his work by applying Gaussian distribution to select more than one features from each cluster. Particularly, a Gaussian distribution was used to determine the number of selected features (m) from each cluster and m features with the highest velocity in a certain cluster were selected. Later, Nguyen et al. [139] also applied statistical clustering to propose a new representation, which had lower dimensionality than the traditional representation. Particularly, a maximum number of selected features from each cluster, which was smaller than the total features from the cluster, was determined. Each position entry belonged to a certain cluster and presented a feature index from the cluster. The experimental results indicated that the proposed algorithms achieved better classification performance and selected a smaller number of features than two other PSO-based algorithms. However, in the new representation a small change of the position might not lead to any different feature subset. Therefore, Nguyen et al. [140] applied Gaussian distribution to propose a new transformation rule, which could form a smoother fitness landscape than the representation in [139].

Premature convergence is a typical problem of PSO, in which the swarm is stuck in local optima. In order to avoid this problem, Chuang et al. [132] proposed a $gbest$ resetting mechanism, which set all $gbest$ position's elements to zero when the best fitness did not change for a finite number iterations. The experimental results showed that the

resetting mechanism helped PSO to evolve a smaller set of features with higher classification accuracy than [141] in most cases. Later, Tran et al. [142] also applied this resetting mechanism cooperating with local searches on $pbest$ to simultaneously reduced the number of selected features and improved the classification performance. In addition, the new fitness calculation was proposed, which was based on the changed features (selected to not selected or vice-versa). The proposed algorithm achieved smaller feature subset with lower classification error than [132]. Mistry et al. [143] embedded the concept of a micro-genetic algorithm (mGA) to improve the swarm diversity. The proposed algorithm used a small secondary swarm to explore five feature subsections. By alternatively selecting the lowest and highest correlation particles from the original swarm to form the secondary swarm, the proposed algorithm could balance local exploitation and global exploration. In addition, the $pbest$ and $gbest$ were modified using an averaging search strategy and a Gaussian mutation which also enhanced the local exploitation and global exploration. The experimental results showed that the proposed algorithm outperformed conventional GAs and PSO-based feature selection algorithms on facial emotion recognition. However, in a general feature selection problem, it is usually difficult to divide the feature space into many feature subsections due to the complex interactions between features. A recent attempt to avoid premature convergence was done by Cheng et al. [144]. In the proposed PSO algorithm, $gbest$ and $pbest$ were removed. The particles had to enter a competition and the winners went directly to the new population. The losers had to learn from the winners i.e. their positions were updated with respect to the positions of the winners, and then they could enter the new population. The proposed algorithm was called a competitive swarm optimizer, which worked well on large scale optimization problems. This modification of PSO was applied to feature se-

lection by Gu et al. [145].

There is not much attempt to reduce the computational costs of wrapper PSO-based feature selection. Shortening the length of particles is an option, which is done in [137, 138, 139]. Although the computation time is reduced, the evaluation time mainly remains the same as standard PSO-based feature selection. The improvement is from the updating position process and the upper bound of the number of selected features. [146] directly modify the fitness measure by splitting a training set into many subsets. From each subset, a number of features are selected and then all selected features are combined to form the final feature subset. However, it is possible that features selected from different subsets might be redundant.

- **PSO for filter-based feature selection**

  In order to solve feature selection problems, PSO has been used with different filter measures, for example, rough set theory [147, 148, 149, 150], fuzzy set [151] and information theory [152].

  The goal of feature selection using rough set is to find the smallest feature subset, which still preserves the classification quality as the original feature set. Therefore, Wang et al. [147] proposed a PSO-based filter feature selection approach, in which the fitness function of a feature subset was the combination of the classification quality of the feature subset calculated by rough set theory and the proportion of the selected features. The experimental results showed that the proposed algorithm could find the optimal solution in a smaller amount of time than a GA using rough sets. Cervante et al. [148] proposed a new fitness function, which used probabilistic rough set theory to minimize the number of equivalence class and maximize the number of instances in each equivalence class. The reported results illustrated that the new fitness function could guide PSO to search for a small subset, which had better classification performance than

the subsets evolved by two other PSO-based feature selection approaches using rough set.

Chakrabortly [151] proposed a filter PSO-based feature selection approach, which used fuzzy set to calculate the fitness value for each particle. Specifically, the membership value of fuzzy set theory was used with more than one threshold to decide whether or not an instance is consistent according to the selected features. The proportion of consistent instances over the total number of instances was the fitness measure of the current feature subset.

Besides rough set and fuzzy set theories, information theory has been also widely combined with PSO to solve feature selection problems. Based on the idea of "Max-relevance and min-redundancy" [120], mutual information was used to form fitness functions, which aimed to find a feature subset with a minimal redundancy within the subset and a maximal relevance between the subset and the class label. In [152], Cervanter et al. proposed two new information-based fitness functions. In the first fitness function, mutual information between two single selected features and between a selected feature and the class label (paired evaluation) were used to respectively compute the relevance and redundancy of the feature subset. These measures were also combined in the second fitness measure. However, in the second measure, instead of using mutual information, information gain (group evaluation) was used to calculate the relevance and redundancy of the feature subset. The results showed that both of fitness function successfully guided PSO to search for small feature subsets, which achieved better classification accuracy than using all features. The subset evolved by the first fitness function was smaller than the one evolved by the second fitness function. However, the second algorithm achieved better classification performance.

Recently, filter and wrapper approaches have been combined to form hybrid approaches, which take the advantages of both filter and wrapper methods. Nguyen et al. [153] proposed a wrapper PSO-based feature selection approach, in which $gbest$ was improved by a local search using a filter measure. The local search mimicked the typical backward feature selection method to remove features from $gbest$ according to the relevant and redundant measure calculated by mutual information. The experimental results showed that the proposed algorithms selected much smaller number of features while still achieved similar or better classification performance than the other PSO-based algorithms. Although the proposed algorithm had to perform an extra task for local search, its computational cost is still cheaper than other PSO-based algorithms because of the smaller number of selected features. In [154], a two phase PSO-based feature selection algorithm was proposed to deal with gene datasets. In these datasets, each gene is considered a single feature. In the first phase, a small number of genes were selected based on a correlation measure which maximized the gene-class correlation (relevance) and minimized the gene-gene correlation (redundancy). The second phase was a wrapper PSO-based feature selection using NB to evaluate feature subsets. The experimental results showed that the proposed algorithm outperformed the standard BPSO algorithm and another hybrid model proposed by [155].

Overall, it can be seen that PSO has been successfully applied to feature selection. However, as a population-based optimization approach, efficiency is a limitation of PSO-based feature selection. In addition, although both continuous and binary PSO have been applied to feature selection, there is more work applying continuous PSO. Xue et al. [156] conducted a comparison between binary and continuous PSO for feature selection, which indicated that generally continuous PSO achieved better performance than binary PSO although binary PSO had more natural rep-

resentation to feature selection.

Other EC techniques for feature selection

Besides PSO, other EC techniques such as GA, GP, DE have also been widely applied for feature selection problems. The rest of this section reviews some recent feature selection approaches using these EC techniques.

- **GAs-based feature selection** GAs could be considered the first EC technique being applied to feature selection. In GAs-based feature selection approaches, each chromosome is represented by a bit string, where each bit corresponds to one original feature. Each bit takes a value of "1" or "0", which respectively illustrates that the corresponding feature is "selected" or "not selected".

  One limitation of traditional GAs is the decrement of diversity due to simple genetic operators. Therefore, when the search space is huge, GAs usually converge quickly to local optima. In order to avoid this problem, Li et al. [157] proposed a multi populations based-feature selection algorithm where the neighboring populations exchanged their information/experience by sharing two individuals. In addition, a local search technique was applied to improve the best individual in each population. The experimental results showed that the proposed algorithm achieved better performance than four other GA-based feature selection approaches. However, the algorithm was tested only on datasets with less than 60 features. On the other hand, Derrac et al. [158] used multi populations GA to simultaneously perform feature selection and instance selection. In particular, the first population was purely used for feature selection, the second population concentrated on instance selection and the third population focused on both feature selection and instance selection. So in the proposed algorithm, two tasks were done simultaneously, which was very effective, especially for datasets which had a large number of

features as well as noisy instances.

In [159], Oreski et al. proposed a hybrid genetic algorithm with neural-networks (HGA-NN) to evolve an optimal feature subset. In the initialization step, the feature set was narrowed by different fast filter techniques. So important features, which were selected by the filter approaches, were used to initialize the major part of the population. The rest of population was filled randomly. In HGA-NN, an incremental stage was applied to enhance the creation of the initial population, which increased the diversity of the genetic material. The proposed algorithm was evaluated on two real-world credit datasets. The experimental results showed that HGA-NN achieved better classification performance than GA-NN technique [160].

Lin et al. [161] proposed a novel GA-based feature selection approach, in which the prior knowledge about financial distress prediction was used to group similar features. After that, a filter approach was used to rank all features in the same group and only top-rank features from each group were chosen to participate in the selection process by GAs algorithm. Although the two-step selection approach was efficient, it skipped the interaction between features. Recently, GAs were also proposed to select features in hierarchical feature spaces [162]. Two new mutation operators were proposed to deal with redundant features in a hierarchical space. The experimental results showed the GAs-based algorithm achieved better performance than two state-of-the-art hierarchical feature selection algorithms. Other GA-based feature selection approaches were developed recently to solve real-world problems, such as [163, 164, 165, 166, 167].

GAs were also widely applied to achieve multi-objective feature selection. Most of them applied NSGA-II [168, 169, 170] and recently NSGA-III [171].

Overall, as the first evolutionary algorithm, GAs have been successfully applied to feature selection for almost 30 years [10]. However, when GAs are applied to feature selection, its two genetic operators i.e. crossover and mutation need to be designed carefully, otherwise, they can break blocks of complementary features.

- **GP-based feature selection** In GP-based feature selection, each individual is represented as a tree. All leaf nodes in each tree are the original features, which are considered selected features.

  A new GP-based hyper-heuristic feature selection approach was proposed by Hunt [172]. The main idea was to evolve new heuristics based on two basic heuristics, which are Greedy Left (SBS) and Greedy Right (SFS). In which, each individual was a newly heuristic rule, which could be applied on an initial subset to evolve a final subset. In particular, GL operation removed a feature from a subset while GR will add a feature to a subset. The quality of the final subset was used as a fitness value of that GP individual. The experimental results on 3 datasets showed that the evolved heuristics were able to produce small feature sets while still improving the classification performance. However, the initial feature subset was very important, which could affect on the whole search process.

  Bhowan [173] proposed two GP-based approaches to evolve a set of features, which was used directly in the Watson system, an intelligent open-domain question answering system. The first approach extracted all features, which were used in the best-of-run evolved GP tree. The second approach considered all evolved trees. Particularly, from the set of GP trees, the top N features with the most frequency were chosen as extracting features. Two values of N used in this paper were 10 and 20. The experimental results showed that the set of features selecting from the best GP tree only worked well when the number of selected features was small. Meanwhile, selecting top

N features from the whole set of trees produced good resulted on both small and large feature set. However, as other ranking features selections, this algorithm did not consider the interactions between features, especially between redundant features.

Most GP-based feature selection algorithms implicitly perform feature selection by selecting features used in the final trees [174, 175]. Recently, Viegas et al. [176] proposed a GP-based feature selection algorithm which could explicitly perform feature selection. In the proposed algorithm, each inner node of GP was a set operator, and each leaf node was an original feature. Therefore, the output of each tree was a set of features rather than a formula as standard GP. The experimental results showed that the proposed algorithm could reduce up to 98% number of features on biological datasets without reducing the classification performance.

Overall, GP is also successfully applied to feature selection. With a flexible representation, GP can perform feature selection implicitly or explicitly. However, the flexible representation makes GP-based feature selection has much larger search space (possibly infinite without a tree depth limit) compared with PSO or GAs-based feature selection. The main reason is a feature subset can be represented by many different GP trees. In general, GP is still more suitable for feature construction [10].

- **DE-based feature selection**

  DE recently has been applied to solve feature selection problem since 2008. Most of the works focused on improving the searching mechanism and representation. Khushaba et al. [177] proposed a hybrid feature selection approach, in which DE operators were applied to improve the feature subsets found by ACO. Experimental results show that the proposed algorithms performed better than other traditional feature selection approaches. Later, [178] proposed a new

encoding scheme, in which each individual as a vector of floating numbers and the vector's length was predefined. Xue et al. [179] proposed a multi-objective DE-based feature selection approach, in which non-dominated sorting concept was applied to control the population. The experimental results showed that the proposed multi-objective feature selection approach evolved smaller feature sets and achieved better classification performance than single-objective approach. DE was also applied to achieve filter-based feature selection in [180]. In the proposed algorithm, two filter criteria, ReliefF and FisherScore, were combined as the ranking measure. Normalized mutual information was used as the relevance measure. The two measures were then either combined to form a single-objective DE-based algorithm or considered as two conflicting objectives to form a multi-objective DE-based algorithm. The experimental results showed that both DE-based algorithms evolved small feature subsets with better classification accuracies than using all features. The proposed fitness function also helps DE to evolve better feature subsets than the traditional filter criteria.

In general, DE can be successfully applied to feature selection. However, in comparison with PSO-based feature selection, the number of DE-based feature selection algorithms is still small. The possible reason is that DE faces some difficulties when it was applied to high dimensional problems [181].

2.7.2 Multi-objective Feature Selection

EC has been widely applied to multi-objective feature selection. Mukhopadhay et al. [168] proposed a multi-objective approach to feature selection, in which NSGA-II was used along with the SVM classification algorithm to identify miRNA markers. The representation encoded both the feature subset and the SVM's parameters. The experimental results on real-

world miRNA datasets showed that the proposed algorithm outperforms five deterministic feature selection methods. In addition, many identified miRNA markers were found to be related to different kinds of cancers. Leandro et al. [182] applied a multi-objective GA (MOGA) to achieve feature selection for face recognition. In this method, there were three objectives, which were the aggregation of the classification accuracy and the feature subset size, the number of selected coefficients, and the mutual information between selected features. The experimental results showed that the multi-objective approach achieved better classification performance than other state-of-the-art approaches. In addition, the solutions found by MOGA selected fewer features and still achieved similar accuracies to that of single-objective GA. Some other NSGA-II-based feature selection algorithms were also proposed [183, 184, 185]. Among EC techniques, GAs-based multi-objective algorithms were the most popular but those works simply applied GAs without considering the characteristic of feature selection such as the partially conflicting objectives or the two objectives are not equally important [10].

Multi-objective PSO was also widely applied to feature selection. Xue et al. [186] proposed the first multi-objective PSO (MOPSO) algorithm for feature selection. The target was to minimize both the classification error and the number of selected features using either continuous or binary PSO. The experimental results showed that MOPSO is superior to NSGA-II, SPEA2, and PAES2 on feature selection problems. A filter MOPSO-based feature selection was also proposed by Xue et al. [187], in which the two objectives were to minimize the number of features and to maximize the relevance between the selected features and the class labels. The relevance was calculated by using either mutual information or information gain. The experimental results suggested that MOPSO algorithms evolve feature subsets with higher classification performance than single-objective feature selection algorithms. Later, Nguyen et al. [188] improved the archive's solutions in MOPSO by applying three local search operators,

which are Inserting, Removing and Swapping. The proposed algorithm could select a smaller number of features and achieved similar or better classification performance than NSGA-II, SPEA2, PAES, and CMDP-SOFS [186] on 12 datasets. Recently, a multi-objective Differential Evolution (DE)-based feature selection algorithm was developed by Xue et al. [189]. During the evolutionary process, if the population size exceeded its maximum limit, solutions with lower dominance levels were removed. The algorithm selected better feature subsets than sequential search algorithms and two single-objective DE-based feature selection algorithms.

Most current multi-objective feature selection algorithms use Pareto dominance-based algorithms, which usually focus on the center of the Pareto front. This problem can be addressed by the MOEA/D framework. To the best of our knowledge, Paul et al. [190] proposed the first filter MOEA/D-based feature selection algorithm, which considered inter-class and intra-class distance measures as two conflicting objectives. A fuzzy rule was developed to extract a single solution from the final Pareto front. The experimental results showed that the proposed algorithm can evolve better feature subsets than sequential search and NSGA-II-based feature selection algorithms. However, the Pareto fronts evolved by MOEA/D and NSGA-II are not compared using any performance indicator. In addition, MOEA/D was applied directly to feature selection without considering the highly discontinuous Pareto front of feature selection.

2.8 Feature-based Transfer Learning

Most feature-based transfer learning approaches aim to build a latent feature space as a bridge between different domains. The projected datasets of two original datasets on the new feature space are expected to be closer than the original datasets.

In [191], a dimensionality reduction method, which used *Maximum Mean Discrepancy* [192] to measure the similarity between two distribu-

tions, was proposed to learn a new low-dimensional feature space. The new source dataset was then used to train a classifier which could be applied to classify the target instances. The work was further extended in [61] to address cases where information about the class label is available in the target domain. The two proposed algorithms, called TCA and STCA, were examined on two real-world applications, indoor WiFi localization, and cross-domain text classification. The results showed that both algorithms reduced the differences between source and target domains to achieve better target performance than using the original feature set. Shi and Sha [193] measured domain differences using mutual information between instances and their binary domain labels. The assumption was that the classes in two domains were well separated and instances across domains had to be close to each other if they belonged to the same class. Shi and Sha showed that by considering discriminative characteristics on both source and target domains, the classification performance was significantly improved over TCA/STCA [61]. However, the latent feature space was assumed to be a linear transformation of the original feature space. Yan et al. [62] proposed two algorithms, named MIDA and SMIDA, which could cope with continuous distributional changes in the feature space. The latent feature space is built to have a maximized independence with domain features. Some works projected both source and target data into two different feature spaces and then built connections between the newly built spaces [194]. Although building a latent feature space might result in good performance, it lost meaning and possible important information of the original features.

An early work on feature selection for domain adaptation was performed by Uguroglu and Carbonel [59]. The task was to select original features that were not much different between two domains, which were called domain-invariant features. Invariant and variant features could be distinguished based on their performance in minimizing domains' gap. It was shown that the proposed algorithm achieved better performance than

TCA [61] on the digital recognition and Wifi localization problems. The efficiency of the work was improved by Tahmoresnezhad and Hashemi [60]. Particularly, the input dataset was split into k smaller sub-datasets. The proposed algorithm was run on all different sub-datasets, so each feature had k different weights. The average weight determined whether a feature was selected. However, in these works, a threshold value had to be predefined to select features. The features were selected individually based on their weights, which meant that feature interactions were ignored. In addition, discriminative abilities on both source and target domains were not considered together, so some selected domain-invariant features might be irrelevant to the class label.

2.9 Summary

This chapter introduced essential concepts of machine learning, classification, transfer learning, feature selection, evolutionary computation techniques, particularly PSO and MOEA/D, and mutual information. This chapter also reviews both EC and non-EC based approaches for feature selection.

It has been shown that EC is successfully applied to achieve feature selection. However, this chapter also highlights the limitations of existing EC-based feature selection algorithms on different aspects such as fitness function (filter/wrapper), searching mechanism, and the number of objectives. The limitations of existing work and the motivations of this research can be summarized as follows.

- **Mutual information estimation for feature selection**

  Mutual information can be used to measure redundancy and relevance between features. It can be calculated by using the counting approach or the estimation approach. Although both approaches have been applied to feature selection, there is no existing work ex-

plicitly comparing the two approaches. Therefore, it is needed to examine the ability to detect feature interactions of the two approaches, especially when they are used with EC techniques.

- **BPSO-based feature selection**

  Feature selection is a combinatorial optimization problem and it has a binary search space. BPSO has been applied to feature selection but its performance is limited in comparison with continuous PSO. Many studies attempted to improve the performance of BPSO by modifying parts of BPSO. However, there is no existing work investigating the inappropriate velocity and momentum concepts in standard BPSO. These two core concepts need to be redefined to make BPSO cope with the characteristics of binary search spaces.

- **Efficient wrapper-based feature selection**

  Wrapper-based feature selection usually achieves good classification performance but its computational cost is expensive. Some studies aim to improve its efficiency by shortening the representation. However, the main reason, which is its expensive evaluation process, has not been paid enough attention.

- **Wrapper MOEA/D-based feature selection**

  Most of the existing multi-objective feature selection algorithms use Pareto dominance-based algorithms, which usually result in a small set of non-dominated feature subsets. MOEA/D, a decomposition-based algorithm, usually evolves a more diverse non-dominated set than the Pareto dominance-based algorithm. However, there is no existing work investigating the use of MOEA/D for wrapper-based feature selection.

- **Feature selection-based transfer learning**

Feature-based transfer learning is one of the most common approaches to achieve transfer learning. However, most existing feature-based methods have to assume models to measure differences between data distributions. In addition, the existing methods do not take into account the interactions between features. Although EC techniques can address the two limitations, there is no work applying EC to select features for transfer learning.

Each of the following five chapters focuses on addressing each of the above limitations.

Chapter 3

Mutual Information for Feature Selection

3.1 Introduction

Among filter measures for feature selection, mutual information gains attention because it can detect non-linear correlations between features easier than other measures [26, 27]. However, most existing mutual information based feature selection algorithms count the number of instances in a dataset to derive probability distributions. This counting approach can result in inaccurate mutual information when there are not enough instances. In addition, the counting approach is applicable only to discrete datasets. To overcome these limitations, several estimation methods have been proposed to estimate mutual information [195]. Although mutual information estimation has been used to achieve feature selection [122], there is no work that explicitly compares the effect of counting and estimation approaches on feature selection, especially when they are used with an Evolutionary Computation algorithm.

3.1.1 Chapter Goal

The overall goal of this chapter is to develop a PSO-based feature selection algorithm with a fitness function based on mutual information estimation to select informative features from the original feature set. The proposed algorithm is expected to reduce the number of features while at least maintaining or improving the classification performance in comparison with using all features. Specifically, we will investigate:

- whether mutual information estimation for feature selection can work well on both discrete and continuous datasets,
- whether mutual information estimation can achieve better performance than the counting approach in terms of the classification performance, and
- whether mutual information estimation can capture the interactions between features better than the counting approach.

Note, the mutual information estimation approaches require that the feature values have a well defined distance metric (more details can be seen in Section 2.5.2 of Chapter 2). All the datasets considered in this chapter, therefore, have numeric features only.

3.2 Proposed Algorithm

In a feature selection algorithm, there are usually two main parts including a searching mechanism and a fitness function, which are responsible for generating feature subset candidates and evaluating the generated subsets, respectively. In this chapter, PSO is used as the searching mechanism. This section firstly describes the PSO's representation when it is applied to feature selection. Secondly, it shows how to form a fitness function using mutual information estimation, which is the main contribution of this chapter. Finally, it presents the overall algorithm.

3.2.1 Representation

As discussed in Chapter 2, PSO has a vector-based representation, which can be naturally applied to feature selection. Each particle's position is a vector of real numbers whose length is equal to n, the total number of original features. Each position entry in the position vector corresponds to an original feature. The entry value, which is in the range [0, 1], is used to determine whether the corresponding feature is selected or not. Particularly, the decision is based on a threshold θ value: if the entry value is greater than θ, the corresponding feature is selected; otherwise the feature is discarded. Therefore, each particle's position represents a candidate feature subset. The following subsection describes how to evaluate the goodness of the candidate feature subsets.

3.2.2 Proposed Fitness Function

The fitness function is used to evaluate feature subsets represented by particles, so it plays a vital role in guiding PSO to search for an optimal feature subset. In this chapter, we use mutual information to form the fitness function. Most mutual information-based feature selection approaches utilize mutual information to measure the relevance and redundancy of a feature subset, using two formulas, Eq. (3.1) and Eq. (3.2):

$$Relevance(S) = MI(S, C) \tag{3.1}$$

$$Redundancy(S) = MvI(f_1, f_2, \ldots, f_m) \tag{3.2}$$

where C is the class label, S is a feature subset, which contains m features $f_1, \ldots, f_m$. Note that MvI is multi-variate mutual information, which is an extension of mutual information to measure the mutual information between a set of features. More details about multi-variate mutual information were given in Section 2.5.1 of Chapter 2.

The aim of feature selection is to produce an optimal feature subset by removing irrelevant and redundant features. The optimal feature subset minimizes the quality measure given in Eq. (3.3).

$$Fitness(S) = -\alpha \times Relevance(S) + (1-\alpha) \times Redundancy(S) \tag{3.3}$$

where α is used to control the contributions of relevance and redundancy to the fitness measure. In feature selection, the goal of increasing the classification performance has a higher priority than reducing the number of features, a larger weight is usually assigned to $Relevance$. Therefore, α is usually in the range [0.5, 1], which makes $(1-\alpha)$ fall in the range [0, 0.5].

Note that in Eq.s (3.1) and (3.2), multi-variate information is used to measure the relevance of a feature subset or the redundancy between a set of features. However, computing it is not an easy task. Suppose that each feature $f_i \in S$ has v_i possible values, then the total number of possible values of the feature set S is $\prod_{i=1}^{m} v_i$. Therefore, accurately calculating the mutual information of a feature set usually requires many instances of each possible value, which is a huge number of instances in the training set. This is seldom satisfied in real-world datasets. For instance, gene expression datasets can have up to thousands of features but a very small number of samples. To cope with the lack of samples, the relevance and redundancy measures are estimated by decomposing them into pair-wise mutual information, which can be seen in Eq. (3.4) and Eq. (3.5).

$$Relevance_{pw}(S) = \sum_{i=1}^{m} MI(f_i, C) \tag{3.4}$$

$$Redundancy_{pw}(S) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} MI(f_i, f_j) \tag{3.5}$$

Based on the two pair-wise terms, the fitness function used in this chapter is defined by Eq. (3.6).

$$Fitness_{pw}(S) = -\alpha \times Relevance_{pw}(S) + (1-\alpha) \times Redundancy_{pw}(S) \tag{3.6}$$

In order to calculate the pair-wise mutual information between two features, it is necessary to know the probability distribution of each feature including the class label. In most current approaches, the probability distribution is estimated by counting the number of instances in the training set with each possible feature value; and constructing a discrete probability distribution. Although this approach is efficient for discrete datasets, it is difficult to apply to continuous datasets since each continuous variable has an infinite number of possible values. In order to be applied to continuous datasets, counting approaches require some ways of discretizing the datasets.

Another approach is to apply mutual information estimation which does not require discretization of continuous datasets. There are many mutual information estimation methods (discussed in section 2.5.2 of Chapter 2), among them kernel density estimation (KDE) [114] and nearest neighbor estimation (NNE) [115] give the most promising results. In KDE, a kernel is used to measure the distance between two instances given a feature subset S. The probability of an instance is estimated based on the number of instances whose distances to the current instance are less than a predefined boundary, called r. NNE uses the K nearest neighbors of an instance to derive the boundary r, so NNE can be seen as an improvement of KDE. However, NNE critically depends on the search for nearest neighbors. It is known that the nearest neighbor search faces difficulties when the number of features or dimensions increases due to the "curse of dimensionality" [196]. In addition, the search for nearest neighbors also makes NNE more computationally intensive than KDE. Therefore, in this chapter KDE is chosen as a representative of estimation approaches to compare with the counting approach.

The two PSO-based feature selection algorithms, which use KDE and the counting approach in their fitness functions, are called PSO-KDE and PSO-C, respectively.

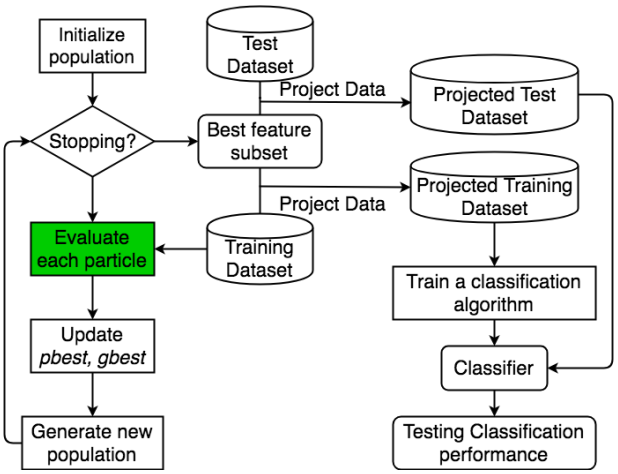


Figure 3.1: Overall feature selection system.

3.2.3 Overall Algorithm

The overall algorithm can be seen in Figure 3.1, where the training set is used in the evaluation process. The best evolved feature subset is then tested on the test set. The obtained testing classification performance is used as the main criterion to compare between different feature selection algorithms. In this chapter, mutual information is used to form the fitness function and the aim is to investigate the effect of two different ways to calculate mutual information, which are the counting and estimation approaches. The main contribution of this chapter is in the evaluation step, which is marked in green in the figure.

The pseudo-code of the proposed PSO-based feature selection algorithm is shown in Algorithm 1.

3.3 Experiment Design

3.3.1 Datasets

KDE and the counting approach are compared on both artificial and real-world datasets. The real-world datasets are shown in Table 1.1. The ar-

Algorithm 1 : Pseudo-code of the proposed algorithm

```
randomly initialize the position and velocity of each particle;
while Maximum iteration is not reached do
    for i = 1 to PopulationSize do
        evaluate the fitness of particle i using Eq. (3.6);
        update the pbest of particle i;
    end for
    update gbest for each particle;
    for i = 1 to PopulationSize do
        update velocity of particle i using Eq. (2.1);
        update position of particle i using Eq. (2.2);
    end for
end while
output the feature subset selected by gbest;
```

tificial datasets can be seen in Table 3.1, where "Con" and "Dis" mean respectively continuous and discrete datasets, #Fs means the total number of features, #Cs means the total number of class values and #Is is the total number of available instances.

There are 7 different artificial datasets, which have different relationships between features and between features and the class labels. The first two artificial datasets have three binary features. In Binary 1, an instance belongs to class 1 if exactly two features have value 1, otherwise, the in-

Table 3.1: Artificial datasets

Dataset	Type	#Fs	#Cs	#Is
Binary 1	Dis	3	2	8
Binary 2	Dis	3	2	8
Monk 1	Dis	6	2	432
Monk 2	Dis	6	2	432
Monk 3	Dis	6	2	432
2-way linear	Con	4	2	200
3-way linear	Con	4	2	200

stance is in class 0. In Binary 2, if all features of an instance have the same values then the instance is in class 1, otherwise, it belongs to class 0. In these two datasets, there is no redundancy and all three features are relevant to the class label. Feature selection on these datasets should select all three features.

Three other artificial datasets are the Monk datasets [32], which have 6 discrete features and one binary class label. The 3^{rd} and 6^{th} features are binary variables, which can be either 1 or 2. The 5^{th} feature has four possible values from 1 to 4. The other features have three values, which range from 1 to 3. In Monk 1 dataset, the class label is 1 if either $f_0 = f_1$ or $f_4 = 1$, so the optimal feature set of Monk 1 is $\{f_0, f_1, f_4\}$. Meanwhile, in Monk 2, the class label is 1 if there are exactly two features taking value 1. In this case, all features are important in the Monk 2 dataset. The last Monk dataset is a bit more complicated, where the class label is 1 if ($f_3 = 1$ and $f_4 = 3$) or ($f_4 \neq 4$ and $f_1 \neq 3$). In the Monk 3 dataset, the most important feature subset is $\{f_1, f_3, f_4\}$. Notice that there is no redundancy in the Monk datasets.

2-way linear and 3-way linear have 4 continuous features. In 2-way linear, the last two features are copies of the first two features ($f_0 = f_2, f_1 = f_3$). The class label is set to 1 if the average of the first two features is greater than 0.5. Therefore, the optimal feature subset for this dataset is one of the four feature subsets, $\{f_0, f_1\}$, $\{f_0, f_3\}$,$\{f_1, f_2\}$ or $\{f_2, f_3\}$. In 3-way linear, the first two features are two random variables, which fall in [0,1]. The 3^{rd} feature is the average of the first two features, $f_2 = \frac{f_0 + f_1}{2}$. The 4^{th} feature (f_3) is just a copy of the first feature. Therefore in this dataset, there is redundancy in any feature subsets that contains f_0 along with f_3 or (f_1 and f_2). The class label is determined by feature f_2. Particularly, the class label is set to 1 if $f_2 > 0.5$, so the optimal feature subset for this dataset is $\{f_2\}$.

3.3.2 Parameter Settings

In this chapter, 10-fold cross-validation is used to conduct the experiments. Particularly, each dataset is divided into 10 folds. Each fold will be selected as a test set and the other folds are used as a training set to select features. For each method, this process is run 30 independent times on each dataset; so there will be 300 evolved feature subsets. Since each dataset has continuous and discrete versions, the selected feature subsets are tested on both versions using three classification algorithms K-nearest neighbor (KNN), Decision Tree (DT) and Naive Bayes (NB), which are representatives of instance-based, tree-based and probabilistic classification algorithms, respectively. For the KNN classification algorithm, K is set to 5 so that the classification algorithm is able to avoid noise instances with a good efficiency. REP (Reduced-Error Pruning) tree is picked as a representative of the DT classification algorithm. The setting of REP tree follows the default settings in Weka [197].

The kernel width r needs to satisfy the condition $K_r \leq N/(3/r)^{n_r}$, where K_r is the number of neighbors fall in the range r, n_r is the number of features and N is the total number of instances. In this case, since only pair-wise mutual information is used, the number of dimensions n_r is 2. Lungarella et al. [198] proposed that K_r should be at least equal to 3 to avoid undersampling effects. Therefore, in this chapter K_r is set to 3. From the above conditions, the kernel width r is specified by $\frac{3}{\log_2 N/3}$.

The weight α in the pair-wise fitness measure Eq. (3.3) has three different values: 0.6, 0.8 and 1.0 to evaluate the effect of different relevance and redundancy's contributions. The three values are evenly selected in the range [0.5, 1], which gives higher priority to the relevance term.

For PSO algorithm, the fully connected topology is used. The parameters are set as follows [199]: $w = 0.7298, c_1 = c_2 = 1.49618, v_{max} = 2.0$. The population size is 30 and the maximum number of iterations is 100. The threshold θ is set as 0.6.

The results of counting and estimation approaches are compared by both the Wilcoxon and ANOVA tests with a confidence interval of 95%.

3.4 Results and Discussion

Experimental results on real-world and artificial datasets are shown in Tables 3.2 and 3.3, respectively. Each table shows the results of PSO-KDE and PSO-C on a dataset. The prefix "Con-" and "Dis-" correspond to the results on the continuous and discrete versions of each dataset. The Wilcoxon significant test between KDE and counting approach is shown in the brackets, besides KDE's accuracies. The confidence interval of the Wilcoxon test is set to 0.95. "+", "=" or "-" mean that PSO-KDE approach is respectively significantly better, similar or significantly worse than PSO-C. Table 3.4 shows which features are selected by the PSO-based feature selection algorithms on artificial datasets.

3.4.1 Real-world Datasets

The results on the 12 real-world datasets are shown in Table 3.2. Given a specific α value, the best testing accuracy on each dataset is marked in bold.

PSO-KDE vs PSO-C on real-world datasets

In terms of the classification accuracy, PSO-KDE is significantly better than PSO-C on the continuous version of most of datasets. For example, on the Wine dataset, the classification accuracy of PSO-KDE is about 10% better than PSO-C on both DT and NB. On Ionosphere and Sonar, the feature subsets generated by PSO-KDE achieve up to 10% better than PSO-C regardless of the similar number of selected features. On WBCD, PSO-KDE is significantly better on all the three classification algorithms when α is set to 0.6 and 0.8. In summary, on the continuous version of datasets,

Table 3.2: Testing accuracies on real-world datasets.

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	13	94.38	80.94	86.86	13	93.25	97.76	97.78
0.6	PSO-C	1.0	82.52	80.48	67.72	2.24	92.58	**93.06**	92.7
	PSO-KDE	2.56	**92.84**(+)	81.69(=)	75.71(+)	2.24	92.42(=)	93.01(=)	92.63(=)
0.8	PSO-C	1.0	83.12	81.17	67.9	4.98	93.72	**96.91**	96.3
	PSO-KDE	4.98	**95.61**(+)	80.98(=)	81.15(+)	4.98	93.74(=)	96.79(=)	96.31(=)
1.0	PSO-C	11.92	94.48	80.81	85.84	11.99	93.45	97.08	**97.39**
	PSO-KDE	11.95	**94.51**(=)	80.76(=)	86.04(=)	12.01	93.46(=)	97.06(=)	97.33(=)

(a) Wine

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	14	84.64	68.99	85.22	14	85.51	85.22	85.22
0.6	PSO-C	3.03	85.04	83.97	**85.56**	3.03	85.07	84.92	**85.55**
	PSO-KDE	2.62	85.06(=)	80.5(-)	85.36(-)	2.62	85.36(+)	84.39(=)	85.33(-)
0.8	PSO-C	5.27	**85.24**	80.33	84.56	5.27	85.22	84.99	85.34
	PSO-KDE	4.49	84.04(-)	74.5(-)	84.85(=)	4.49	85.26(=)	84.7(=)	**85.39**(=)
1.0	PSO-C	12.33	84.68	69.07	**85.08**	12.33	85.46	85.3	85.39
	PSO-KDE	12.58	84.68(=)	69.28(=)	85.01(=)	12.58	**85.49**(=)	85.38(=)	85.39(=)

(b) Australian

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	18	85.93	83.04	81.32	18	85.93	83.04	81.32
0.6	PSO-C	1.02	**75.8**	75.01	71.12	1.02	**75.8**	75.01	71.12
	PSO-KDE	1.97	75.17(-)	74.41(=)	74.39(+)	1.97	75.17(-)	74.41(=)	74.39(+)
0.8	PSO-C	1.18	76.96	76.07	71.73	1.18	76.96	76.07	71.73
	PSO-KDE	3.83	**81.43**(+)	80.1(+)	78.69(+)	3.83	**81.43**(+)	80.1(+)	78.69(+)
1.0	PSO-C	15.96	**85.49**	82.47	81.18	15.96	**85.49**	82.47	81.18
	PSO-KDE	16.25	85.43(=)	82.46(=)	81.33(=)	16.25	85.43(=)	82.46(=)	81.33(=)

(c) Vehicle

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	24	74.2	68.2	73.5	24	74.2	68.2	73.5
0.6	PSO-C	3.2	70.11	67.53	70.66	3.2	70.11	67.53	70.66
	PSO-KDE	3.02	70.88(+)	68.36(=)	**71.33**(+)	3.02	70.88(+)	68.36(=)	**71.33**(+)
0.8	PSO-C	4.97	71.81	70.09	72.39	4.97	71.81	70.09	72.39
	PSO-KDE	5.03	72.4(+)	71.0(+)	**72.97**(+)	5.03	72.4(+)	71.0(+)	**72.97**(+)
1.0	PSO-C	19.76	73.95	68.69	73.18	19.76	73.95	68.69	73.18
	PSO-KDE	19.98	**74.21**(=)	68.95(=)	73.58(=)	19.98	**74.21**(=)	68.95(=)	73.58(=)

(d) German

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	30	94.73	93.32	88.57	30	91.91	96.49	94.38
0.6	PSO-C	1.37	88.21	87.32	75.93	2.07	92.09	91.74	92.73
	PSO-KDE	2.14	**91.81**(+)	90.31(+)	84.76(+)	2.07	92.07(=)	91.75(=)	**92.73**(=)
0.8	PSO-C	1.9	90.25	89.93	80.72	4.21	93.0	94.39	**94.87**
	PSO-KDE	3.79	**93.49**(+)	90.9(+)	89.26(+)	4.2	93.02(=)	94.41(=)	94.85(=)
1.0	PSO-C	24.96	94.14	92.94	88.49	25.01	92.31	96.06	94.19
	PSO-KDE	24.83	**94.21**(=)	93.04(=)	88.79(=)	25.0	92.35(=)	**96.07**(=)	94.18(=)

(e) WBCD

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	34	89.17	84.33	35.9	34	90.87	85.19	90.58
0.6	PSO-C	2.4	81.52	79.7	81.17	2.31	**84.89**	84.49	84.65
	PSO-KDE	2.37	84.1(+)	**84.71**(+)	83.3(+)	2.25	84.75(=)	84.42(=)	84.54(=)
0.8	PSO-C	2.65	80.01	78.01	81.68	4.03	88.93	89.11	89.22
	PSO-KDE	4.08	87.75(+)	**88.12**(+)	80.86(=)	4.0	88.89(=)	89.17(=)	**89.24**(=)
1.0	PSO-C	27.87	88.53	83.73	35.9	27.55	90.59	84.89	90.55
	PSO-KDE	27.75	**89.03**(=)	84.14(+)	35.9(=)	27.59	**90.65**(=)	84.89(=)	90.56(=)

(f) Ionosphere

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	60	74.0	80.17	50.38	60	72.57	85.07	75.98
0.6	PSO-C	1.57	57.48	56.88	51.86	2.11	63.87	62.06	**64.29**
	PSO-KDE	2.18	61.7(+)	**62.03**(+)	52.32(=)	2.13	63.41(=)	61.64(=)	63.83(=)
0.8	PSO-C	1.58	57.27	57.87	52.09	2.69	68.3	67.11	68.38
	PSO-KDE	2.67	**67.21**(+)	67.05(+)	50.64(=)	2.69	68.46(=)	67.05(=)	**68.58**(=)
1.0	PSO-C	46.29	72.96	80.22	51.11	45.99	73.13	**83.75**	75.19
	PSO-KDE	45.79	72.81(=)	**80.54**(=)	50.02(-)	45.91	73.35(+)	83.66(=)	75.15(=)

(g) Sonar

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	100	58.25	56.93	51.98	100	49.26	50.9	51.23
0.6	PSO-C	4.34	**53.19**	53.01	50.41	4.34	48.47	49.94	**51.45**
	PSO-KDE	4.34	53.07(=)	52.86(=)	50.84(+)	4.34	48.59(=)	50.06(=)	51.43(=)
0.8	PSO-C	4.35	53.19	52.94	50.5	4.35	48.47	50.01	51.4
	PSO-KDE	4.44	**53.21**(=)	52.9(=)	50.67(=)	4.44	48.67(=)	50.1(=)	**51.51**(=)
1.0	PSO-C	72.2	57.36	56.92	52.06	72.2	49.7	51.0	**51.15**
	PSO-KDE	72.23	**57.55**(=)	56.93(=)	52.06(=)	72.23	49.65(=)	51.01(=)	51.14(=)

(h) Hillvalley

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	166	74.59	86.97	65.36	166	64.59	86.97	65.36
0.6	PSO-C	11.21	73.13	76.46	68.91	11.2	73.11	76.51	68.98
	PSO-KDE	11.81	73.67(=)	**76.91**(=)	68.03(=)	11.81	73.67(=)	**76.91**(=)	68.03(=)
0.8	PSO-C	11.19	73.05	76.31	69.23	11.24	73.17	76.4	69.28
	PSO-KDE	12.06	73.65(=)	**76.92**(=)	68.28(=)	12.06	73.65(=)	**76.92**(=)	68.28(=)
1.0	PSO-C	113.27	75.59	85.9	74.89	113.27	75.59	85.93	74.89
	PSO-KDE	113.7	75.1(=)	**86.01**(=)	74.92(=)	113.7	75.1(=)	**86.01**(=)	74.92(=)

(i) Musk1

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	278	94.86	93.57	94.96	278	94.86	93.57	94.96
0.6	PSO-C	41.85	93.71	93.3	93.75	41.85	93.71	93.3	93.75
	PSO-KDE	41.79	93.71(=)	93.29(=)	**93.75**(=)	41.79	93.71(=)	93.29(=)	**93.75**(=)
0.8	PSO-C	42.42	**93.91**	93.48	93.85	42.42	**93.91**	93.48	93.85
	PSO-KDE	42.24	93.89(=)	93.5(+)	93.84(=)	42.24	93.89(=)	93.5(+)	93.84(=)
1.0	PSO-C	174.63	94.67	93.75	95.01	174.63	94.67	93.75	95.01
	PSO-KDE	174.67	94.67(=)	93.75(=)	**95.01**(=)	174.67	94.67(=)	93.75(=)	**95.01**(=)

(j) Arrhythmia

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	500	78.58	72.88	50.0	500	79.69	57.62	50.0
0.6	PSO-C	70.4	56.43	53.84	50.72	70.4	56.13	51.15	52.89
	PSO-KDE	70.39	**58.11**(=)	54.74(=)	50.85(=)	70.39	**57.59**(=)	51.56(=)	53.7(=)
0.8	PSO-C	70.29	56.8	53.94	50.72	70.29	56.72	51.65	52.82
	PSO-KDE	70.48	**58.58**(=)	55.15(=)	50.96(=)	70.48	**58.29**(=)	51.93(=)	54.05(+)
1.0	PSO-C	297.56	76.73	71.6	50.0	297.56	**78.74**	57.01	59.85
	PSO-KDE	297.68	**76.85**(=)	71.16(=)	50.0(=)	297.68	78.26(=)	56.8(=)	59.96(=)

(k) Madelon

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	649	98.62	98.99	82.0	649	98.48	99.6	82.15
0.6	PSO-C	110.84	97.41	98.45	98.4	110.84	97.46	99.25	98.98
	PSO-KDE	108.71	97.38(=)	**98.46**(=)	98.37(=)	108.71	97.48(=)	**99.28**(=)	98.97(=)
0.8	PSO-C	110.18	97.55	98.39	**98.47**	110.18	97.61	99.28	98.99
	PSO-KDE	110.13	97.55(=)	98.23(=)	98.46(=)	110.13	97.64(=)	**99.3**(=)	99.02(=)
1.0	PSO-C	382.24	98.49	**98.87**	82.0	382.24	98.46	99.57	99.2
	PSO-KDE	384.55	98.46(=)	98.86(=)	82.0(=)	384.55	98.46(=)	**99.57**(=)	99.17(=)

(l) Multiple Features

in almost all cases PSO-KDE achieves similar or better performance than PSO-C in the three classification algorithms.

On the discrete version of each dataset, PSO-KDE also achieves similar or better performance than PSO-C. In most cases, PSO-KDE outperforms PSO-C when α is set to 0.8. For example, on the Vehicle dataset (Table (3.2c)), the improvements of PSO-KDE over PSO-C on KNN, DT and NB are 4.5%, 4%, and 7% respectively. Despite selecting the same number of features, with $\alpha = 0.8$, PSO-KDE's accuracies on all the three classification algorithms are up to 1% higher than the results of PSO-C. The experimental results show that KDE is not only able to cope with both continuous and discrete datasets but also guides PSO to achieve similar or better classification performance than the counting approach, which only works well with discrete datasets. However, when α is smaller than 1, PSO-KDE achieves lower classification accuracy than using all features. Only when α is set to 1.0, PSO-KDE achieves comparative classification accuracy while selecting about half of the original features. The possible reason is that the redundancy makes both PSO-based algorithms remove

too many features from the original feature set (up to 90%), which may contain many relevant features.

In terms of the number of selected features, when α increases, which means the contribution of redundancy into the fitness function decreases, the numbers of features selected by both PSO-KDE and PSO-C increase. The extreme case is when redundancy is ignored (α = 1.0), on the datasets with small numbers of features, almost all original features are selected. Meanwhile, when the number of original features is larger, the proportion of selected features is smaller. The reason might be that a dataset with a large number of features might contain many irrelevant features.

Given a smaller contribution of redundancy, the number of features selected by both PSO-KDE and PSO-C are smaller. However, it does not mean that the redundancy measure works well in this case. The reason is that $Redundancy_{pw}$, which is shown in Eq. (3.5), is a monotonic function. Regardless of which features are selected, adding any feature into the feature subset results in additional MI, which increases $Redundancy_{pw}$ because mutual information is non-negative. In this case, it only can be confirmed that PSO does find out optimal or near-optimal feature subsets when $\alpha = 1.0$. It would be difficult to analyze the effect of $Relevance_{pw}$ and $Redundancy_{pw}$ in the real datasets since the optimal feature subset is unknown. Therefore, a deep analysis on the artificial datasets is provided in the next section.

3.4.2 Artificial Datasets

Tables 3.3 and 3.4 show respectively the test accuracies and the feature subsets selected by PSO-C and PSO-KDE on the seven artificial datasets. Given a specific α value, the best testing accuracy on each dataset is marked in bold. The results of two datasets Binary 1 and Binary 2 are not shown because DT, KNN, and NB are not able to classify these problems (0% accuracy). In terms of the classification accuracy, as can be seen from Table

3.3, PSO-KDE achieves similar or significantly better results than PSO-C. The largest difference between the two algorithms is in the Monk 1 dataset, where PSO-KDE's accuracies are about 25% better PSO-C's accuracies.

The more important factor to be considered in the artificial datasets is the evolved feature subsets. For each α value, feature selection algorithms are run 30 independent times on each dataset. The 10-fold cross validation is also used on the artificial datasets. Therefore there will be 300 (30×10) feature subsets generated for each α value and each dataset. The feature subsets selected by PSO-C and PSO-KDE are shown in Table 3.4. In the table, all indexes of selected features are in the curly brackets, which follows by the number of times that the feature subset is selected. For example, $\langle$ {0,1,2}: 300$\rangle$ means that the feature subset $\{f_0, f_1, f_2\}$ are selected 300 times.

In two Binary datasets, the optimal set is the original feature set. According to the experimental results, regardless of the values of α, the original feature set is selected by PSO-KDE in more than 98% of the 300 times. Because there is no redundancy in these datasets, the α values should not affect on the evolved feature subsets. Therefore, the redundancy measured by KDE works well in this case. For the counting approach, the proportion of the original feature set to all feature subset ranges from 20% to 100% when α increases from 0.6 to 1.0. When redundancy contributes to the fitness function, counting approach still results in a smaller set than the optimal set regardless of the fact that redundancy should be 0. Therefore, it can be seen that the redundancy measured by the counting approach does not work well on Binary datasets. Particularly, redundancy between two independent features, measured by the counting approach is greater than 0.

In the Monk 1 dataset, the optimal feature subset is {f_0, f_1, f_4} and there is no redundancy in this dataset. Three features f_2, f_3, and f_5 are irrelevant to the class label. Once more, since the redundancy in this dataset is 0, the α values should not affect the selected feature subsets. This fact is com-

Table 3.3: Testing accuracies on artificial datasets.

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	85.87	94.21	75.0	6	85.87	94.21	75.0
0.6	PSO-C	1.59	75.0	63.22	75.0	1.59	75.0	63.22	75.0
	PSO-KDE	3.0	99.77(+)	**100.0**(+)	75.0(=)	3.0	99.77(+)	**100.0**(+)	75.0(=)
0.8	PSO-C	2.79	75.0	66.79	75.0	2.79	75.0	66.79	75.0
	PSO-KDE	3.0	99.77(+)	**100.0**(+)	75.0(=)	3.0	99.77(+)	**100.0**(+)	75.0(=)
1.0	PSO-C	5.94	85.88	93.2	75.0	5.94	85.88	93.2	75.0
	PSO-KDE	3.0	99.77(+)	**100.0**(+)	75.0(=)	3.0	99.77(+)	**100.0**(+)	75.0(=)

(a) Monk 1

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	79.63	69.46	66.45	6	79.63	69.46	66.45
0.6	PSO-C	4.67	65.96	57.58	66.26	4.67	65.96	57.58	66.26
	PSO-KDE	5.94	**78.92**(+)	68.7(+)	66.48(+)	5.94	**78.92**(+)	68.7(+)	66.48(+)
0.8	PSO-C	5.24	69.83	62.04	66.24	5.24	69.83	62.04	66.24
	PSO-KDE	5.94	**78.92**(+)	68.7(+)	66.48(+)	5.94	**78.92**(+)	68.7(+)	66.48(+)
1.0	PSO-C	5.95	78.84	68.74	66.46	5.95	78.84	68.74	66.46
	PSO-KDE	5.94	**78.92**(=)	68.7(=)	66.48(=)	5.94	**78.92**(=)	68.7(=)	66.48(=)

(b) Monk 2

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	6	**100.0**	99.54	97.23	6	**100.0**	99.54	97.23
0.6	PSO-C	3.29	**100.0**	**100.0**	97.23	3.29	**100.0**	**100.0**	97.23
	PSO-KDE	3.0	**100.0**(=)	**100.0**(=)	97.23(=)	3.0	**100.0**(=)	**100.0**(=)	97.23(=)
0.8	PSO-C	3.97	**100.0**	**100.0**	97.23	3.97	**100.0**	**100.0**	97.23
	PSO-KDE	3.0	**100.0**(=)	**100.0**(=)	97.23(=)	3.0	**100.0**(=)	**100.0**(=)	97.23(=)
1.0	PSO-C	5.97	**100.0**	99.53	97.23	5.97	**100.0**	99.53	97.23
	PSO-KDE	3.0	**100.0**(=)	**100.0**(+)	97.23(=)	3.0	**100.0**(=)	**100.0**(+)	97.23(=)

(c) Monk 3

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	4	92.5	94.5	46.0	4	92.5	94.5	46.0
0.6	PSO-C	1.0	69.5	69.3	54.0	1.0	69.5	69.3	54.0
	PSO-KDE	2.0	92.5(+)	**94.5**(+)	54.0(=)	2.0	92.5(+)	**94.5**(+)	54.0(=)
0.8	PSO-C	1.0	69.5	69.3	54.0	1.0	69.5	69.3	54.0
	PSO-KDE	2.0	92.5(+)	**94.5**(+)	54.0(=)	2.0	92.5(+)	**94.5**(+)	54.0(=)
1.0	PSO-C	4.0	92.5	**94.5**	46.0	4.0	92.5	**94.5**	46.0
	PSO-KDE	2.0	92.5(=)	**94.5**(=)	54.0(+)	2.0	92.5(=)	**94.5**(=)	54.0(+)

(d) 2-way linear

α	Method	Con-Size	Con-DT	Con-KNN	Con-NB	Dis-Size	Dis-DT	Dis-KNN	Dis-NB
	Full	4	99.5	95.5	52.0	4	99.5	95.5	52.0
0.6	PSO-C	1.0	80.4	76.13	48.0	1.0	80.4	76.13	48.0
	PSO-KDE	2.8	**99.5**(+)	95.0(+)	48.0(=)	2.8	**99.5**(+)	95.0(+)	48.0(=)
0.8	PSO-C	1.0	80.4	76.13	48.0	1.0	80.4	76.13	48.0
	PSO-KDE	2.8	**99.5**(+)	95.0(+)	48.0(=)	2.8	**99.5**(+)	95.0(+)	48.0(=)
1.0	PSO-C	4.0	**99.5**	95.5	52.0	4.0	**99.5**	95.5	52.0
	PSO-KDE	2.8	**99.5**(=)	95.0(-)	48.0(-)	2.8	**99.5**(=)	95.0(-)	48.0(-)

(e) 3-way linear

Table 3.4: Selected feature subsets (all indexes of features are in the curly brackets followed by the number of times the feature subset is selected).

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{0,1,2\} : 90\rangle$, $\langle\{0,1\} : 73\rangle$, $\langle\{1,2\} : 46\rangle$, $\langle\{2\} : 30\rangle$, $\langle\{0\} : 30\rangle$, $\langle\{1\} : 30\rangle$, $\langle\{0,2\} : 1\rangle$	$\langle\{0,1,2\} : 293\rangle$, $\langle\{0\} : 3\rangle$, $\langle\{1\} : 3\rangle$, $\langle\{2\} : 1\rangle$
α =0.8	$\langle\{0,1,2\} : 210\rangle$, $\langle\{0,1\} : 42\rangle$, $\langle\{0,2\} : 39\rangle$, $\langle\{1,2\} : 9\rangle$	$\langle\{0,1,2\} : 297\rangle$, $\langle\{1\} : 2\rangle$, $\langle\{2\} : 1\rangle$
α =1.0	$\langle\{0,1,2\} : 300\rangle$	$\langle\{0,1,2\} : 297\rangle$, $\langle\{2\} : 1\rangle$, $\langle\{0\} : 1\rangle$, $\langle\{1\} : 1\rangle$

(a) Binary 1 (Optimal subset: $\{f_0, f_1, f_2\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{0\} : 100\rangle$, $\langle\{2\} : 85\rangle$, $\langle\{0,1,2\} : 60\rangle$, $\langle\{1\} : 35\rangle$, $\langle\{1,2\} : 10\rangle$, $\langle\{0,1\} : 8\rangle$, $\langle\{0,2\} : 2\rangle$	$\langle\{0,1,2\} : 300\rangle$
α =0.8	$\langle\{0,2\} : 75\rangle$, $\langle\{0,1\} : 68\rangle$, $\langle\{0,1,2\} : 61\rangle$, $\langle\{1,2\} : 58\rangle$, $\langle\{2\} : 17\rangle$, $\langle\{0\} : 15\rangle$, $\langle\{1\} : 6\rangle$	$\langle\{0,1,2\} : 300\rangle$
α =1.0	$\langle\{0,1,2\} : 240\rangle$, $\langle\{2\} : 18\rangle$, $\langle\{0\} : 15\rangle$, $\langle\{1,2\} : 9\rangle$, $\langle\{0,1\} : 8\rangle$, $\langle\{1\} : 8\rangle$, $\langle\{0,2\} : 2\rangle$	$\langle\{0,1,2\} : 300\rangle$

(b) Binary 2 (Optimal subset: $\{f_0, f_1, f_2\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{4\} : 122\rangle$, $\langle\{3,4\} : 118\rangle$, $\langle\{1,4\} : 57\rangle$, $\langle\{2,4\} : 3\rangle$	$\langle\{0,1,4\} : 300\rangle$
α =0.8	$\langle\{0,3,4\} : 60\rangle$, $\langle\{1,2,4\} : 39\rangle$, $\langle\{4\} : 30\rangle$, $\langle\{1,4\} : 30\rangle$, $\langle\{3,4\} : 30\rangle$, $\langle\{0,2,4,5\} : 30\rangle$, $\langle\{3,4,5\} : 30\rangle$, $\langle\{1,2,3,4\} : 28\rangle$, $\langle\{2,3,4\} : 19\rangle$, $\langle\{2,4,5\} : 2\rangle$, $\langle\{1,3,4\} : 2\rangle$	$\langle\{0,1,4\} : 300\rangle$
α =1.0	$\langle\{0,1,2,3,4,5\} : 281\rangle$, $\langle\{0,1,2,3,4\} : 6\rangle$, $\langle\{1,2,3,4,5\} : 4\rangle$, $\langle\{0,2,3,4,5\} : 4\rangle$, $\langle\{0,1,3,4,5\} : 4\rangle$, $\langle\{0,1,2,4,5\} : 1\rangle$	$\langle\{0,1,4\} : 300\rangle$

(c) Monk 1 (Optimal subset: $\{f_0, f_1, f_4\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{0,1,3,4\} : 98\rangle$, $\langle\{0,1,2,3,4\} : 59\rangle$, $\langle\{0,1,2,3,4,5\} : 59\rangle$, $\langle\{0,1,3,4,5\} : 54\rangle$, $\langle\{0,1,3\} : 28\rangle$, $\langle\{0,1,4\} : 2\rangle$	$\langle\{0,1,2,3,4,5\} : 282\rangle$, $\langle\{0,2,3,4,5\} : 7\rangle$, $\langle\{0,1,2,3,5\} : 6\rangle$, $\langle\{0,1,2,4,5\} : 4\rangle$, $\langle\{0,1,2,3,4\} : 1\rangle$
α =0.8	$\langle\{0,1,2,3,4,5\} : 114\rangle$, $\langle\{0,1,3,4,5\} : 81\rangle$, $\langle\{0,1,2,3,4\} : 62\rangle$, $\langle\{0,1,3,4\} : 42\rangle$, $\langle\{0,1,3,5\} : 1\rangle$	$\langle\{0,1,2,3,4,5\} : 282\rangle$, $\langle\{0,2,3,4,5\} : 7\rangle$, $\langle\{0,1,2,3,5\} : 6\rangle$, $\langle\{0,1,2,4,5\} : 4\rangle$, $\langle\{0,1,2,3,4\} : 1\rangle$
α =1.0	$\langle\{0,1,2,3,4,5\} : 285\rangle$, $\langle\{0,1,2,3,4\} : 11\rangle$, $\langle\{0,1,3,4,5\} : 4\rangle$	$\langle\{0,1,2,3,4,5\} : 282\rangle$, $\langle\{0,2,3,4,5\} : 7\rangle$, $\langle\{0,1,2,3,5\} : 6\rangle$, $\langle\{0,1,2,4,5\} : 4\rangle$, $\langle\{0,1,2,3,4\} : 1\rangle$

(d) Monk 2 (Optimal subset: $\{f_0, f_1, f_2, f_3, f_4, f_5\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{1,3,4\} : 214\rangle$, $\langle\{1,3,4,5\} : 57\rangle$, $\langle\{1,2,3,4\} : 29\rangle$	$\langle\{1,3,4\} : 300\rangle$
α =0.8	$\langle\{1,2,3,4\} : 88\rangle$, $\langle\{1,3,4,5\} : 87\rangle$, $\langle\{1,3,4\} : 66\rangle$, $\langle\{0,1,2,3,4\} : 28\rangle$, $\langle\{0,1,3,4,5\} : 28\rangle$, $\langle\{0,1,3,4\} : 3\rangle$	$\langle\{1,3,4\} : 300\rangle$
α =1.0	$\langle\{0,1,2,3,4,5\} : 290\rangle$, $\langle\{0,1,2,3,4\} : 6\rangle$, $\langle\{1,2,3,4,5\} : 4\rangle$	$\langle\{1,3,4\} : 300\rangle$

(e) Monk 3 (Optimal subset: $\{f_1, f_3, f_4\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{0\}:110\rangle, \langle\{3\}:80\rangle, \langle\{2\}:70\rangle, \langle\{1\}:40\rangle$	$\langle\{2,3\}:110\rangle, \langle\{1,2\}:81\rangle, \langle\{0,1\}:60\rangle, \langle\{0,3\}:49\rangle$
α =0.8	$\langle\{0\}:110\rangle, \langle\{3\}:80\rangle, \langle\{2\}:70\rangle, \langle\{1\}:40\rangle$	$\langle\{2,3\}:110\rangle, \langle\{1,2\}:80\rangle, \langle\{0,1\}:60\rangle, \langle\{0,3\}:50\rangle$
α =1.0	$\langle\{0,1,2,3\}:300\rangle$	$\langle\{2,3\}:110\rangle, \langle\{1,2\}:80\rangle, \langle\{0,1\}:60\rangle, \langle\{0,3\}:50\rangle$

(f) 2-way linear (Optimal subset: $\{f_0, f_1\}, \{f_0, f_3\}, \{f_1, f_2\}, \{f_2, f_3\}$)

	PSO-C	**PSO-KDE**
α =0.6	$\langle\{0\}:110\rangle, \langle\{3\}:80\rangle, \langle\{2\}:70\rangle, \langle\{1\}:40\rangle$	$\langle\{1,2,3\}:144\rangle, \langle\{0,1,2\}:96\rangle, \langle\{0,1\}:32\rangle, \langle\{1,3\}:28\rangle$
α =0.8	$\langle\{0\}:110\rangle, \langle\{3\}:80\rangle, \langle\{2\}:70\rangle, \langle\{1\}:40\rangle$	$\langle\{1,2,3\}:144\rangle, \langle\{0,1,2\}:96\rangle, \langle\{0,1\}:32\rangle, \langle\{1,3\}:28\rangle$
α =1.0	$\langle\{0,1,2,3\}:300\rangle$	$\langle\{1,2,3\}:144\rangle, \langle\{0,1,2\}:96\rangle, \langle\{0,1\}:32\rangle, \langle\{1,3\}:28\rangle$

(g) 3-way linear (Optimal subset: $\{f_2\}$)

pletely reflected by PSO-KDE which selects the optimal subset {f_0, f_1, f_4} all the 300 times. Meanwhile, PSO-C selects very different feature subsets even within the same α values. On all α values, f_2 and f_3 appear frequently in the feature subsets, which indicates that the relevance calculated by the counting approach still gives some scores to these irrelevant features. An obvious evidence is that PSO-C selects all features when $\alpha = 1$, which means the irrelevant features are selected. For the Monk 2 dataset, it is important to select all original features. According to the experimental results, for all values of α, PSO-KDE always selects no less than 5 features, in which all features are selected more than 280 times out of the 300 times. Meanwhile, the size of feature subsets selected by PSO-C ranges from 3 to 6 features. In the Monk 3 dataset, the most complicated Monk dataset, the optimal feature subset is { f_1, f_3 ,f_4}, which is also selected by PSO-KDE in all cases regardless of the α values. Meanwhile, PSO-C still selects irrelevant features like f_0, f_2 and f_5 very frequently. With the Monk datasets, it can be seen that the counting-based approach, PSO-C, is not able to detect irrelevant features, which is done well by the estimation-based approach, PSO-KDE.

In the remaining two artificial datasets, the 2-way and 3-way linear

datasets, there is no irrelevant feature but there are redundant features. In the 2-way linear dataset, the class label can be determined by one of the following feature subsets {f_0, f_1}, {f_0, f_3}, {f_1, f_2} and {f_2, f_3}, which are also the only 4 feature subsets selected by PSO-KDE. On the other hand, PSO-C always selects a single feature when α is set to 0.6 or 0.8. Once more the result shows that the redundancy between two independent features is not correctly calculated by the counting approach. In addition, the KDE approach is able to detect the complementary feature subsets, although it is a hard problem when pair-wise fitness function is used. On the 3-way linear dataset, once more PSO-C always selects a single feature when α is less than 1.0. On the other hand, PSO-KDE selects only 4 feature subsets, which are {f_1, f_2, f_3}, {f_0, f_1, f_2}, {f_0, f_1} and {f_1, f_3}. PSO-KDE never selects f_0 and f_3 together because they are redundant. According to the linear datasets, KDE is able to detect the complementary feature subset and remove the redundant features, which can not be done by the counting approach.

The experimental results suggest that KDE for mutual information works well on both continuous and discrete datasets. The feature subsets generated by KDE achieve similar or better classification performance than the counting approach. The main reason is that the counting approach can not correctly calculate the redundancy measure and detect the complementary interaction between features, which can be achieved by KDE. The following simple example demonstrates that the counting approach fails to calculate redundancy in a continuous dataset.

As can be seen that, f_1 and f_2 are two independent variables, so their mutual information is expected to be 0. When the Counting approach is applied, it does not consider that 1.1 and 1.15 are two similar values (similarly 2.2 and 2.23). Therefore, under the Counting approach f_1 has 2 possible values, f_2 has 4 possible values. so $H(f_1) = 1$, $H(f_2) = 2$. The joint variables (f_1, f_2) has 4 possible values, so $H(f_1, f_2) = 2$. Thus, the Counting approach results in $MI_C(f_1, f_2) = 2 + 1 - 2 = 1 > 0$, which indicates that f_1

Table 3.5: Example of redundancy calculated by KDE and Counting.

f_1	f_2
3	1.1
3	2.2
4	1.15
4	2.23

and f_2 are not independent. When the KDE approach is used, it takes the distances between values into account, so 1.1 and 1.15 are considered one value (similarly 2.2 and 2.23). Therefore, f_2 has only 2 possible values, and $H(f_2) = 1$. Thus, the KDE approach results in $MI_{KDE}(f_1, f_2) = 1+1-2 = 0$, which indicates that f_1 and f_2 are independent.

3.4.3 Consistency of PSO-KDE and PSO-C

As can be seen from the results, on most datasets the order of classification accuracies is preserved after the feature selection process. For example, in the Vehicle dataset, the highest classification accuracy belongs to DT classifier and KNN is the second best classifier. After performing feature selection using either KDE or the counting approach, the best classifier is still DT, which is followed by KNN. The consistent results show that the mutual information measure does not produce features particularly bias to any classification algorithm. Mutual information is able to extract a general feature subset, which is meaningful to all the three classification algorithms. This is the property of filters, which is preserved by both PSO-KDE and PSO-C.

3.4.4 ANOVA Test Analysis

Since the counting and estimation approaches are compared on 12 different real-world datasets, using Wilcoxon test is not enough to confirm

Table 3.6: ANOVA test results

Factor	F value	Pr(>f)	Significant
Method	1757.806	<2e-16	*
Dataset	3963.685	<2e-16	*
TypeDataset	1018.247	<2e-16	*
Classifier	4939.826	<2e-16	*
Method : Dataset	239.405	<2e-16	*
Method : TypeDataset	37.153	1.12e-09	*
Method : Classifier	259.040	<2e-16	*

which method is better since this is a multi-test problem. An ANOVA test (Analysis of Variance), with a confidence interval of 0.95, is run to compare the two methods and shows the interactions between the two methods and other factors such as datasets, kinds of datasets and classification algorithms. The test results are shown in Table 3.6, in which "F value", "Pr(>f)" are the obtained *F-value* and *p-value*. If the *p-value* is less than the significance level (0.05), the corresponding factor or interacting factor has a significant effect on the classification accuracy which is shown by "*" in the "Significant" column. It can be seen that the interactions between methods and other factors produce significant different accuracies, which means KDE and the counting approach are significantly different on different datasets, different types of datasets and different classification algorithms.

In order to see which method is better, Fig 3.2 plots these interactions in terms of the classification accuracy. In the figures, "c" and "e" stands for "counting" and "KDE" methods, respectively. As can be seen from the figure, the "e" line is always on the top of the "c" line, which means that KDE is better than the counting approach in different levels of other factors such as datasets, types of datasets or classification algorithms.

3.4.5 Computational Cost

The computational costs of PSO-KDE and PSO-C are shown in Table 3.7. As can be seen from the table, PSO-KDE is more expensive than PSO-C.

Figure 3.2: Comparisons between two methods with different factors

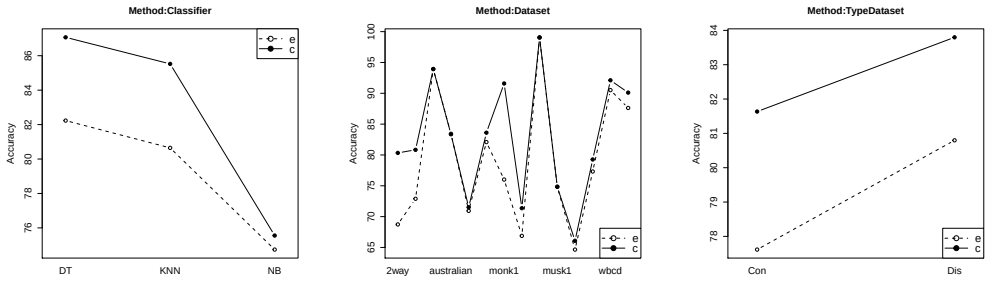


Table 3.7: Computational time on real-world datasets

Datset	**PSO-KDE (ms)**	**PSO-C (ms)**
Wine	344.49	38.74
Australian	74.19	1.61
Vehicle	244.7	1.64
German	145.45	2.19
WBCD	6288.96	88.69
Ionosphere	4941.29	98.97
Sonar	6819.77	187.77
Hillvalley	5584.8	7.93
Musk1	253546.83	424.43
Arrhythmia	4020.61	36.22
Madelon	926408.74	58.92
MultipleFeatures	786234.85	88.66
Binary 1	0.77	0.5
Binary 2	0.75	0.46
Monk 1	39.49	0.55
Monk 2	69.78	0.64
Monk 3	43.83	0.55
2-way linear	8.01	0.45
3-way linear	11.11	0.57

The reason is that in order to calculate the mutual information, KDE needs to calculate distances from each instance to all other instances, which is about N times slower than the counting approach (N is the total number of available instances).

3.5 Chapter Summary

The goal of this chapter is to propose a new mutual information-based fitness function for a PSO-based feature selection algorithm, which can work directly on continuous datasets without any pre-processing step such as discretization. To achieve this goal, instead of the traditional counting approach, kernel density estimation (KDE) is used to estimate mutual information. The experimental results show that KDE assists PSO to achieve similar or better classification performance than the counting approach on both continuous and discrete datasets.

This chapter shows that the mutual information estimation can overcome the limitation of the counting approach, which usually requires a large number of instances to derive reliable probability distributions for each feature. Mutual information estimation, specifically KDE, can capture the interactions between features more accurately than the counting approach. Particularly, KDE, which is originally proposed for continuous datasets, can work well on both continuous and discrete datasets. Meanwhile, the counting approach achieves good performance only on discrete datasets. In terms of detecting feature interactions, KDE is able to handle redundant and irrelevant features, which cannot be achieved by the counting approach. KDE can also detect complementary features, which assist PSO to select the optimal feature subsets on the artificial datasets.

However, in terms of the efficiency, the estimation method, KDE, is still slower than the counting approach. Since the estimation cost depends mainly on the number of instances, the estimation's efficiency can be improved if the number of instances decreases. In addition, removing noisy

instances may also increase the accuracy of the estimator. Therefore, it is important to develop instance selection algorithms along with feature selection algorithms. In addition, as can be seen from Table 3.2, when α is smaller than 1, the testing accuracy decreases because the number of selected features is too small. It is important to well balance the number of selected features and the classification accuracy. A solution to this problem is to develop multi-objective methods which can consider both objectives, including classification accuracies and the number of selected features. In terms of searching mechanisms, this chapter applies continuous PSO to achieve feature selection although feature selection is a binary combinatorial problem. In the next chapter, this problem will be addressed by a novel binary PSO algorithm which is expected to cope well with the characteristics of binary search spaces to evolve better solutions for binary problems.

Chapter 4

Novel Binary PSO for Feature Selection

4.1 Introduction

The original PSO is continuous PSO (CPSO) which is applied and extended to solve many continuous problems [200]. In CPSO, particles can move smoothly in a direction, so defining a velocity as a vector of real numbers is meaningful. PSO is also extended to solve binary problems i.e. binary PSO (BPSO) in which each particle's position is a vector of binary numbers. In standard BPSO [29], the velocity and momentum concepts of CPSO are directly applied to BPSO. However, in a binary search space, a particle moves by flipping its position entries. Such a movement is not correctly described as a velocity, and directly applying the notions of speed, direction, and momentum to the binary domain is not valid. This inappropriate application leads to BPSO's limited performance compared with CPSO [99].

As shown in Chapter 2, *velocity* in PSO has three main components: *momentum*, *cognitive*, and *social* factors. It is important to control contributions of the three components to balance between *exploitation* and *exploration* during the search process [201]. The exploration ability corresponds

to a tendency to discover new search regions, while the exploitation ability corresponds to finding the best solution within the current region. The balance between exploration and exploitation relates directly to the inertia weight and the two acceleration parameters [202], which control the momentum, cognitive and social components, respectively. In CPSO, a larger inertia weight, which gives a larger momentum, results in more exploration and a smaller inertia weight guides the swarm to focus more on exploitation. A control strategy, which starts with a high inertia weight and gradually decreases the weight, results in more exploration at the beginning and more exploitation at the end of each run, and has been widely used in CPSO.

However, the inertia weight in BPSO has an opposite effect, which is shown in [203] and theoretically proved in [97] (on the assumption that *pbest* and *gbest* are not changed). In contrast to CPSO, the velocity in BPSO does not directly determine the new position of a particle. It is used to derive the probability of the position entry being 1. This makes BPSO's movement very different from that of CPSO. In addition, since the movement to the new position ignores the previous location, one cannot say the new position is far or close to the previous location.

The velocity, momentum, exploration, and exploitation in BPSO need appropriate formations so that the particles can move through a binary search space in a meaningful way. However, no existing work is performed to specify the concepts in BPSO. The aim of this chapter is to develop a new BPSO algorithm in which the binary movement is reflected more accurately and the balance between exploration and exploitation is better controlled.

4.1.1 Chapter Goal

The overall goal of this chapter is to develop a new BPSO algorithm, which can consider properties of binary search spaces to explore the search space

more effectively and produce better solutions for binary problems. In order to achieve this goal, the key concepts, which are momentum and velocity, are revised so that the particles move around the search space using more effective mechanisms. In addition, a dynamic parameter setting strategy is developed to further enhance the proposed BPSO's search ability by considering the trade-off between exploration and exploitation. The proposed BPSO algorithm is compared with three EC algorithms, which are Genetic Algorithms (GAs) [79], Up BPSO [97] and Binary Differential Evolution (DE) [**?**]. The comparisons are performed on two types of well-known binary problems: knapsack and feature selection. Specifically, we will investigate the following objectives:

- Investigate whether applying the two revised concepts can help BPSO to evolve better solutions (i.e. item subsets) with higher profit in the knapsack problems;

- Investigate whether the new momentum and velocity can assist the particles to better explore the large and complex search space of feature selection, and can result in smaller feature subsets with higher classification performance;

- Investigate whether the proposed dynamic parameter setting strategy can balance between exploration and exploitation to further improve the search ability of BPSO on both knapsack and feature selection problems; and

- Analyze the effect of the two revised concepts and the dynamic strategy in terms of exploration and exploitation during the BPSO's search process.

4.2 Proposed Algorithm

This section presents the new momentum and velocity concepts for BPSO. It then defines exploration and exploitation capabilities to cope with the movement strategy of the particles. Based on the definitions of the two capabilities, a dynamic strategy is developed to better control the trade-off between these capabilities.

4.2.1 Sticky BPSO (SBPSO)

In standard BPSO [29], the new position is determined without considering the previous position, which can be seen in Eq. (4.1).

$$x_d^{t+1} = \begin{cases} 1, \textbf{ if } rand() \leq \dfrac{1}{1+e^{-v_d^{t+1}}} \\ 0, \textbf{ otherwise} \end{cases} \tag{4.1}$$

Particles do not move smoothly as in CPSO since they change their positions by flipping position entries either from 0 to 1 or from 1 to 0. This kind of probabilistic binary change cannot be usefully described as a continuous velocity. Rather it is better to describe the change in terms of the probability of flipping. Therefore, a flipping vector p is used in the proposed method instead of the velocity vector, in which each entry shows the probability of flipping the corresponding position entry.

To guide particles toward promising regions, the PSO velocity vector consists of three main components: momentum, cognitive and social factors. All the three factors need modification for the binary domain. Similar to velocity, momentum is a fundamental continuous concept and needs to be replaced by a more appropriate concept that still captures the role of momentum in CPSO. In CPSO, the momentum corresponds to a tendency to keep moving in the current direction. However in BPSO, instead of moving in a direction, a particle's movement is described as whether its entries are flipped or not. Therefore, in BPSO, the momentum is replaced

by a measure of the tendency to stick with the current position, which we call $stickiness$, or stk for short. The idea is that a high stickiness for an entry means that the particle should stick with the current value for a while, so that the particle can exploit around the entry's value, rather than switching to a different region of the space. If the stickiness of an entry is set to a high value and has not been changed for a while or never changes, the particle might get stuck in an unproductive region. Therefore, a strategy to control the stickiness is in need.

Currently, the stickiness of an entry is high when the entry is flipped and it is decayed over time until it is 0 or the entry flips again. A linear decay is used to reduce the stickiness from 1 to 0 over a fixed number of steps (*ustkS*). The stickiness property of the d^{th} bit is updated using:

$$stk_d^{t+1} = \begin{cases} 1, \textbf{ if} \text{ the bit is just flipped} \\ \max(stk_d^t - \dfrac{1}{ustkS}, 0), \textbf{ otherwise} \end{cases} \tag{4.2}$$

where t means the t^{th} iteration. By using the above way, it can achieve the idea of stickiness, i.e. if one bit is not flipped for a number of iterations, its stickiness value stk_d will become 0, which significantly increases the probability of this bit to be flipped (*un-stick*) in the next iteration. The number of iterations or steps, which makes a bit un-stick with its current value, is called $ustkS$.

The cognitive and social factors (based on $pbest$ and $gbest$, respectively) are still important – they guide the particle towards the regions containing $pbest$ and $gbest$. However, the acceleration factors (and the random multipliers) are only appropriate in the continuous domain. For binary search spaces, we need important weights that increase the flipping probability when the current position is different from the $pbest$ and $gbest$. Using the stickiness property in place of the momentum factor and the modified cognitive and social factors, the flipping probability of a particle's d^{th} position entry is designed as shown in Eq. (4.3).

$$p_d = i_s \times (1 - stk_d) + i_p \times |pbest_d - x_d| + i_g \times |gbest_d - x_d| \tag{4.3}$$

where i_s, i_p, and i_g are the importance of the $stickiness$, the cognitive and social factors, respectively. As shown in Eq. (4.3), if $gbest$ and $pbest$ are not changed, the smaller the $stickiness$ the more likely the d^{th} bit will be flipped, which gives a high flipping probability to the bit that is not changed for a large number of iterations.

According to the flipping probability vector, the new position is determined by Eq. (4.4), which does consider the previous position to determine the new position.

$$x_d^{t+1} = \begin{cases} 1 - x_d^t & , \textbf{if } rand() < p_d \\ x_d^t & , \textbf{otherwise} \end{cases} \tag{4.4}$$

4.2.2 Exploration and Exploitation in SBPSO

In CPSO, a velocity shows how far a particle is going to move from the current position to the new position. A large velocity facilities exploration while a small one leads to more exploitation. In a binary search space, Hamming distance can be used to measure the distance between two binary solutions, which is the number of bits that the two solutions are different. If a large number of bits are mutated, the particle is exploring the search space. On the other hand, a few bits being flipped means the particle exploits the region around the current position. However, this is not clearly reflected in the standard BPSO algorithm [29] since the updating equation ignores the previous position. In contrast, the difference between the new position and previous position is naturally shown by the flipping operation in SBPSO. As can be seen in Eq. (4.4), p_d shows the probability of flipping the d^{th} position entry, which means the larger p_d the more likely the entry is flipped. Therefore, more bits to be flipped results in the particle towards exploration and a smaller probability vector lets the particle focus more on exploitation. Based on the defined exploration and exploitation terms in SBPSO, a dynamic strategy is proposed in the next section to balance the two abilities in SBPSO.

4.2.3 Dynamic Strategy

As can be seen from Eq. (4.2) and Eq. (4.3), the flipping probability is affected by four main parameters, which are i_s, i_p, i_g and $ustkS$. During the evolutionary process, there might be four possible relationships between a current position and two best positions, $pbest$, and $gbest$, in each bit. Therefore, Eq. (4.3) can be written as follows based on the four possible relationships.

$$p_d = \begin{cases} i_s \times (1 - stk_d) & \textbf{if } x_d = pbest_d = gbest_d \\ i_s \times (1 - stk_d) + i_g & \textbf{if } x_d = pbest_d \neq gbest_d \\ i_s \times (1 - stk_d) + i_p & \textbf{if } x_d = gbest_d \neq pbest_d \\ i_s \times (1 - stk_d) + i_p + i_g & \textbf{if } x_d \neq pbest_d = gbest_d \end{cases} \tag{4.5}$$

Since Eq. (4.5) represents the flipping probability of a single bit, it is not possible to state exactly whether a case happens at the beginning, the middle or at the end of the evolutionary process. However, it is more likely that $pbest_d \neq gbest_d$ when the searching process has just started and $pbest_d = gbest_d$ at the end of the evolutionary process when the swarm starts converging. Suppose that $gbest$ and $pbest$ have the same contribution to particles' movements as in the CPSO algorithm, which means that $i_p = i_g$. The largest value of p_d is $(i_s + i_p + i_g)$ when a bit is different from the bit's value in both $pbest$ and $gbest$ for a number of iterations without any improvement. Therefore, $(i_s + i_p + i_g)$ is set to 1, which ensures the bit is flipped to match the values of $pbest$ and $gbest$. According to the above conditions, both i_p and i_g have the same value which is $(1 - i_s)/2$ or $(0.5 - 0.5 * i_s)$. By substituting the values into Eq. (4.5), the equation can be rewritten as below:

$$p_d = \begin{cases} i_s \times (1 - stk_d) & \textbf{if } x_d = pbest_d = gbest_d \\ 0.5 + i_s \times (0.5 - stk_d) & \textbf{if } pbest_d \neq gbest_d \\ 1 - i_s \times stk_d & \textbf{if } x_d \neq pbest_d = gbest_d \end{cases} \tag{4.6}$$

where stk_d is calculated based on $ustkS$ using Eq. (4.2). Therefore, the flipping probability now mainly depends on two parameters i_s and $ustkS$. For a specific value of i_s, a smaller $ustkS$ results in a larger p_d, more exploration, while a larger $ustkS$ guides the swarm towards exploitation because the flipping probability is smaller. On the other hand, suppose that $ustkS$ is not changed, except for the case ($x_d \neq pbest_d = gbest_d$), decreasing i_s makes p_d smaller, which guides the swarm to exploit more. A static setting for i_s and $ustkS$ might just encourage either exploration or exploitation during the evolutionary process, which results in either missing optimal solutions or stucking at local optima. Therefore, a dynamic setting mechanism for i_s and $ustkS$ is proposed to allow the search process to change gradually from exploration to exploitation during the evolutionary process. Particularly, in the proposed mechanism, i_s is linearly decreased and $ustkS$ is linearly increased with respect to the number of iterations, which can be seen in Eq. (4.7).

$$\begin{aligned} ustkS^t &= ustkS^L + \frac{t}{T} \times (ustkS^U - ustkS^L) \\ i_s^t &= i_s^U - \frac{t}{T} \times (i_s^U - i_s^L) \end{aligned} \tag{4.7}$$

where t stands for the t^{th} iteration, T is the maximum number of iterations, $ustkS^U$ and $ustkS^L$ stand for the upper bound and the lower bound of $ustkS$, i_s^U and i_s^L are the upper bound and the lower bound of i_s. Since each bit is likely to be flipped after $ustkS$ iterations, $ustkS$ should be smaller than the length of a particle, which ensures that there is at least one bit being flipped in $ustkS$ iterations. Therefore, $ustkS^L$ and $ustkS^U$ are roughly bounded by the particle's dimension.

4.2.4 Overall Structure

The dynamic strategy is applied to propose a new SBPSO algorithm, called Dynamic SBPSO, which is shown in Fig. 4.1. The standard SBSPO without the dynamic mechanism is called Static SBPSO. The difference between

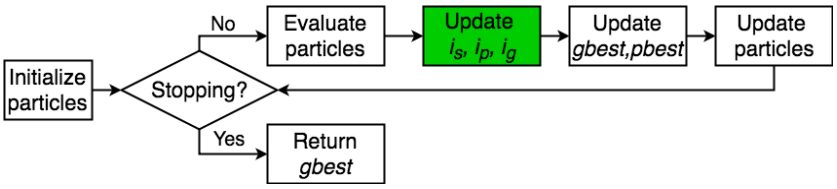


Figure 4.1: Dynamic SBPSO Overview.

Static and Dynamic SBPSO is the green block, which updates the three important weights.

The pseudo-code of Dynamic SBPSO is given in Algorithm 2. In the pseudo-code, the green line is the difference between the Dynamic and Static SBPSO, where the parameters of Dynamic SBPSO is updated using Eq. (4.7). The pseudo-code of Static SBPSO can be obtained by removing the green line.

Algorithm 2 : Pseudo-code of Dynamic Sticky BPSO (SBPSO)

randomly initialize the position of each particle;
while Maximum iteration is not reached **do**
  for $i = 1$ to $PopulationSize$ **do**
    evaluate the fitness of particle i;
    update the $pbest$ of particle i;
  end for
  update $gbest$ for each particle;
  for $i = 1$ to $PopulationSize$ **do**
    update the stickiness property using Eq. (4.2);
    update the flipping probability p of the i^{th} particle using Eq. (4.3);
    update the position of the i^{th} particle using Eq. (4.4);
  end for
  update parameters of SBPSO according to Eq. (4.7);
end while
output the solution represented by $gbest$;

4.3 Experiment Design

In this chapter, the Dynamic (Dyn) and Static SBPSO (Stat) are compared with GAs, a state-of-the-art dynamic BPSO algorithm named Up BPSO (Up) [97] and binary DE (DE) [204] - a recently proposed algorithm. A statistical significance test, called Wilcoxon test with a confidence interval of 0.95, is used to compare their performance.

4.3.1 Benchmark Problems

Static and Dynamic SBPSO are compared with other algorithms on two well-known binary problems which are knapsack and feature selection. Knapsack is a traditional binary optimization problem, which can be described as follows: given a set of n items and a knapsack; each d^{th} ($d = 1, \ldots, n$) item has a positive profit pf_d and a number of positive resource consumptions $r_{d1}, r_{d2}, \ldots, r_{dm}$ corresponding to m resources; the knapsack has m capacities $C_j (j = 1, \ldots, m)$ for each resource; the task is to select a subset of items so that the total profit of the selected items is maximized while for each resource, the total resource consumption does not exceed the resource's capacity. The problem can be described using the following formula:

$$
\begin{aligned}
& max \sum_{d=1}^{n} pf_d \times x_d, \\
& s.t. : \sum_{d=1}^{n} r_{dj} \times x_d \leq C_j, \forall j \in \{1, 2, \ldots, m\} \\
& x_d \in \{0, 1\}, \forall d \in \{1, 2, \ldots, n\}
\end{aligned}
$$

where the d^{th} item is selected if and only if $x_d = 1$. Knapsack has been widely used to evaluate many evolutionary algorithms such as PSO [205, 206, 207, 208], DE [209, 210], GAs [211, 212], and multi-objective algorithms [213].

Knapsack's fitness function is quite easy to calculate. Therefore, it is chosen as an initial benchmark problem to evaluate the proposed algorithm. Feature selection is another combinatorial problem, which is more difficult than knapsack because of its computationally intensive fitness function and the complex interactions between features.

4.3.2 Parameter Settings

In this chapter, a PSO-based algorithm is developed to tune the parameters of SBPSO algorithms. The parameter is tuning on 5 difficult Knapsack datasets with different numbers of items and capacities, which are Gk01, Pet7, Sento2, Weing8 and Weish30 [210]. Knapsack is chosen because of its low computational cost. Note that the tuning process does not aim to find a perfect parameter settings working on all problems. The aim is to have a good enough setting for SBPSO so that its comparisons with other well-known benchmark algorithms are relatively fair.

Since it is assumed that $i_m + i_p + i_g = 1$, the task is to optimize two parameters i_m and i_g, from which i_p can be obtained easily. However, from the assumption, there is a constraint that $i_m + i_g \leq 1$, which must be maintained during the evolutionary process. Therefore, in order to make an easy optimization where all parameters to be optimized are in the range [0, 1], instead of finding an exact value of i_g, we aim to find the ratio of i_p to i_g, called α, which means that $i_p = \alpha \times i_g$. By tuning the two parameters i_m and α, the other two parameters i_p and i_g can be obtained as below:

$$i_g = \frac{1 - i_m}{1 + \alpha} \tag{4.8}$$

$$i_p = \alpha \times i_g = \frac{\alpha \times (1 - i_m)}{1 + \alpha} \tag{4.9}$$

In dynamic SBPSO, i_m decreases from i_m^U to i_m^L, and $ustkS$ increases from $ustkS^L$ to $ustkS^U$. We set 0.05 and 5 as initial values for $ustkS^L$ and i_m^L, respectively. The two values are selected so that a position entry still

Table 4.1: Parameter settings evolved by OP-PSO on each dataset.

Setting	Dataset	i_m	α	$ustkS$
S1	Gk01	0.7684	1.00	46
S2	Sento2	0.3882	0.99	26
S3	Pet7	0.4984	1.00	36
S4	Weing8	0.6052	0.93	40
S5	Weish30	0.1154	1.0	40

has a small flipping probability when its value is equal to the entry value in both $gbest$ and $pbest$. Instead of finding $ustkS^U$, we find $ustkS_{scale} = (ustkS^U - ustkS^U_{min})/(ustkS^U_{max} - ustkS^U_{min})$ which ensures all the variables to be optimized are in the same range [0,1]. In this chapter, $ustkS^U_{min}$ and $ustkS^U_{max}$ are set to 20 and 60, since the five Knapsack datasets have at least 60 items.

In summary, there are three parameters being optimized, which are i_m, α and $ustkS_{scale}$. Since the three parameters are in the range [0,1], a standard continuous PSO is used to tune them. The particle's length is 3, in which each position entry corresponds to one parameter. For the sake of convenience, the parameter's optimization algorithm is named OP-PSO.

In OP-PSO, each particle can be considered a parameter setting strategy, which is applied to SBPSO to find an item subset with the highest total profit. Each SBPSO algorithm with the specific parameter setting from OP-PSO is run 10 times and each run contains 30 iterations. The average of total profit over the 10 runs is returned as the fitness value of the particle. Since OP-PSO runs on 5 different datasets, 5 different parameter settings are returned, which can be seen in Table 4.1.

In order to select a setting among the five evolved settings, each settings is further tested on all the five datasets. After that, on each dataset, the settings are sorted according to the profit obtained on the dataset, which means that a setting can be assigned a rank from 1 to 5. The smaller the rank, the better the corresponding setting. The average ("Average") and standard deviations ("StD") of the five settings on five datasets are shown in Table 4.2. As can be seen from the table, "S5" has the best (low-

Table 4.2: Ranks of parameter settings on the five datasets.

Setting	Average	StD
S1	1.4	1.74
S2	3.4	0.49
S3	1.8	1.33
S4	2.2	1.17
S5	**1.2**	**0.75**

Table 4.3: Parameter settings of PSO algorithms.

Algorithm	Setting
Up BPSO	$\overline{w} = 1.0$; $\underline{w} = 0.4$; $c_1 = c_2 = 2.0$ v_{max} = 6.0, v_{min} = -6.0
Static SBPSO	$i_s = 0.1154$; $i_p = i_g = 0.4423$; $ustkS = 40$
Dynamic SBPSO	$i_p = i_g = 0.5 * (1 - i_s)$ $i_s^U = 0.1154$; $i_s^L = 0.05$ $ustkS^U = 40$; $ustkS^L = 5$

est) average ranking with a low standard deviation. Although "S1" and "S4" have quite good average rankings, their standard deviations are high illustrating they are less stable than "S5". Therefore, "S5" is chosen as the parameter setting for SBPSO algorithms. Notice that the optimal α is 1, which means the assumption $i_p = i_g$ in the previous section is accurate.

The parameters of Up BPSO are set according to the original paper [97]. The parameter settings of PSO-based algorithms are given in Table 4.3. In binary DE, the crossover rate and scale factor are set as 0.25 and 1.0, respectively [204]. In GAs, the mutation rate is $1/n$, where n is the number of bits in each individual. The crossover probability is 0.9 [214].

4.4 Experiments on Knapsack

4.4.1 PSO for Knapsack

On knapsack problems, the five algorithms are compared on 30 knapsack datasets selected from 5 Knapsack instances [210] with different numbers of items and resources. The four instances Weish (10-30), Weing (7-8),

Sento (1-2), Pet (7) are from the SAC-94 Knapsack library with n ranging from 10 to 105 and m ranging from 2 to 30. The second instance was provided by Glover and Kochenberger [215], named as "Gk", which have larger n (from 100 to 150) and m (from 15 to 50) than SAC-94. For each dataset, each algorithm is run 50 independent times, and each run contains 3000 iterations. The swarm size is equal to the number of items as in [97].

In terms of representation, each position entry corresponds to one item, which means that a particle's length is the total number of items. In the five algorithms, the position entry is 1 or 0, which shows that the corresponding item is selected or discarded. A penalty function strategy is used to transform the problem's binary constraint into a fitness function as below:

$$fitness_{KS} = \sum_{d=1}^{n} p_d \times x_d + \beta \times \sum_{j=1}^{m} min(C_j - \sum_{d=1}^{n} r_{dj} \times x_d, 0) \qquad (4.10)$$

where β is used to penalize any infeasible candidate solution. β is set to 10^6, which is large enough to ensure that the final item subset is feasible. The aim of PSO is to maximize the fitness function given in Eq. (4.10).

4.4.2 Profits

Table 4.4 shows the average and standard deviation of the results of five algorithms on 30 knapsack datasets, where the number of items (n) ranges from 50 to 150 items and the number of resources (m) varies from 2 to 50. "Opt" represents either the optimal or the best known profit on each dataset. "Hit Rate" shows how many times an algorithm evolves a solution with the same profit as the optimal or best known solution. "Average (Std)" are the averages and standard deviations of profits obtained by each algorithm in the 50 independent runs. On each dataset, the best (highest) hit rate and average profit are marked with bold letters. The results are presented by a Win/Draw/Lost (W/D/L) ratio, which are the number of

Table 4.4: Experimental results on knapsack.

Dataset	m	n	Opt	Hit Rate					Average (Std)				
				GAs	DE	Up	Stat	Dyn	GAs	DE	Up	Stat	Dyn
Gk01	15	100	3.77E3	0	0	0	0	0	3.58E3(2.4E1)	3.69E3(9.6E0)	3.71E3(1.1E1)	3.72E3(1.2E1)	**3.73E3(9.6E0)**
Gk02	25	100	3.96E3	0	0	0	0	0	3.77E3(1.9E1)	3.88E3(9.3E0)	3.90E3(1.2E1)	3.91E3(1.3E1)	**3.92E3(1.1E1)**
Gk03	25	150	5.66E3	0	0	0	0	0	5.42E3(1.9E1)	5.52E3(1.3E1)	5.56E3(1.5E1)	5.56E3(1.3E1)	**5.59E3(1.2E1)**
Gk04	50	150	5.73E3	0	0	0	0	**1**	5.55E3(1.8E1)	5.64E3(1.1E1)	5.67E3(1.3E1)	5.67E3(1.4E1)	5.**69E3(1.5E1)**
Pet7	5	50	1.65E4	0	**31**	7	3	3	1.56E4(3.3E2)	**1.65E4(9.0E0)**	**1.65E4(3.9E1)**	1.64E4(7.0E1)	**1.65E4(4.4E1)**
Sento1	30	60	7.77E3	0	**33**	10	17	23	4.68E3(8.5E2)	**7.77E3(8.0E0)**	7.74E3(2.9E1)	7.74E3(3.6E1)	7.75E3(3.0E1)
Sento2	30	60	8.72E3	0	**14**	2	1	3	7.67E3(4.2E2)	**8.72E3(5.3E0)**	8.70E3(1.7E1)	8.71E3(1.2E1)	8.71E3(1.2E1)
Weing7	2	105	1.10E6	0	1	0	0	**3**	1.02E6(2.1E4)	1.09E6(6.0E2)	1.09E6(1.4E3)	**1.10E6(4.7E2)**	**1.10E6(3.4E2)**
Weing8	2	105	6.24E5	0	0	1	**3**	2	2.53E5(6.0E4)	6.15E5(6.7E3)	6.19E5(4.7E3)	**6.20E5(1.3E3)**	**6.20E5(1.1E3)**
Weish10	5	50	6.34E3	0	**47**	32	38	42	4.67E3(6.5E2)	**6.34E3(4.2E0)**	**6.33E3(1.9E1)**	**6.33E3(1.7E1)**	**6.33E3(1.6E1)**
Weish11	5	50	5.64E3	0	**48**	16	16	20	3.64E3(7.7E2)	**5.64E3(7.8E-1)**	5.62E3(3.4E1)	5.60E3(4.7E1)	5.62E3(3.9E1)
Weish12	5	50	6.34E3	0	**50**	36	46	46	4.49E3(6.8E2)	**6.34E3(0.0E0)**	6.33E3(1.8E1)	**6.34E3(2.7E-1)**	**6.34E3(9.9E0)**
Weish13	5	50	6.16E3	0	**50**	44	48	46	4.13E3(6.9E2)	**6.16E3(0.0E0)**	**6.15E3(2.5E1)**	**6.16E3(1.5E1)**	**6.15E3(2.3E1)**
Weish14	5	60	6.95E3	0	**50**	37	40	40	4.71E3(6.1E2)	**6.95E3(0.0E0)**	**6.94E3(2.4E1)**	**6.95E3(1.2E1)**	**6.95E3(2.0E1)**
Weish15	5	60	7.49E3	0	**50**	29	48	49	4.89E3(5.8E2)	**7.49E3(0.0E0)**	7.47E3(2.4E1)	**7.48E3(7.5E0)**	**7.49E3(4.5E0)**
Weish16	5	60	7.29E3	0	**40**	16	17	25	5.05E3(6.1E2)	**7.29E3(7.4E-1)**	7.28E3(1.1E1)	7.28E3(2.6E1)	7.28E3(1.9E1)
Weish17	5	60	8.63E3	0	**49**	27	34	38	7.79E3(3.1E2)	**8.63E3(2.0E0)**	8.63E3(6.8E0)	8.63E3(5.1E0)	8.63E3(5.0E0)
Weish18	5	70	9.58E3	0	**43**	13	16	18	7.94E3(4.4E2)	**9.58E3(2.4E0)**	9.56E3(1.5E1)	9.57E3(1.2E1)	9.57E3(9.7E0)
Weish19	5	70	7.70E3	0	**47**	21	28	32	5.14E3(6.6E2)	**7.70E3(3.1E0)**	7.67E3(3.4E1)	7.69E3(1.1E1)	7.69E3(8.8E0)
Weish20	5	70	9.45E3	0	**44**	22	45	37	7.18E3(5.9E2)	**9.45E3(1.9E0)**	9.44E3(1.8E1)	9.45E3(7.3E0)	9.45E3(7.7E0)
Weish21	5	70	9.07E3	0	**46**	24	28	34	6.65E3(5.8E2)	**9.07E3(8.2E0)**	9.06E3(2.0E1)	9.06E3(2.2E1)	9.06E3(2.2E1)
Weish22	5	80	8.95E3	0	**20**	11	18	15	6.31E3(6.6E2)	**8.93E3(1.4E1)**	8.91E3(3.4E1)	8.92E3(2.6E1)	8.92E3(2.7E1)
Weish23	5	80	8.34E3	0	**26**	7	11	12	5.59E3(6.6E2)	**8.34E3(2.0E1)**	8.31E3(3.6E1)	8.33E3(2.5E1)	8.33E3(1.8E1)
Weish24	5	80	1.02E4	0	**31**	11	29	26	8.54E3(5.2E2)	**1.02E4(1.1E1)**	1.02E4(2.3E1)	**1.02E4(1.6E1)**	**1.02E4(1.3E1)**
Weish25	5	80	9.94E3	0	**25**	7	13	11	7.72E3(5.8E2)	**9.93E3(8.9E0)**	9.92E3(1.6E1)	9.92E3(1.1E1)	9.92E3(9.6E0)
Weish26	5	90	9.58E3	0	15	5	9	**27**	6.45E3(7.9E2)	9.56E3(2.1E1)	9.54E3(2.2E1)	9.56E3(1.8E1)	**9.57E3(2.0E1)**
Weish27	5	90	9.82E3	0	35	27	**46**	44	6.72E3(6.4E2)	**9.81E3(1.7E1)**	9.80E3(4.5E1)	**9.81E3(3.6E1)**	**9.81E3(3.1E1)**
Weish28	5	90	9.49E3	0	**34**	23	35	31	6.55E3(6.7E2)	**9.48E3(2.3E1)**	9.46E3(3.6E1)	**9.48E3(1.8E1)**	**9.48E3(2.3E1)**
Weish29	5	90	9.41E3	0	23	12	24	**26**	6.29E3(8.1E2)	9.38E3(5.2E1)	9.36E3(4.1E1)	9.38E3(3.6E1)	**9.39E3(3.1E1)**
Weish30	5	90	1.12E4	0	22	12	22	**26**	8.90E3(5.9E2)	**1.12E4(1.5E1)**	1.12E4(1.8E1)	**1.12E4(1.2E1)**	**1.12E4(1.2E1)**

Table 4.5: W/D/L on knapsack.

Algorithm	GAs	DE	Up	Stat
Stat	30/0/0	5/10/15	17/12/1	
Dyn	30/0/0	6/10/14	21/8/1	7/23/0

times that an algorithm is significantly better, similar or worse than the other. Table 4.5 shows the comparisons between SBPSO algorithms and other algorithms in terms of W/D/L ratio.

As seen from Table 4.4, on **all** the datasets, both static and dynamic SBPSO obtain solutions with higher average profit than GAs. In comparison with Up BPSO, on most datasets (29 out of the 30 datasets), the static SBPSO algorithm achieves similar or better average profit. Static SBPSO also has higher hit rates on most of the 30 datasets. For example on Weish27, Static SBPSO's hit rate is almost two times higher than that of Up BPSO. According to the significance test results given in Table 4.5, static SBPSO is significantly better than Up BPSO on 17 datasets while achieving similar profits on 12 datasets. The results show that applying the stickiness property as BPSO's momentum, SBPSO can better explore the search space to evolve item subsets with higher profits than Up BPSO.

Similar to static SBPSO, dynamic SBPSO also achieves similar or better performance than Up BPSO on 29 out of the 30 datasets. According to Table 4.5, dynamic SBPSO is significantly better than Up BPSO on 21 datasets while being worse than Up BPSO on only one dataset. In comparison with static SBPSO, dynamic SBPSO achieves significantly higher profits on seven datasets, which contain a large number of items, such as the four Gk datasets. On most datasets, dynamic SBPSO has a smaller standard deviation than static SBPSO, which shows that dynamic SBPSO is also more stable.

As can be seen from Table 4.4, on the datasets with less than 80 items, DE outperforms all the other algorithms with higher profits and hit rates. However, when the number of items is increased, dynamic SBPSO achieves significantly higher profits than binary DE. Especially, on the four Gk datasets,

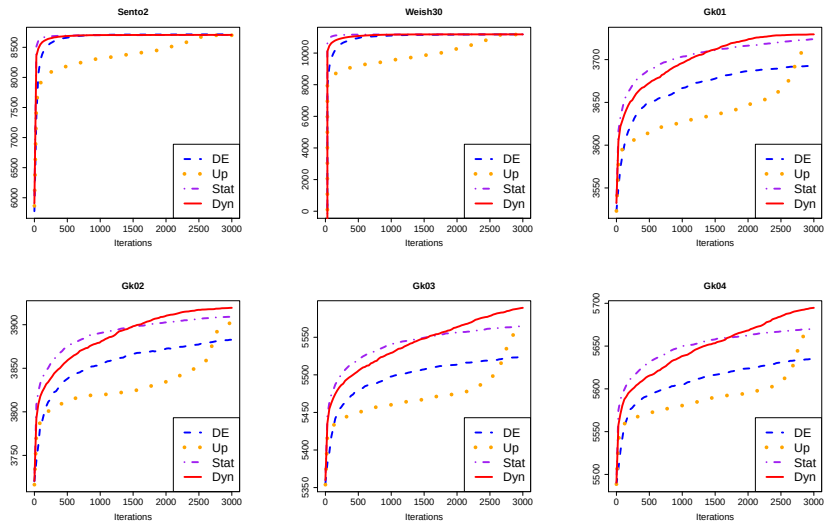


Figure 4.2: Evolutionary process of the four algorithms on 3000 iterations.

dynamic SBPSO outperforms all the other algorithms. On Gk04, the most difficult dataset, only dynamic SBPSO can achieve an optimal profit.

4.4.3 Evolutionary Processes

In order to examine the search ability of the five algorithms, their evolutionary processes are recorded. The evolutionary process is obtained by averaging the best fitness value achieved by the whole population in each iteration over the 50 independent runs. For example, if each run contains 400 iterations, the evolutionary process consists of 400 average values. To clearly show the difference between the other four algorithms, the evolutionary process of GA is not shown since GA's profit is much lower than the other algorithms. The evolutionary processes of the four methods are shown in Fig. 4.2.

As can be seen in Fig. 4.2, the static SBPSO algorithm is consistently superior to Up BPSO even in the last 600 iterations. Although binary DE is significantly better than the two SBPSO algorithms on the Sento2 dataset, the difference between them is very small. Meanwhile, on the Gk datasets,

where the dynamic SBPSO algorithm outperforms binary DE, and the difference between the two algorithms can be seen clearly during their evolutionary process.

The principle of SBPSO is to explore the search space firstly. Once it finds a promising region, it will try to exploit that region for a while before continue exploring other regions. In SBPSO, the exploration and exploitation are performed alternately. On the other hand, Up BPSO aims to explore the search space first to find promising areas before focusing on exploitation in the later iterations. Probably because of doing exploitation earlier than Up BPSO, both SBPSO algorithms achieve better profit than Up BPSO in the first 2000 iterations. After that, Up BPSO starts exploiting the discovered promising areas, which leads to a significant jump in the last 1000 iterations. On the other hand, by alternatively performing exploration and exploitation, the performance of SBPSO algorithms is steadily improved. Finally, at the end, the SBPSO algorithms outperform Up BPSO.

4.4.4 Evolutionary Processes with 6000 Iterations

On the four Gk datasets, after 3000 iterations, Up BPSO still has a tendency to further improve the solutions. Therefore, we perform another set of experiments, in which each algorithm is run another 50 independent times and each run contains 6000 iterations, which is roughly two times longer than the previous experiments. The evolutionary process on 6000 iterations can be seen in Fig. 4.3. It can be seen that even when the number of iterations is increased, the Up BPSO algorithm still has the same pattern as it is run for 3000 iterations. The reason is that the parameters of Up BPSO linearly change with respect to the number of iterations, which means that in the first 70% iterations, Up BPSO still performs exploration to search for promising areas. Only in the last 30% iterations, Up BPSO can significantly improve its solutions, but they are not as good as the solutions evolved by the SBPSO algorithms.

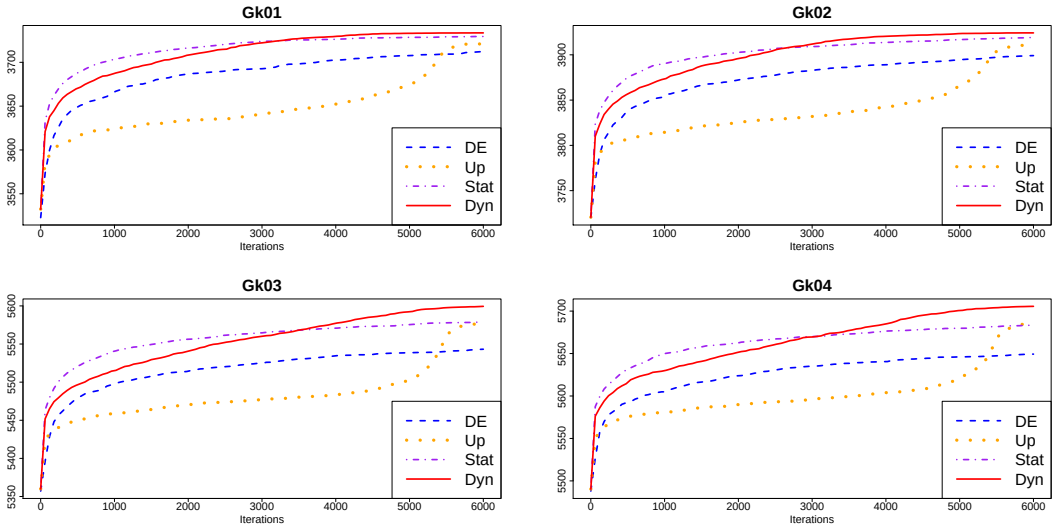


Figure 4.3: Evolutionary process of the four algorithms on 6000 iterations.

4.4.5 Effect of Dynamic Strategy

To analyze the effect of dynamic strategy, we compare static with dynamic SBPSO. Dynamic SBPSO starts with the smallest $ustkS$ and the largest value of i_s in order to have a larger flipping probability than static SBPSO. Therefore, in the beginning, static SBPSO explores less and spends more time on exploitation than dynamic SBPSO. On large datasets like the Gk datasets, in the first half of the evolutionary process, the candidate solutions evolved by dynamic SBPSO are not as good as that of static SBPSO. However, in the following iterations, when the search phase moves to exploit the promising areas, dynamic SBPSO shows its good exploitation ability to search for superior solutions and leaves static SBPSO behind.

The experimental results show that the stickiness property helps the particles to move around the search space in a more meaningful way and allows the swarm to do exploration and exploitation alternatively. This searching behavior results in better performance than doing exploration and exploitation sequentially as in Up BPSO. In addition, the dynamic strategy assists SBPSO to focus more on exploration at the beginning and

Table 4.6: Computational time (in seconds) on knapsack.

Dataset	GAs	DE	Up	Stat	Dyn	Dataset	GAs	DE	Up	Stat	Dyn
Gk01	**1.87**	12.75	4.24	2.04	2.20	Weish16	**0.62**	4.21	1.52	0.68	0.77
Gk02	**2.26**	13.04	4.55	2.29	2.44	Weish17	**0.55**	4.04	1.59	0.71	0.94
Gk03	**3.75**	29.52	10.10	4.84	5.38	Weish18	**0.76**	6.06	2.98	1.18	1.01
Gk04	**5.81**	30.89	11.69	5.95	6.23	Weish19	**0.70**	5.80	2.14	0.94	0.97
Pet7	0.76	2.98	1.08	**0.55**	0.73	Weish20	**0.71**	6.00	2.98	1.11	1.03
Sento1	**0.97**	4.36	1.88	1.11	1.15	Weish21	**0.75**	5.94	1.95	0.90	1.02
Sento2	**0.92**	4.57	1.89	1.09	1.03	Weish22	**0.84**	7.87	2.60	1.47	1.32
Weing7	**1.02**	13.07	5.24	1.86	2.63	Weish23	**0.84**	7.65	2.60	1.47	1.29
Weing8	**0.98**	12.97	3.97	2.21	1.83	Weish24	**0.84**	7.99	3.79	1.16	1.26
Weish10	0.55	2.75	1.06	**0.50**	0.55	Weish25	**0.79**	7.89	2.78	1.18	1.24
Weish11	0.58	2.73	1.06	**0.50**	0.55	Weish26	**1.04**	9.90	3.46	1.43	1.87
Weish12	0.54	2.78	1.06	0.67	**0.53**	Weish27	**1.09**	9.87	4.61	2.16	1.54
Weish13	0.60	2.77	1.16	**0.52**	0.55	Weish28	**1.00**	9.74	3.46	1.43	1.98
Weish14	**0.68**	4.16	1.51	**0.68**	0.74	Weish29	**0.99**	9.84	3.15	1.42	1.85
Weish15	**0.67**	4.27	2.19	0.69	0.74	Weish30	**0.88**	10.19	5.11	1.87	2.00

steadily exploit more at the end. Therefore, dynamic SBPSO outperforms both static SBPSO and Up BPSO.

4.4.6 Computational Time

The computational times of the five algorithms are shown in Table 4.6. In the table, the shortest computational time on each dataset is marked in bold. As can be seen from the table, on most of the datasets, GAs is the most efficient algorithm however its performance is not as good as other algorithms. Although DE has a good performance on datasets with small numbers of items, its computational cost is the most expensive. Static SBPSO is the second most efficient algorithm on 29 out of the 30 datasets. In comparison with static SBPSO, Up BPSO usually needs at least a double computational time due to updating parameters and the exponential calculation. Dynamic SBPSO is slightly more expensive than static SBPSO since it needs to perform an extra step to update the parameters, but it is more efficient than Up BPSO. Among the five algorithms, SBPSO algorithms have the best trade-off between efficiency and effectiveness.

4.5 Experiments on Feature Selection

4.5.1 PSO for Feature Selection

The five algorithms are compared on 12 feature selections datasets chosen from the UCI machine learning repository [32]. The datasets are different in the numbers of features, classes, and instances, which can be seen in Table 1.1. In the previous chapter, the proposed algorithm is a filter approach which has a low computational cost, so 10-fold cross validation is applied. However, this chapter and the following chapters focus on wrapper-based methods which are usually more expensive than a filter approach. Therefore, from this chapter, we use a strategy to split the dataset into training and test sets, so that they contain 70% and 30% instances, respectively, and the class distribution is roughly preserved. More importantly, since the datasets in this thesis do not have limited numbers of instances, it is not that necessary to apply 10-fold cross validation which is usually applied when the number of instances is small. For each dataset, each algorithm is run 50 independent times, and each run contains 400 iterations. The swarm size is set to the number of features and it is bounded by 100.

The representation of the PSO algorithms for feature selection is the same as the representation for knapsack, in which each position corresponds to one original feature and indicates whether the feature is selected or not.

In feature selection, there are two main objectives, which are to maximize the classification performance and to minimize the number of selected features. The two objectives can be combined to form a fitness function as below:

$$fitness_{FS} = \gamma \times ErrorRate + (1 - \gamma) \times \frac{\#selected}{\#all} \tag{4.11}$$

where *ErrorRate* is the classification error of the selected features, *#selected* and *#all* represent the number of selected features and the total number of original features, respectively. γ is used to control the contribution of

the two objectives. Since the classification performance is preferred over the number of selected features, γ is usually set to 0.9. The task is to find a feature subset to minimize the fitness function given in Eq. (4.11). In this chapter and the following chapters that are wrapper-based feature selection approaches, KNN is used to calculate the classification error. The reason for selecting KNN is that it is one of the simplest classification algorithms. There is a study conducted by Xue et al. [214] showing that if a wrapper approach uses a simple classification algorithm, the selected features are more generalized than using more complicated classification algorithm. The other reason is KNN itself cannot perform feature selection or feature weighting. Some different well-known classification algorithms such as DT, SVM have a feature selection process embedded, which makes it more difficult to judge which feature selection approach is better. Another reason is that KNN relies on distance measures, so it is more harmful by the "curse of dimensionality". In this chapter, $K$is set as 5 to ensure that KNN can avoid noisy data while still maintaining its efficiency [214].

In this section, the five algorithms are compared on 12 feature selection problems, which have different numbers of features, classes, and instances.

4.5.2 Feature Subsets

The comparisons are performed on three terms: training accuracy, testing accuracy and the number of selected features, which are shown in Table 4.7. Tables 4.8 and 4.9 show the classification accuracies on training and test sets, respectively. In the tables, "All" means that all features are used in the classification process. The bold numbers/accuracies indicate that the corresponding algorithms achieve the highest performance. The smallest number of features is marked in bold. The static and dynamic SBPSO algorithms are compared with other algorithms using Wilcoxon test with a confidence interval of 0.95.

Table 4.7: W/D/L on feature selection.

Method	GAs	DE	Up	Stat
Stat	9/1/2	3/6/3	0/8/4	
Dyn	8/2/2	4/7/3	4/6/2	5/7/0

(a) Training Accuracy

Method	GAs	DE	Up	Stat
Stat	7/3/2	5/6/1	1/10/1	
Dyn	8/3/1	7/4/1	5/6/1	3/9/0

(b) Testing Accuracy

Method	GAs	DE	Up	Stat
Stat	8/2/2	8/3/1	2/10/0	
Dyn	10/2/0	7/1/4	6/5/1	4/7/1

(c) Number of Features

Table 4.8: Training results on feature selection.

Dataset	Number of Features						Training Accuracies					
	All	GAs	DE	Up	Stat	Dyn	All	GAs	DE	Up	Stat	Dyn
Wine	13.0	3.6	3.7	4.0	3.4	**3.3**	88.17(0.00)	96.22(0.44)	96.59(0.27)	**96.77(0.00)**	96.42(0.27)	96.38(0.30)
Australian	14.0	**2.7**	2.9	3.0	**2.7**	2.8	75.78(0.00)	82.40(7.42)	86.06(4.01)	**86.55(2.84)**	80.47(9.46)	83.71(7.44)
Vehicle	18.0	**4.8**	5.0	5.0	4.9	**4.8**	88.51(0.00)	89.41(0.73)	89.71(0.50)	89.51(0.43)	**89.74(0.78)**	89.69(0.82)
German	24.0	6.8	**5.2**	5.3	5.5	5.4	**80.14(0.00)**	80.11(2.13)	79.26(0.65)	78.87(1.16)	78.91(1.96)	78.59(2.01)
WBCD	30.0	2.4	**2.0**	**2.0**	**2.0**	**2.0**	94.97(0.00)	**95.55(0.75)**	95.23(0.00)	95.23(0.00)	95.17(0.24)	95.19(0.20)
Ionosphere	34.0	6.0	**3.0**	3.2	3.6	3.5	85.77(0.00)	92.75(1.20)	94.29(0.17)	94.22(0.36)	94.14(0.51)	**94.30(0.38)**
Sonar	60.0	14.3	14.7	13.6	13.1	**12.9**	83.45(0.00)	90.23(1.98)	92.55(1.34)	**92.56(1.24)**	91.45(1.81)	91.89(1.62)
Hillvalley	100.0	29.2	22.8	24.1	23.9	**22.7**	71.46(0.00)	73.89(1.19)	**74.91(0.97)**	74.84(0.92)	74.62(1.07)	74.76(1.09)
Musk1	166.0	57.1	64.0	60.7	58.6	**50.9**	92.19(0.00)	94.27(1.03)	95.29(0.64)	95.57(0.64)	95.25(0.78)	**95.91(0.60)**
Arrhythmia	278.0	65.2	65.0	38.8	46.1	**26.3**	94.35(0.00)	95.23(0.19)	95.60(0.2)	96.03(0.24)	96.06(0.21)	**96.38(0.17)**
Madelon	500.0	198.9	201.7	188.5	184.5	**171.8**	83.24(0.00)	87.70(0.78)	89.17(0.57)	90.39(0.66)	90.23(0.65)	**91.23(0.51)**
Multiple Features	649.0	174.0	188.4	107.0	130.9	**76.4**	99.33(0.00)	99.50(0.04)	99.53(0.04)	99.57(0.05)	99.57(0.06)	**99.61(0.04)**

Table 4.9: Testing results on feature selection.

Dataset	Testing Accuracies					
	All	GAs	DE	Up	Stat	Dyn
Wine	76.54(0.00)	96.94(3.04)	97.21(2.03)	96.30(0.00)	**98.30(2.37)**	98.15(2.89)
Australian	70.05(0.00)	81.42(7.83)	84.64(3.98)	**85.10(2.84)**	79.02(9.46)	82.26(7.44)
Vehicle	**84.06(0.00)**	84.03(0.98)	83.70(0.27)	83.81(0.26)	83.72(0.57)	83.78(0.57)
German	68.00(0.00)	68.23(2.01)	68.75(0.99)	68.92(1.30)	69.31(2.06)	**69.32(1.56)**
WBCD	92.98(0.00)	93.83(0.97)	**94.74(0.00)**	**94.74(0.00)**	94.60(0.56)	94.65(0.46)
Ionosphere	83.81(0.00)	**88.76(2.16)**	85.73(0.13)	86.51(1.52)	87.25(2.19)	86.59(1.65)
Sonar	76.19(0.00)	77.91(3.11)	**80.83(3.30)**	79.71(2.95)	79.37(3.74)	79.71(3.35)
Hillvalley	56.59(0.00)	58.24(1.38)	58.32(1.53)	58.47(1.16)	58.55(1.70)	**59.02(1.48)**
Musk1	83.92(0.00)	85.44(2.70)	85.96(2.03)	**86.32(1.94)**	85.47(2.31)	86.00(2.23)
Arrhythmia	93.78(0.00)	94.38(0.33)	94.74(0.28)	95.09(0.36)	95.09(0.34)	**95.26(0.39)**
Madelon	70.90(0.00)	78.35(1.24)	80.01(1.31)	81.11(1.23)	80.79(1.33)	**81.95(1.31)**
Multiple Features	98.57(0.00)	98.96(0.13)	98.99(0.09)	99.03(0.11)	99.05(0.11)	**99.07(0.08)**

As can be seen from Tables 4.8 and 4.9, on most of the datasets, the two SBPSO algorithms can successfully evolve small feature subsets with similar or better classification accuracy than using all features. For example on Arrhythmia, static and dynamic SBPSO select less than 16% of the original features while achieving almost 2% higher accuracy than using all features.

In terms of training accuracy, on most of the datasets, static and dynamic SBPSO achieve higher training accuracy than GAs. As can be seen in Table 4.7(a), the SBPSO algorithms are significantly better than GAs on at least nine out of the 12 datasets. In comparison with DE, the SBPSO algorithms achieve similar or better performance on 11 datasets. Although static SBPSO outperforms Up BPSO on only one dataset, dynamic SBPSO is significantly better than Up BPSO on five datasets. On the four largest datasets, dynamic SBPSO can evolve the best feature subsets in comparison with all the other algorithms.

As can be seen in Table 4.7(b), in terms of testing accuracy, static SBPSO is significantly better than GAs, DE and Up on seven, five and one datasets, respectively. Meanwhile, except for being worse than GAs on 2 datasets, static SBPSO is worse than the other two algorithms on only 1 dataset. On

all datasets, dynamic SBPSO is always similar or better than static SBPSO. In comparison with other non-SBPSO algorithms, dynamic SBPSO is significantly better on at least five datasets while being worse on only one dataset. On the three largest datasets, dynamic SBPSO achieves the best training and testing accuracies. This pattern shows that dynamic SBPSO is able to select feature subsets, which are consistently good for training and test sets.

Beside classification accuracies, the number of selected features is also an important objective. As can be seen in Table 4.8, on eight out of the 12 datasets, dynamic SBPSO selects the smallest number of features. For example, on the largest dataset, Multiple Features, dynamic SBPSO selects on average only 76.4 features, which is 30 features less than the second smallest number of features selected by Up BPSO. The significance test results in Table 4.7(c) show that in terms of the number of selected features, dynamic SBPSO selects more features than other algorithms on at most four out of the 12 datasets.

In general, on 10 out of the 12 datasets, dynamic SBPSO can achieve the highest classification accuracy or the smallest number of selected features, which are the two main objectives of feature selection. However, since both objectives contribute to the fitness function, analyzing them separately cannot show the performance of the algorithms. Therefore, the evolutionary process will be analyzed in the next section.

4.5.3 Evolutionary Processes

In order to analyze the evolutionary process, the best fitness value of each iteration is recorded. Since each algorithm is run 50 independent times, the average of 50 best fitness values in each iteration from the 50 runs is used to show the evolutionary process. Therefore, for each algorithm, there will be 400 average fitness values corresponding to the 400 iterations. The 400 average values are used to draw an algorithm's evolutionary process,

which is shown in Fig. 4.4. Remember that in feature selection, the target is to minimize the fitness function so the lower the fitness value, the better the algorithm. In the figure, only 8 evolutionary processes are shown, since the processes are similar on the other datasets.

As shown in the figure, on the first eight datasets (Wine-Sonar) with small numbers of features, although in the first 100 iterations, the SBPSO algorithms evolve better fitness values, Up BPSO and binary DE still can achieve lower fitness values than the SBPSO algorithms at the end of the evolutionary process. However, except for the German dataset, on the other seven datasets, the differences between the algorithms are not significant.

When the number of features is increased, the final solution of dynamic SBPSO achieves the lowest fitness values. Binary DE does not perform well on the datasets with a large number of features, where its fitness value is worse than that of the SBPSO algorithms during the whole evolutionary process. Similar to the knapsack problem, Up BPSO significantly improves its fitness value only in the last 100 iterations, when probably it focuses on exploitation. Among the two SBPSO algorithms, static SBPSO usually generates better solutions in the first half of a run. Since at the beginning dynamic SBPSO focuses on exploration more than static SBPSO, it tends to discover more promising regions rather than focusing on some specific regions. Therefore, in the second half when dynamic SBPSO focuses more on exploitation, it can improve its fitness value over static SBPSO. It can be seen that at the beginning dynamic SBPSO explores more than static SBPSO and less than Up BPSO, while in the later iterations, dynamic SBPSO increases its exploitation over static SBPSO.

4.5.4 Computational Time

Table 4.10 shows the computational times of the five algorithms. On each dataset, the least computational time is marked in bold. As can be seen

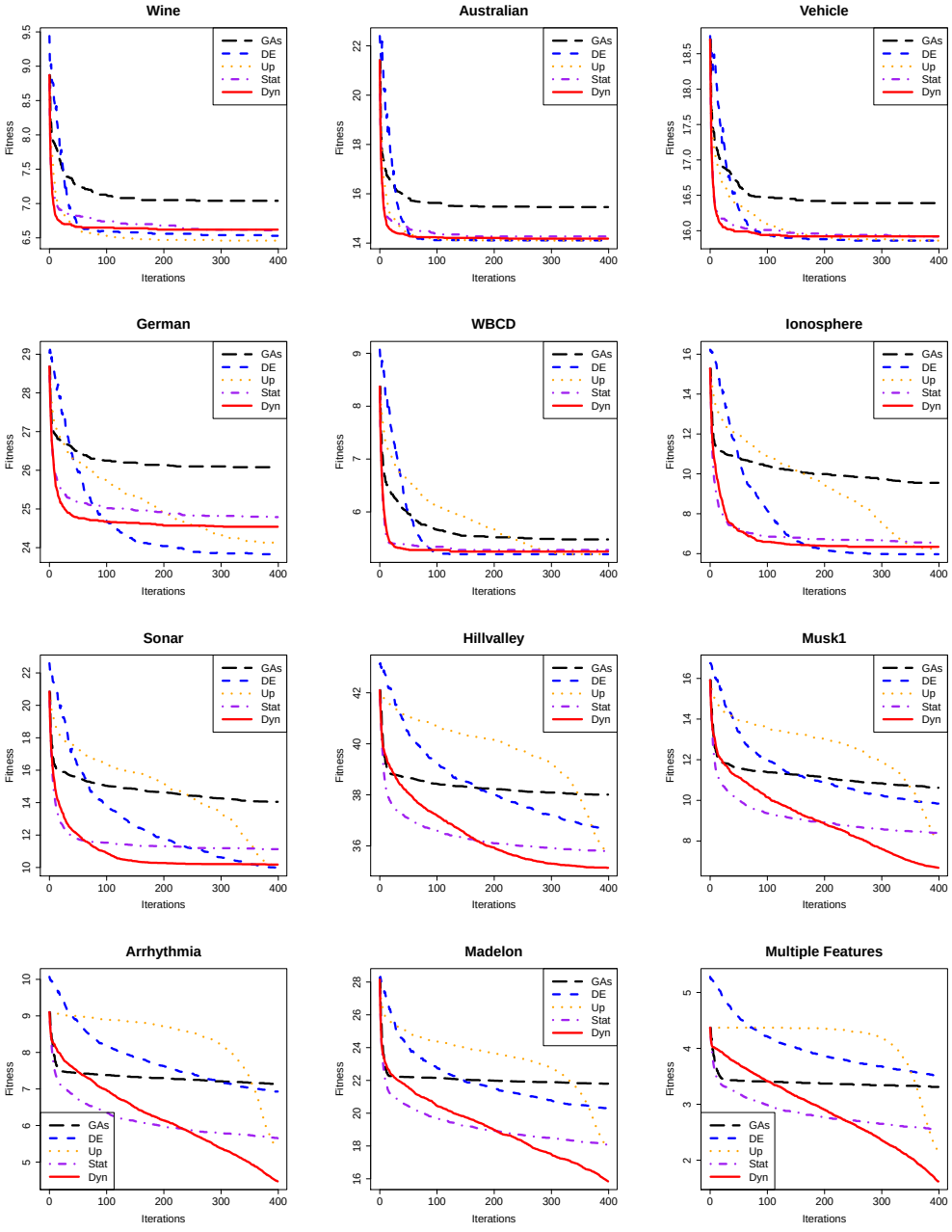


Figure 4.4: Evolutionary processes on feature selection.

Table 4.10: Computational time (in seconds) on feature selection.

Dataset	GAs	DE	Up	Stat	Dyn
Wine	25.30	25.72	13.36	**12.89**	13.21
Australian	493.92	467.81	**234.11**	243.62	238.37
Vehicle	854.23	905.47	470.63	458.68	**448.22**
German	1675.08	1676.74	864.95	854.35	**848.89**
WBCD	641.35	636.09	321.34	**307.54**	310.33
Ionosphere	296.15	276.36	**136.50**	138.04	137.32
Sonar	169.73	170.90	87.83	**83.22**	84.28
Hillvalley	11725.56	11826.99	6489.73	**5705.48**	5961.86
Musk1	2137.65	2097.33	1134.12	**1024.06**	1038.67
Arrhythmia	1815.37	2039.85	1118.84	**933.95**	967.77
Madelon	96815.06	107603.77	56782.26	**49948.55**	52157.75
Multiple Features	52111.53	79564.87	33012.97	27977.13	**26939.52**

from the table, on 10 out of the 12 datasets, the SBPSO algorithms are the most efficient algorithms. Specifically, static SBPSO has the lowest computational time on seven datasets while dynamic SBPSO is the fastest one on three datasets. Since in feature selection, the most time is consumed by the fitness evaluation, which is directly related to the number of selected features, DE is slower than the SBPSO algorithms due to its large number of selected features. In comparison with Up BPSO, although static SBPSO and Up BPSO select similar number of features, Up BSPSO is still slower in most of the datasets because its updating equation contains exponential calculation.

The results show that the dynamic strategy can balance between the exploitation and exploration better than the other three algorithms to achieve good solutions on large and complicated search spaces.

4.5.5 Further Discussions

According to the experimental results, it can be seen that the SBPSO algorithms can better explore the large search space by performing exploration and exploitation alternatively. In order to further analyze the behavior of the swarm, the idea of an evolutionary factor [216] is applied. Firstly,

the mean distance from each particle i to the other particles is calculated, called d_i. The evolutionary factor f is calculated as follows:

$$f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \in [0, 1] \tag{4.12}$$

where d_g is d_i of the globally best particle, d_{min} and d_{max} are the minimum and maximum values of all d_i. Based on the evolutionary factor, a set of fuzzy rules are applied to identify the corresponding evolutionary state. In general, the population distribution and the fitness function are used to estimate the real-time evolutionary state of a PSO-based algorithm. There are four evolutionary states, which are *exploration*, *exploitation*, *convergence* and *jumping out*. The evolutionary states of Up BPSO, static and dynamic SBPSO, in one run on the Sonar dataset, are shown in Fig. 4.5. In the figure, the horizontal axis represents the iterations and the vertical axis is the evolutionary state. The values of *exploration*, *exploitation*, *convergence* and *jumping out* are 1, 2, 3 and 4, respectively.

As can be seen from Fig. 4.5, in the first 200 iterations, Up BPSO performs much more exploration than the two SBPSO algorithms. In the last 200 iterations, most of the time, Up BPSO is in exploitation and convergence state. On the other hand, the two SBPSO algorithms explore even more frequently than at the beginning. It can be seen from the figure that the SBPSO algorithms alternatively explore and exploit while Up BPSO focuses on exploration at the beginning and on exploitation at the end. In comparison between dynamic SBPO and static SBPSO, in the later half of the evolutionary process, dynamic SBPSO jumps out more frequently than static SBPSO, which can avoid being stuck at local optima.

4.5.6 Comparison with non-EC based Feature Selection

Recently, sparse learning based feature selection methods gain a lot of attention. The idea is to find an optimal weight vector, which minimizes the fitting error along with some regularization terms. Because of the sparse

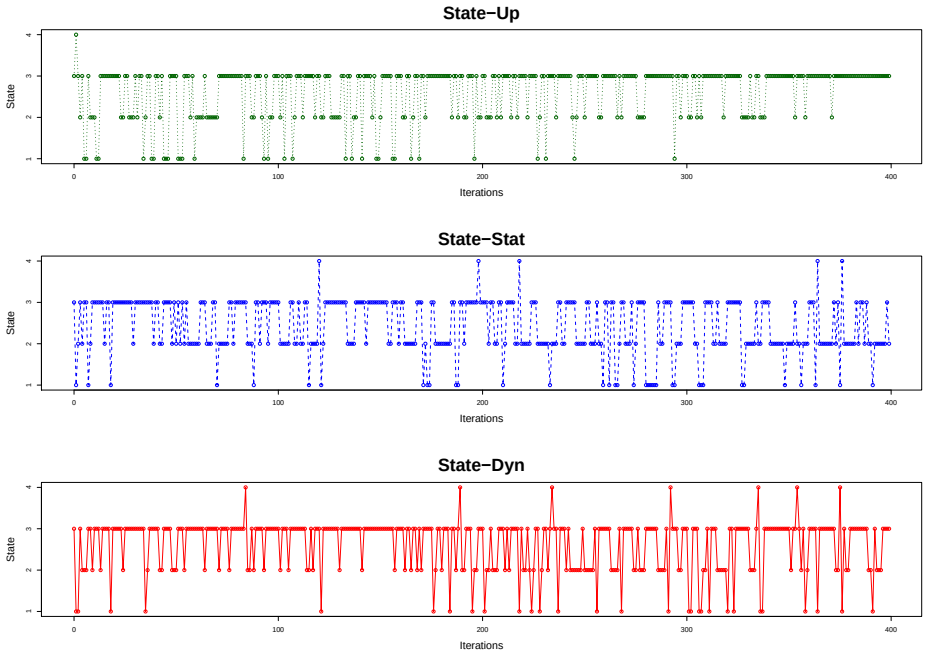


Figure 4.5: Evolutionary state of BPSO algorithms.

regularizer, some learned weights will be very small and their corresponding features are discarded. However, these methods usually require a predefined number of selected features. Among sparse learning based feature selection methods, the robust feature selection method (RFS) [217] is one of the most popular ones. Instead of using the ℓ_2-norm based loss function, $\ell_{2,1}$-norm is applied to avoid outliers in data points. In addition, $\ell_{2,1}$ is also cheaper to calculate. The formula of $\ell_{2,1}$ of a vector V is given in Eq. (4.13).

$$||V||_{2,1} = \sqrt{\sum_{i=1}^{|V|} V_i^2} \tag{4.13}$$

Experimental results on six datasets show the better performance of RFS over several popular traditional feature selection approaches. Therefore, in this chapter, RFS is selected as a benchmark algorithm to compare

Dataset	Wine	Australian	Vehicle	German	WBCD	Ionosphere
RFS	91.36	83.57	80.31	43.33	91.81	81.90
Dyn	**98.15**	**82.26**	**83.78**	**69.32**	**94.65**	**86.59**

Dataset	Sonar	Hillvalley	Musk1	Arrhythmia	Madelon	MFs
RFS	73.02	54.95	83.22	92.76	**84.36**	91.43
Dyn	**79.71**	**59.02**	**86.00**	**95.26**	81.95	**99.07**

Table 4.11: Classification accuracies of Dyn and RFS.

with our dynamic SBPSO-based feature selection (Dyn).

To ensure a relatively fair comparison, the average numbers of features selected by Dyn are used as pre-defined numbers of features for RFS. Results of the two algorithms are presented in Table 4.11, where the bold letter means that the corresponding method is significantly better than the other one. It can be seen that on 11 out of the 12 datasets, Dyn achieves significantly higher accuracies than RFS. The largest difference is on the German dataset, where Dyn is 26% better than RFS. Notice that on Australian, although the Dyn's average classification accuracy is lower than the one of RFS, Dyn is still significantly better than RFS. The reason is that on 42 out of the 50 independent runs, Dyn achieves 85.52% which is 2% better than RFS. There might be an outlier with too low classification accuracy, which reduces the average accuracy.

The superior of Dyn is the result of several reasons. Firstly, RFS assumes a linear relationship between feature vectors and the class label, which is not guaranteed in most real-world datasets. Furthermore, RFS is based on its built-in weight matrix to select a number of top features, which may ignore the interactions between features and is likely to select redundant features. By applying dynamic SBPSO, it is not needed to predefine the number of selected features as in RFS. In SBPSO-based feature selection algorithms, the selected features are evaluated as a group, so the feature interactions are considered.

4.6 Chapter Summary

In this chapter, a new BPSO approach is proposed to better explore the search space and evolve better solutions for binary problems. In standard BPSO, the $velocity$ and $momentum$ from continuous PSO are applied directly despite the fact that the binary search space is not as smooth as the continuous search space. In the proposed algorithm, the two concepts are revised so that the particles move around the search space in a more meaningful way. In addition, a dynamic strategy is also developed to control the contributions of $momentum$, $pbest$ and $gbest$ to the movement of particles, which results in the balance between $exploration$ and $exploitation$ during the evolutionary process. Depend on whether the dynamic mechanism is applied or not, the proposed BPSO algorithm has two versions: dynamic and static SBPSO, respectively. To examine the performance of the proposed algorithm, a large number of experiments have been conducted on two well-known types of binary problems: knapsack and feature selection. The performance of SBPSO is compared with GAs, DE, and Up BPSO which is a state-of-the-art BPSO algorithm. The experimental results show that the dynamic SBPSO algorithm can achieve similar or better performance than the other four algorithms on most of the datasets from the two binary problems. Especially on feature selection, SBPSO usually selects the smallest number of features while maintaining or improving the classification performance over other algorithms. While DE works quite well with small knapsack datasets, it cannot achieve as good results as both static and dynamic SBPSO on the datasets with a large number of items or features. In comparison with static SBPSO, dynamic SBPSO can better balance the trade-off between exploration and exploitation to finally evolve better solutions than static SBPSO. The evolutionary processes show that dynamic SBPSO performs exploration and exploitation iteratively, which keeps improving the fitness value. On the other hand, Up BPSO significantly improves the fitness values in the last 16% iterations.

The chapter brings the following major contributions. Firstly, by considering the differences between binary and continuous search spaces, the $velocity$ of BPSO is redefined as the probability of flipping the position entries, which reflects the movements in binary search space more accurately. The $momentum$ of BPSO is also modified since there is no direction in a binary search space. Specifically, it is defined as the tendency to $stick$ with the current position, known as the $stickiness$ property. Secondly, the two revised concepts help to define $exploration$ and $exploitation$ easier in BPSO, from which a dynamic mechanism is proposed to balance between $exploration$ and $exploitation$. The experimental results show that in comparison with the static SBPSO, the dynamic one explores more at the beginning while jumping out of the local optima more frequently in the later iterations, which avoids the premature convergence problem. Last but not least, the experimental results also show that iteratively switching between $exploration$ and $exploitation$ during the evolutionary process achieves good results on complicated search spaces which have a large number of dimensions and/or complex interactions between dimensions. Given the ability to alternate between $exploration$ and $exploitation$, it is expected that SBPSO can be widely applied to other complicated binary problems.

This chapter develops a novel BPSO algorithm with good results. There is still a lot of future work can be done in this direction. For example, it can be seen that the dynamic SBPSO algorithm performs extremely well on the large and complex search spaces. However, the performance is not significantly improved in the small problems. This could be tackled by developing an adaptive mechanism for parameter settings which considers the feedback from search spaces to automatically adjust the dynamic mechanism. Although the proposed BPSO algorithm has better efficiency than other algorithms in feature selection, the improvement is not quite significant since the evaluation process, which is the most time-consuming step, is the same for all algorithms. In the next chapter, a surrogate model

is proposed to speed up the evaluation process.

Chapter 5

Surrogate Model for Feature Selection

5.1 Introduction

Wrapper-based feature selection approaches usually achieve high classification performance. However, the effectiveness comes along with an expensive computational cost. The question is how to reduce the computational cost of wrappers while still maintaining or even improving their performance. In general, the most computationally intensive part of wrappers is the evaluation step which involves a classification process. In order to speed up wrappers, it is necessary to make the evaluation step faster, which is the main motivation of this chapter.

5.1.1 Chapter Goal

The overall goal of this chapter is to propose an effective surrogate model for wrapper PSO-based feature selection, which is expected to reduce the computational cost of wrappers while maintaining or even improving the classification performance. Particularly, a surrogate training set, which contains a small number of informative instances, is built to assist PSO

quickly locate promising regions in the search space. In this chapter, the informative instances can be selected by two approaches including DROP3 [218] (an instance selection algorithm) and a hierarchical clustering algorithm [219]. Furthermore, the relationship between surrogate and full training sets is also investigated, from which a dynamic surrogate model is proposed so that it can adapt with different datasets to select a small number of features with high discriminating abilities. Specifically, we will investigate the following questions:

- whether the two methods can achieve similar or better performance than using the original training set;
- which of the two approaches are better to build the surrogate training set;
- whether a bigger surrogate training set can lead to better feature subsets. Note that when the size of the surrogate training set is increased, the surrogate training set is more similar to the original training set, and
- whether the proposed dynamic surrogate model can select suitable surrogate training sets, which helps to evolve better feature subsets in a shorter training time than using the original training set.

5.2 Proposed Methods

5.2.1 Static Surrogate Model for PSO-based Feature Selection

A large training set is one of the main reasons for a classification process being expensive, which leads to a computationally intensive cost for wrapper-based feature selection. In addition, there might be some noisy instances which may deteriorate the classification performance. In order

to reduce the computational cost and at least maintain the accuracy, an instance selection algorithm can be used to select a small number of representative instances to form an instance subset, called a surrogate training set, which is expected to contain essential information from the original training set. The fitness function calculated on the *surrogate training set* is called a *surrogate fitness function*, which can be considered as an estimation of the original fitness function. In this section, two efficient approaches are considered to build the surrogate training set, which are DROP3 [218] and a clustering algorithm [219]. Firstly, the two approaches are described in details, which is followed by an explanation about how to use the surrogate training set in PSO-based feature selection.

DROP3-based surrogate training set

Wilson et al. [220] propose several efficient instance selection algorithms. Among the proposed algorithms, it has been shown in [218] that DROP3 achieves good performance. In general, DROP3 reduces the number of instances by removing central instances and retaining border instances. The main reason is that the internal instances do not affect the decision boundaries as much as the border instances. Therefore, removing the central instances has less side effect on the classification performance than removing the border instances.

DROP3 starts with filtering out all noisy instances, which are assigned to wrong class labels by their Knearest neighbors. This step also removes instances in the middle of two or more class boundaries, which creates a smoother decision boundary. For each remaining instance, there is an “enemy" instance, which is the closest instance with a different class label. The distance from an instance to its “enemy" instance is called the “enemy" distance. Therefore, the larger the “enemy" distance an instance has, the farther the instance is to its class label boundary. All remaining instances are sorted according to their “enemy" distances so that the internal instances, which are far from their class label boundary, are removed first.

For each instance, its removing process relates to its associated instances, which have the instance as one of their Knearest neighbors. If removing the instance does not reduce the number of correctly classified associated instances, then the instance will be removed. The set of selected instances is used as the *surrogate training set*.

Clustering-based surrogate training set

The second approach to build the surrogate training set is to use clustering algorithms, which groups similar instances in the same group or cluster. A representative is formed for each cluster, which will contribute as one instance into the surrogate training set. Therefore, the size of the surrogate training set is equal to the number of clusters. In this chapter, the centroid of a cluster, which is the instance closest to the cluster's mean, is selected as the representative. The main reason is that using original instances can preserve the relationships/interactions between features while building a new instance from a cluster is more likely to construct new feature interactions, which do not exist in test sets. Note that a cluster may not be pure, which means that it may contain instances from different classes. Therefore, only instances from the majority class, which contributes the largest number of instances in the cluster, are used to select the representative for the cluster.

The question is which clustering algorithm should be used here. K-means has proposed about 50 years ago and has been widely used in clustering [221], which may be a good option. However, the main task of this chapter is to analyze how surrogate training sets with different sizes affect performances of the selected feature subset. Therefore, K-means has to be run many times with different numbers of clusters, which is time-consuming. Agglomerative clustering (AGG) [219] is a bottom-up hierarchical clustering algorithm in which each instance starts with its own cluster. When moving up the hierarchy, the two closest clusters are merged into one cluster. An example of the agglomerative clustering algorithm is

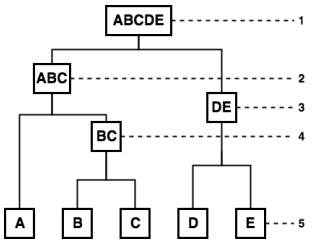


Figure 5.1: An example of the agglomerative clustering algorithm.

given in Fig. 5.1.

Note that although both DROP3 and AGG are deterministic algorithms, they have very different outputs and behaviors. DROP3 directly produces a unique surrogate training set for each dataset. On the other hand, AGG results in a set of possible clustering partitions, whose numbers of clusters ($\#c$) can be from 1 to the total number of instances in the original training set (as shown in Fig. 5.1). If $\#c$ is decided, AGG produces only one unique clustering partition containing a unique set of clusters. Since only the centroid instance is selected from each cluster, the size of surrogate set formed by AGG is equal to the number of clusters $\#c$.

Surrogate training process

Surrogate models for fitness evaluations can be divided into three categories: individual-based models, generation-based models and population-based models [222]. In individual-based models, some individuals from the population are calculated by using the original fitness function, and other ones are evaluated by the surrogate fitness function. Population-based models have more than one sub-populations, which might use different surrogate fitness functions. The communication and exchanging individuals from different sub-populations are allowed. For generation-based models, the surrogate fitness function is used in some of the generations before the original fitness function is used in the rest of the generations. In PSO, the swarm starts by exploring the search space to locate

promising areas, which are then exploited in the later iterations. It can be said that in early iterations the swarm tries to estimate the possible regions of global optima. In the sense of estimation, it would be safe to use the surrogate model at the beginning of a PSO algorithm to locate promising areas before using the original fitness function to find out the exact optima. Therefore, the idea of generation-based model is suitable here. Specifically, the particles are evaluated using the surrogate training set in the first I_s iterations, while in the rest iterations, the whole training set is used. Given I is the maximum number of iterations, the task is to figure out the value of I_s iterations so that the classification performance is still maintained or even improved over using the original fitness function. The ratio between I_s and I is called the surrogate rate α_s, i.e. $\alpha_s = \frac{I_s}{I}$.

Local search: sampling on *gbest*

One advantage of using the surrogate training set is its efficiency. However, the surrogate approach aims to estimate the fitness function, which usually achieves at most the same performance as using the whole training set. Given that we save a lot of computational time, we can spend a part of the saved computation time on a local search with an expectation to obtain a better solution despite using the surrogate training set. Of course, the local search should be efficient so that both the effectiveness and efficiency can be improved over using the original training set.

In a PSO algorithm, the main idea is to use the current $gbest$ and $pbest$ to guide the particles to follow promising trajectories. However, the $gbest$ from the previous iterations might contain some useful information, which can assist the swarm to achieve better solutions. For instance, in feature selection, features which appear in $gbest$ for many iterations tend to be good features. Moreover, since feature selection has a large and complex search space, some good features might not be selected together in the $gbest$ solutions. These features might be complementary features, which provide even more information about the class label when appearing in one fea-

ture subset. Therefore the main idea of the local search is to keep features selected on all $gbest$ and use them to improve the current $gbest$.

Suppose that S_{best} is the set of features which are selected by $gbest$ from previous iterations. Each feature from S_{best} has a score (explained later and shown in Eq. (5.1)). The local search constructs $\frac{P}{2}$ candidate solutions by using S_{best}, where P is the population size. $|gbest|$, the number of features in the current $gbest$, is the maximum number of features in each candidate feature subsets. Specifically, based on the calculated scores, a tournament selection is used to select $|gbest|$ features from S_{best}, which form a sampled feature subset. The higher a feature's score is, the more chance that feature is selected. In addition, in $|gbest|$ selected features, there might be some duplicated features, which means that the size of the sampled feature subset can be less than $|gbest|$. All sampled feature subsets are then compared with each other based on their surrogate fitness values to find the best candidate feature subset. In this way, the local search utilizes the surrogate training set to approximate a good solution in a short time. After that, the best candidate subset and the current $gbest$ compete on the current fitness function, $fitness_{cur}$, which is the surrogate fitness function in the first I_s iterations or the original fitness function in the last $(I - I_s)$ iterations. The winner becomes the new $gbest$.

The task now is to define the scores of features in S_{best}. The main idea is to give higher scores to features, which are selected more frequently and recently by *gbest*. The first component of the score is $freq$, which measures the number of iterations, in which a feature is selected in $gbest$. The higher $freq$ a feature has, the better the feature's quality. Before contributing to the score, $freq$ is normalized to $freq_n$ so that the total $freq_n$ of all features from S_{best} is 1. The second component of the score is related to the current $gbest$, called gc. If a feature in S_{best} is also selected in the current $gbest$, its gc is set to $\frac{1}{|gbest|}$. Otherwise its gc is 0. The score of the f^{th} feature in S_{best} is the sum of its $freq_n$ and gc, which can be seen in Eq. (5.1).

$$score_f = freq_{nf} + gc_f \tag{5.1}$$

where

$$freq_{nf} = \frac{freq_f}{\sum_{f=1}^{|S_{best}|} freq_f}$$

$$gc_f = \begin{cases} \frac{1}{|gbest|}, & \text{if } f \in gbest \\ 0, & \text{otherwise} \end{cases}$$

As illustrated in Eq. (5.1), given the same $freq_n$, the gc component gives more scores to features which are recently selected. In addition, since the best sampled subset might replace the current $gbest$, it is preferred to keep the sampled subset close to the current $gbest$ so that the swarm is not distracted. The gc component lets the features from the current $gbest$ to have more chance to be chosen, which allows to build new feature subsets not far from the current $gbest$. The local search is applied after the $gbest$ of the current iteration is determined and before that $gbest$ is informed to all particles.

Overall algorithm

Sticky binary PSO, which is proposed in Chapter 3, is applied to achieve feature selection, in which each position entry corresponds to one original feature. The value 1 of a position entry indicates that the corresponding feature is selected; otherwise, the corresponding feature is not selected. In this chapter, a feature subset is evaluated by the fitness function given in Eq. (5.2)

$$fitness = \gamma \times ErrorRate + (1 - \gamma) \times \frac{\#\text{selected}}{\#\text{all}} \tag{5.2}$$

where PSO needs to minimize the classification error determined by a classification algorithm and the number of selected features. The proportions of the two objectives are controlled by γ. If the classification algorithm is trained on the surrogate training set, the corresponding fitness function is a surrogate fitness function, denoted by $fitness_{sur}$. If the whole training set is used to build the classifier then the fitness function is the original

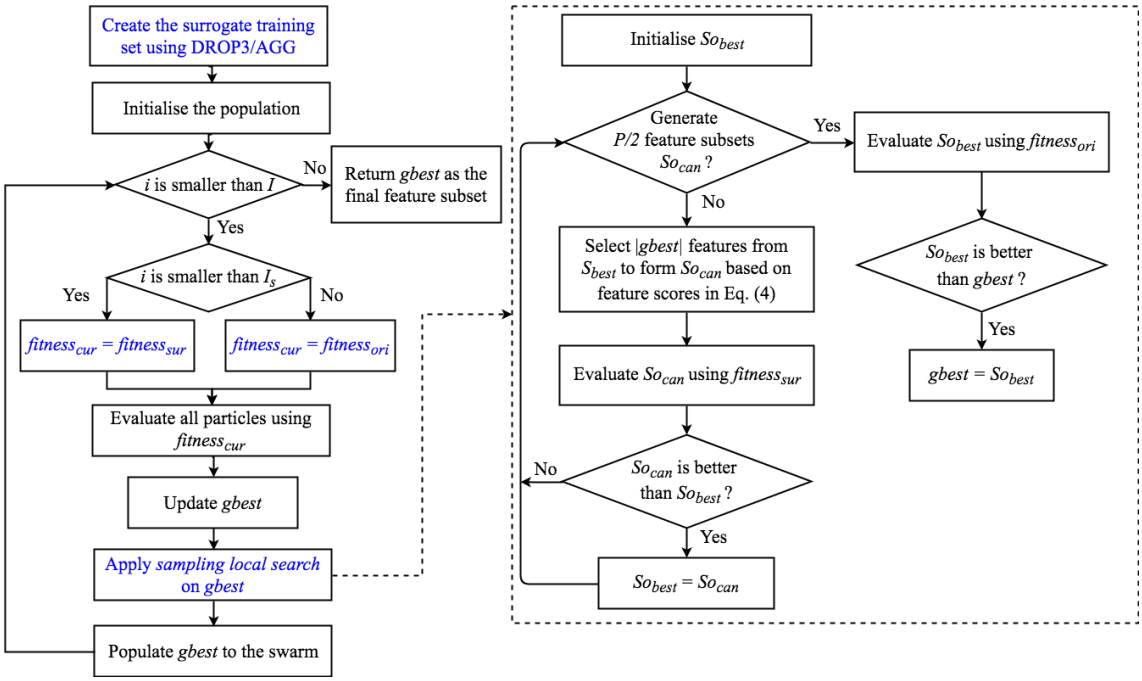


Figure 5.2: Overall algorithm using surrogate models.

one, called $fitness_{ori}$. The overall PSO-based feature selection algorithm using this static surrogate model is described in Fig. 5.2, in which the contribution of this chapter is marked in blue. So_{can} is a sampled feature subset, So_{best} is the best So_{can} generated by the sampling process, and P is the swarm size.

5.2.2 Dynamic Surrogate Model

Since the surrogate training set is used to estimate possible good regions, it is important to ensure that the surrogate fitness value should be consistent with the original fitness value i.e. the difference between the surrogate fitness value and the original fitness value should be small. However, during the evolutionary process, the consistency between the surrogate training set and the original training set may not be preserved. To address this problem, a dynamic clustering-based surrogate model is proposed. The task can be described as: "Given a pool of surrogate training sets,

$P = \{S_1, S_2, ..., S_m\}$, which surrogate training set S_i should be used to evaluate feature subsets."

In the initialization process, each particle is randomly initialized. After evaluating the particles using the original training set S_0, the position x_{best} with the best real fitness value f_0 is recorded to find out the most suitable surrogate training set. Particularly, x_{best} is evaluated on m surrogate training sets, which results in m surrogate fitness values $\{f_1, f_2, ..., f_m\}$. The surrogate training set, which has the smallest difference in comparison with the original training set, i.e. the smallest $|f_i - f_0|$, is used to evaluate feature subsets in the following iterations. Hence, even in the initialization step, the surrogate training set is dynamically determined based on its consistency with the original training set.

In the first I_s iterations, feature subsets are evaluated by the surrogate training set, which is also dynamically updated to preserve the consistency with the original training set. However, updating the surrogate set too frequently makes PSO more difficult to adapt to changes in the fitness landscape. Therefore, the surrogate one is only updated when the real fitness value of $gbest$ is not improved for a certain number of iterations ($NIStep$). The process of finding the most suitable surrogate training set is similar to the method used in the initialization process, except for that x_{best} is replaced by $gbest$. After the surrogate process, i.e. the first I_s iterations, is finished, the original training set is used to evaluate the candidate solutions.

Therefore, the main difference between the static and the dynamic surrogate model is which surrogate training set is used. While the surrogate training set in the static model is fixed from the beginning, the surrogate training set in the dynamic model is updated during the evolutionary process.

5.3 Experiment Design

The proposed methods are tested on 12 datasets chosen from the UCI machine learning repository [32]. The datasets are selected so that they have different numbers of features (#Fs), classes and instances, which can be seen in Table 1.1. Each dataset is divided into training and test sets, so that they contain 70% and 30% instances, respectively, and the class distribution is roughly preserved.

A KNN classification algorithm is used to classify instances, where K=5. The weight γ in Eq. (5.2) is set to 0.9 so that the search process focuses more on the classification performance than the number of features. For sticky PSO, the population size is equal to the number of features and limited by 100. The maximum number of iterations is 100.

5 different values of I_s ranging from 0 to 100 are examined and the results show that 75 is the most suitable setting. Table 5.1 shows results of statistical significance tests, which compare the value 75 and the other four different values of I_s on three datasets with different numbers of features. "+"/ "=" / "-" means that 75 is significantly better/similar/worse than the other values. It can be seen that I_s = 75 achieves similar or better performance than the three smaller values (0, 25, 50) while being less computationally intensive. In addition, I_s = 75 is significantly better than 100 i.e. using only the surrogate training sets. $NIStep$ is set to 5 as an indication that the algorithms might be trapped in local optima. An evolutionary process of PSO on the Madelon dataset is shown in Fig. 5.3. It can be seen that if the $gbest$'s fitness value (vertical axis) is not changed for more than 5 iterations, it is very likely that the fitness value is not changed in the following iterations.

Table 5.1: Compare different I_s values against $I_s = 75$.

Dataset	$I_s = 0$	$I_s = 25$	$I_s = 50$	$I_s = 100$
German	+	=	=	+
Sonar	=	=	=	=
Arrhythmia	-	=	=	+

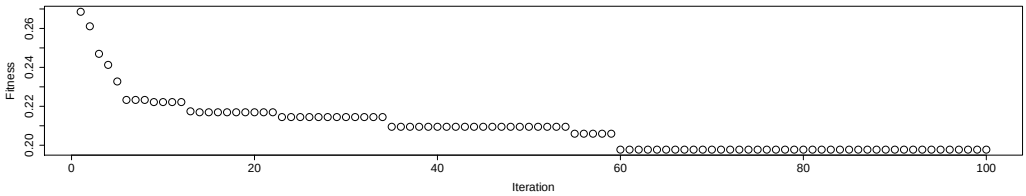


Figure 5.3: Evolutionary process of PSO on the Madelon dataset

5.4 Results and Discussions

5.4.1 DROP3 vs AGG

Firstly, the DROP3 algorithm and the AGG algorithm are compared. To ensure a relatively fair comparison, the number of clusters in the clustering algorithm is equal to the number of instances selected by DROP3. Therefore, the two algorithms result in two surrogate training sets with the same size. The surrogate training sets built by DROP3 and AGG are used in PSO-based feature selection to form two PSO-based algorithms called "P-DROP3" and "P-AGG", respectively.

Table 5.2 shows the comparison between P-DROP3 and P-AGG. In the table, "#Features" means the number of selected features, "Training" and "Testing" represent the training and testing accuracies, respectively. Note that both DROP3 and AGG use the same number of instances to build surrogate training sets. The two models are compared using Wilcoxon test, a significance signed rank test with significance level set to 0.05. "↑" or "↓" means that P-DROP3 is significantly better or worse than P-AGG, while "∘" indicates that there is no significant difference between the two algorithms. In terms of the training accuracy, P-AGG is significantly bet-

Table 5.2: DROP3 vs Agglomerative Clustering algorithms.

Dataset	#Features		Training		Testing		Time	
	P-DROP3	P-AGG	P-DROP3	P-AGG	P-DROP3	P-AGG	P-DROP3	P-AGG
Wine	3.500(↑)	3.900	96.26(∘)	96.20	95.97(∘)	94.69	0.02	0.02
Australian	2.500(↑)	2.800	77.34(∘)	81.52	76.14(∘)	80.49	0.26	0.29
Vehicle	5.000(∘)	4.8	89.56(∘)	89.73	84.10(∘)	83.89	0.58	0.64
German	5.300(↓)	3.600	78.36(↑)	75.60	69.16(∘)	69.37	0.95	1.00
WBCD	2.000(∘)	2.000	94.64(↓)	95.18	93.18(↓)	94.54	0.34	0.38
Ionosphere	3.300(∘)	3.500	93.97(∘)	93.76	86.31(↓)	87.68	0.16	0.18
Sonar	10.20(∘)	11.10	89.77(↓)	91.24	78.84(∘)	77.78	0.14	0.15
Hillvalley	22.30(↓)	15.50	74.37(∘)	74.65	58.55(∘)	59.07	7.06	7.35
Musk1	60.90(↓)	46.80	94.01(↑)	93.32	84.20(∘)	84.96	1.52	1.43
Arrhythmia	26.20(↑)	33.40	95.88(∘)	95.92	94.94(∘)	94.90	0.84	1.04
Madelon	195.3(↓)	152.6	88.99(↓)	89.91	79.64(↓)	81.85	62.19	55.65
Multiple Features	94.00(∘)	101.4	99.52(↓)	99.55	99.00(∘)	99.01	27.28	32.11

ter than P-DROP3 on four datasets while being worse only on German and Musk1. On the test set, the feature subsets selected by P-AGG are never worse than the subsets selected by P-DROP3. On three out of the 12 datasets, P-AGG's accuracies are significantly better than DROP3's. In addition, the feature subsets selected by P-AGG are similar or smaller than the ones selected by P-DROP3 on 10 datasets. The experimental results show that given the same number of selected instances, P-AGG can maintain more informative instances to form more consistent surrogate training sets, which results in better classification accuracies.

The reason that P-AGG outperforms P-DROP3 can be explained as follows. The main idea of DROP3 is to preserve all instances on class boundaries and remove all inner instances, which requires the training set is nicely distributed. Let consider two examples given in Figs. 5.4 and 5.5, where there are two class labels (marked by red and green) and a KNN classification algorithm is used with $K = 3$. In Fig. 5.4, the two green ones inside the dotted circle have the largest distances to instances from other classes, which means that they are considered to be removed first. It is obvious that removing the two instances does not affect any other green instances since they are too far from the two instances. Therefore, DROP3

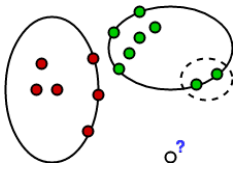


Figure 5.4: DROP3 may remove informative instances.

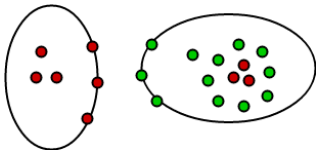


Figure 5.5: DROP3 cannot remove noisy instances.

will remove them despite they are on the class boundary. The consequence can be seen in classifying an unseen instance (marked by a question mark). If the two green instances are not removed, it will be classified as a green instance, but removing them changes the class label of the unseen instance. Thus DROP3 removes informative instances. Meanwhile, AGG is likely to group the two green instances in one cluster, which ensures that the information from the two instances is preserved.

Fig. 5.5 gives an example where DROP3 is not able to remove noisy instances. It can be seen that in Fig. 5.5, there are three noisy red instances located inside the region of the green class. The distances from the three red instances to the green class are definitely smaller than any other red instances, so according to DROP3 they are likely to be on the class boundary. In addition, removing one of the three red instances will wrongly classify the other two, so none of them is discarded by DROP3. On the other hand, AGG groups the three noisy red instances with their surrounding green instances in the same cluster and the noisy instances will be removed since the red class is the minority one in this cluster. It is possible that DROP3 removes informative instances or keeps noisy instances, which can be avoided by using AGG.

5.4.2 Results of Clustering-based Surrogate Models

Given the good results achieved by the clustering-based surrogate model, it is used to analyze the effect of different sizes of the surrogate set. Five surrogate training sets, whose sizes range from 10% to 50% of the original training set, are generated. The lower bound 10% is to ensure that the surrogate training sets contain enough training instances. The upper bound is set to 50% so that the surrogate model still can significantly reduce the computational cost over using the original training set. The five surrogate training sets form a training set pool, from which the dynamic surrogate model picks the most suitable training set during the surrogate process. The percentages i.e "10%", "20%", "30%", "40%" and "50%" are used to name the PSO-based feature selection algorithms which statically use the corresponding surrogate training sets. "Dyn" stands for the PSO-based feature selection algorithm which dynamically determines the suitable surrogate training set (dynamic surrogate model). "Ori" stands for the PSO-based feature selection algorithm which uses the original training set in the whole evolutionary process. Since Chapter 4 already showed that the SBPSO-based feature selection algorithm using the original training set ("Ori") was already better than using all features, this chapter focuses on comparing the proposed surrogate models with "Ori" to illustrate the effect of using surrogate models.

Effect of the static surrogate model

The results of clustering-based static surrogate models with different sizes of the surrogate set are shown in Table 5.3. The best classification accuracies on both training and test sets are marked in bold. In comparison with other clustering-based models, P-AGG does not achieve the best classification accuracy on any dataset. In terms of the number of selected features, P-AGG usually selects a smaller number of features than the other methods. The reason for this pattern is that P-AGG uses the smallest number of

Table 5.3: Results of clustering-based surrogate models.

Dataset	Training accuracy							Time (minutes)						
	Ori	**P-AGG**	**10%**	**20%**	**30%**	**40%**	**50%**	**Dyn**	**Ori**	**P-AGG**	**10%**	**20%**	**30%**	**40%**	**50%**	**Dyn**
Wine	96.29(↑)	96.20(○)	96.04(○)	96.26(↑)	**96.31**(○)	96.04(○)	96.26(○)	96.13	0.06	0.02	0.02	0.02	0.03	0.03	0.04	0.04
Australian	84.22(○)	81.52(○)	**85.42**(○)	82.13(○)	80.78(○)	81.82(○)	79.22(○)	80.15	0.92	0.29	0.29	0.29	0.35	0.43	0.47	0.46
Vehicle	89.79(○)	89.73(○)	89.59(○)	89.60(○)	**90.09**(○)	89.48(○)	89.88(○)	89.88	1.88	0.64	0.55	0.61	0.72	0.87	1.07	0.85
German	79.31(○)	75.60(↓)	76.77(○)	80.41(↑)	**82.55**(↑)	81.21(↑)	80.97(○)	78.56	3.62	1.00	0.95	1.16	1.38	1.69	2.05	1.35
WBCD	95.13(○)	95.18(○)	**95.92**(↑)	95.19(○)	94.74(↓)	95.06(○)	95.31(↑)	95.14	1.34	0.38	0.37	0.42	0.49	0.59	0.73	0.67
Ionosphere	**94.04**(↑)	93.76(○)	93.44(○)	93.57(○)	93.75(○)	93.89(○)	93.89(○)	93.66	0.59	0.18	0.17	0.19	0.23	0.28	0.34	0.30
Sonar	**91.97**(○)	91.24(○)	90.28(↓)	91.13(○)	91.08(○)	91.38(○)	91.19(○)	91.08	0.36	0.15	0.11	0.12	0.14	0.17	0.21	0.17
Hillvalley	74.80(○)	74.65(○)	74.60(○)	74.38(○)	**74.92**(○)	74.48(○)	74.82(○)	74.85	23.86	7.35	6.43	7.10	8.35	10.08	12.74	10.78
Musk1	**95.67**(↑)	93.32(↓)	93.48(○)	94.23(○)	93.90(○)	94.24(○)	93.85(○)	94.02	4.98	1.43	1.03	1.21	1.46	1.81	2.16	1.51
Arrhythmia	96.01(↓)	95.92(↓)	95.74(↓)	95.73(↓)	96.06(↓)	96.13(○)	**96.18**(○)	96.17	4.22	1.04	1.07	1.28	1.45	1.74	2.15	2.14
Madelon	89.92(↓)	89.91(↓)	89.64(↓)	89.82(↓)	90.09(○)	90.47(○)	**90.61**(○)	90.41	219.83	55.65	52.65	60.48	66.62	81.18	99.94	91.34
MultipleFs	99.55(○)	99.55(○)	99.54(○)	99.55(○)	**99.57**(○)	99.55(○)	99.55(○)	99.56	119.71	32.11	30.23	50.40	41.05	46.99	54.76	58.74

Dataset	Testing accuracy							#Features						
	Ori	**P-AGG**	**10%**	**20%**	**30%**	**40%**	**50%**	**Dyn**	**Ori**	**P-AGG**	**10%**	**20%**	**30%**	**40%**	**50%**	**Dyn**
Wine	**97.53**(↑)	94.69(○)	95.23(○)	94.40(○)	97.20(↑)	91.52(○)	97.49(↑)	94.16	3.2	3.9	3.2	3.2	3.4	3.1	3.2	3.1
Australian	**82.82**(○)	80.49(○)	82.50(○)	81.02(○)	79.36(○)	80.45(○)	78.02(○)	78.88	2.9	2.8	3.0	2.8	2.8	2.8	2.6	2.6
Vehicle	83.75(○)	83.89(○)	**84.32**(↑)	83.88(○)	83.43(○)	83.47(○)	83.54(○)	83.60	4.8	4.8	5.3	4.9	4.9	4.5	5.3	5.0
German	69.08(○)	69.37(○)	68.59(○)	68.99(○)	**69.50**(↑)	69.22(○)	69.45(↑)	68.56	5.7	3.6	3.7	5.7	6.1	6.0	5.8	4.9
WBCD	94.51(○)	94.54(○)	93.39(↓)	93.88(○)	92.71(↓)	93.53(↓)	**94.62**(↑)	94.13	2.0	2.0	2.6	2.2	2.4	2.4	2.1	2.2
Ionosphere	87.24(○)	87.68(○)	88.09(○)	88.00(○)	**88.25**(○)	87.52(○)	87.65(○)	87.52	4.2	3.5	3.6	3.4	3.4	3.6	3.7	3.4
Sonar	79.68 (○)	77.78(○)	77.88(○)	78.63(○)	**79.74**(○)	79.15(○)	79.52(○)	79.42	12.8	11.1	11.6	10.7	12.3	12.8	11.9	13.1
Hillvalley	58.84(○)	59.07(○)	58.98(○)	58.44(○)	58.86(○)	**59.14**(○)	58.37(○)	58.88	24.3	15.5	20.5	19.2	16.6	16.1	18.4	19.3
Musk1	85.85(○)	84.96(○)	84.13(○)	84.50(○)	84.03(○)	**86.08**(○)	85.76(○)	84.92	58.4	46.8	38.6	47.8	53.0	54.8	48.6	44.4
Arrhythmia	95.02(○)	94.90(○)	94.81(↓)	94.74(↓)	94.94(↓)	95.09(○)	**95.16**(○)	**95.16**	41.4	33.4	44.9	43.3	37.7	33.5	28.2	28.3
Madelon	80.28(↓)	81.85(↓)	80.80(↓)	81.91(○)	82.65(○)	**83.34**(↑)	82.89(○)	82.64	189.1	152.6	169.3	154.2	144.4	124.8	132.9	138.4
MultipleFs	99.03(○)	99.01(○)	99.04(○)	99.04(○)	**99.07**(○)	99.04(○)	99.05(○)	99.06	116.8	101.4	112.3	109.5	95.2	92.0	88.6	88.5

instances, so it does not need to select as many features as the other methods. This is an example of underfitting, where AGG does not have enough instances to select a sufficient number of informative features which are necessary for classifying unseen instances.

As can be seen in Table 5.3, depending on characteristics of the datasets, the best accuracies are achieved by different sizes of surrogate training sets. Mostly the surrogate training sets ranging from 30% to 50% produce the best accuracies since these training sets are more similar to the original ones. However, on 3 datasets, Australian, Vehicle and WBCD, 10% achieves the best performance, which may be an indication that the 3 datasets have noisy instances and small size surrogate training sets help to eliminate these instances. An important pattern shown in Table 5.3 is the consistency between training and testing performance. Specifically, on 6 out of the 12 datasets, both best training and testing accuracies are achieved by the same method. On the other datasets, although the exact consistency does not happen, the method with the best testing accuracy usually has the second best training performance. This pattern shows that to some extent, using surrogate models can help to avoid overfitting.

In comparison with using the whole training set shown in column "Ori" of Table 5.3, the static surrogate model usually achieves better classification accuracy. Particularly, on both training and test sets, there is at least one static surrogate model achieves better classification accuracy than Ori on most datasets. In terms of the number of selected features, Ori usually selects more features than most static surrogate models, especially on the datasets with a large number of features. In addition, the three datasets on which Ori achieves the best training accuracy are totally different from the two datasets on which Ori achieves the best testing accuracy. This pattern indicates that some features selected by using the whole training set might not be useful on the test set, which is partially avoided by using surrogate training sets.

In order to analyze the condition for a surrogate model to locate good

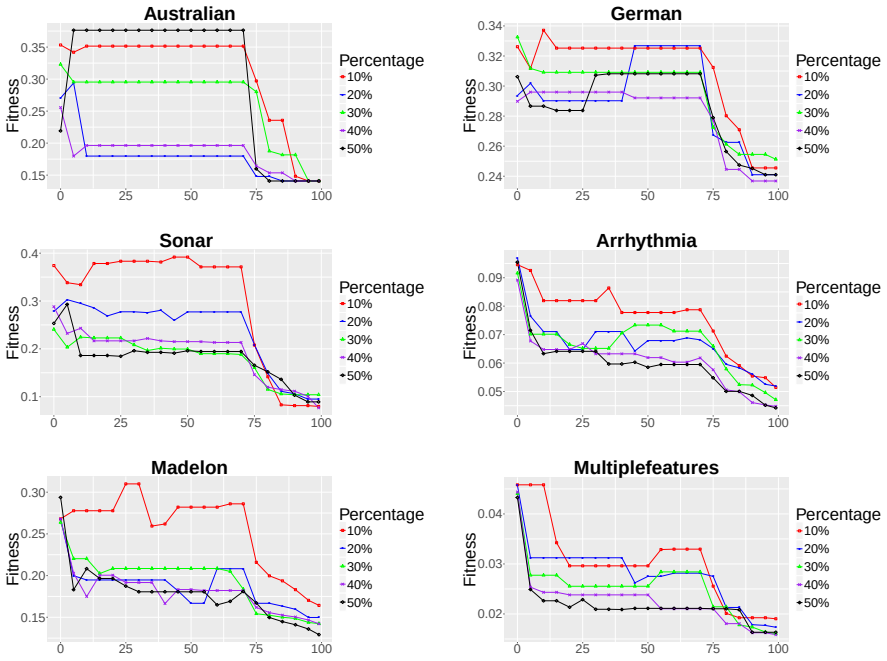


Figure 5.6: Real evolutionary processes on static surrogate models.

search regions, for each surrogate model (10%-50%), the evolutionary process of the best run is shown in Fig. 5.6. The horizontal axis is iterations and the vertical axis shows the fitness value of $gbest$ on each iteration. In the figure, the evolutionary processes of Australian, German, Sonar, Arrhythmia, Madelon and Multiple Features are shown. The evolutionary processes on the other 6 datasets have similar patterns. Note that in the first 75 iterations, particles are evaluated by the surrogate set, which means that in terms of the surrogate fitness value, the $gbest$ in the later iteration is always not worse than the $gbest$ in the earlier iterations. However, in the figure, the $gbest$ is re-evaluated by using the original training set, which does not guarantee that the $gbest$ in the later iteration always has similar or better original fitness value than the $gbest$ in the earlier iterations. Therefore, the less fluctuating evolutionary process shows that the corresponding surrogate model is more consistent with the original training set. By collating Table 5.3 and Fig. 5.6, it can be seen that usually

the method with the least fluctuating evolutionary process yields the best classification accuracy. For example, on the Arrhythmia dataset, the best training and testing accuracies are achieved by the 50% surrogate model, which has the least fluctuating evolutionary process.

Effect of the dynamic surrogate model

It is important to maintain the consistency between the surrogate and the original training sets. However, which surrogate set should be selected heavily depends on datasets. Therefore, the dynamic surrogate model is designed with an expectation of automatically selecting the most suitable surrogate training set during the evolutionary process. The results of the dynamic one are shown by the "Dyn" column in Table 5.3. A Wilcoxon signed rank test is used to compare the dynamic and other fixed-size surrogate models. "↑"/ "↓" or "∘" shows that the other algorithms are significantly better/worse or similar compared with the dynamic surrogate model.

In comparison with using the original training set, Dyn mostly achieves similar testing classification accuracies while selecting significantly smaller numbers of features. In comparison with other static surrogate models on the training set, the dynamic model (Dyn) usually achieves similar or better classification performance. Specifically, Dyn is significantly better than P-AGG on 5 datasets. From 10% to 30%, the number of datasets, on which Dyn is superior, ranges from 3 to 2. Dyn also achieves the similar training accuracy as 40% and 50% on 11 out of the 12 datasets. Similarly, on the test sets, except for 50%, Dyn achieves similar or better classification accuracies on most datasets. In terms of the number of selected features, Dyn often selects a smaller number of features than other static surrogate models.

Given 30%, 40% and 50% are the most promising static surrogate models, they are further compared with Dyn in terms of the best evolved fitness values, which is shown in Table 5.4. As can be seen from the table,

Table 5.4: Fitness values (x100) of different surrogate models

Dataset	30%	40%	50%	Dyn
Wine	**6.830**(○)	7.120(↓)	6.870(○)	6.940
Australian	15.21(○)	14.76(○)	14.77(○)	**14.50**
Vehicle	16.11(○)	16.10(○)	16.13(○)	**16.02**
German	26.24(○)	25.97(○)	**25.81**(○)	26.00
WBCD	6.270(↓)	5.850(↓)	**5.220**(↑)	5.520
Ionosphere	7.090(○)	7.120(○)	7.070(○)	**6.970**
Sonar	13.52(○)	**12.40**(○)	12.41(○)	13.01
Hillvalley	35.94(○)	35.93(○)	36.03(○)	**35.89**
Musk1	9.970(○)	9.540(○)	**9.080**(○)	9.310
Arrhythmia	5.330(↓)	5.100(↓)	**4.850**(○)	4.870
Madelon	16.98(○)	**15.89**(↑)	16.22(○)	16.45
Multiple Features	1.980(○)	1.950(○)	**1.890**(○)	**1.890**

Dyn achieves the best fitness value on five out of the 12 datasets, which is equal to 50%. The significant test shows that Dyn is similar or significantly better than 30% on all datasets. In comparison with 40%, Dyn is significantly better on three datasets while being worse on only one out of the 12 datasets.

The experimental results show that the dynamic model can adapt with different datasets to select the suitable surrogate training set, which results in similar or better classification performance while selecting a smaller number of features than using the whole training set and other static surrogate models on most datasets.

Computational time

In terms of computational time shown in Table 5.3, all surrogate models including the dynamic model are more efficient than using the original training set. Particularly, 10% is the most efficient model, which is about four times faster than using the original training set. 50%, the slowest static surrogate model is still two times more efficient than using the original training set. The dynamic model is a bit more efficient than 50%, which is also two times faster than using the original training set. This results

illustrates that the surrogate training set successfully reduces the computational cost while maintaining or even improving the quality of selected feature subsets.

5.5 Chapter Summary

This chapter investigates the effect of surrogate models on wrapper PSO-based feature selection. A surrogate training set is formed by selecting a subset of informative instances from the original training set. The surrogate set is used to assist PSO quickly estimate promising regions in the search space before the located regions are further explored using the original training set. Experimental results show that surrogate training sets containing from 30% to 50% training instances can help PSO to select smaller feature subsets with similar or better classification accuracy than using all features. Based on the static surrogate model, a dynamic model is developed to automatically select the suitable surrogate training set during the evolutionary process. The proposed dynamic model achieves similar classification accuracy while selecting significantly smaller feature subsets than static surrogate models on most datasets.

This chapter finds that the surrogate training set, which contains a subset of informative instances, can help wrapper PSO-based feature selection algorithms to find promising regions in a shorter time than using the whole training set. The surrogate training set brings improvements not only in efficiency but also in effectiveness, where the number of selected features is significantly reduced while the classification performance is maintained or even improved. The quality of surrogate training set plays an important role during the training process. Particularly, it is important to select enough informative instances while removing outliers, which can avoid underfitting, overfitting problems and reduce the number of selected features. More importantly, the chapter finds that in order to achieve good performance, the surrogate training set should be consis-

tent with the original training set. This is shown by the promising results achieved by a dynamic mechanism which updates the surrogate training set to maintain the consistency.

Although the dynamic surrogate model achieves good results, there are issues which can be investigated in the future. Firstly, the pool of surrogate training sets used in the dynamic surrogate model can be improved. Particularly, the ratios between the sizes of the surrogate training sets (in the pool) and the size of the original training set are fixed on all datasets. It can be seen that different datasets have their own characteristics such as the number of noisy instances, so it would be better if the pool is designed with respect to each dataset. Secondly, the surrogate training sets are built based on the original feature set. However, during the evolutionary process, the feature subset selected by each particle in each iteration can be different. Therefore, it would be better if the surrogate training sets can be updated during the evolutionary process with respect to the feature subsets selected by the particles so far. Last but not least, it can be seen that the surrogate models focus more on reducing the number of selected features while mainly maintaining the classification performance. It is not an easy task for a single-objective feature selection algorithm to simultaneously improve both objectives in feature selection due to the partial conflict between the two objectives. In the next chapter, a multi-objective algorithm for feature selection is developed to address the above problem.

Chapter 6

Decomposition-based Multi-objective Feature Selection

6.1 Introduction

Feature selection has two main objectives which are to minimize the classification error and the number of selected features. Since the two objectives are usually in conflict, feature selection can be considered a multi-objective problem. However, multi-objective feature selection has its own characteristics such as its highly discontinuous Pareto front, a strong preference for the objective of minimizing the classification error over reducing the number of features, and the two objectives are *not always* in conflict.

As a family of population-based optimization techniques, EC can be naturally applied to evolve a set of trade-off solutions for multi-objective problems, including feature selection. A number of different evolutionary multi-objective methods (EMO) have been proposed. Some methods evaluate candidate solutions by using a Pareto dominance relation together with a crowding distance to maintain the population's diversity, which are called Pareto dominance-based algorithms [223]. Non-dominated Sorting Genetic Algorithm (NSGA-II) [224], Strength Pareto Evolutionary Algorithm (SPEA2) [102], and OMOPSO [104] are well-known representatives

of this type of EMO algorithms. Pareto dominance-based algorithms work well on problems having two or three objectives but not on combinatorial problems. For example, Pareto dominance-based algorithms cannot find non-dominated solutions on the edges of the Pareto front for knapsack [225] and feature selection [214] problems. This is probably because the crowding distance has only a small effect in comparison with the Pareto dominance in the case of two or three objectives. Thus the population loses its diversity quickly at the beginning [223].

In contrast to dominance-based approaches, decomposition-based EMO has good search ability for combinatorial multi-objective problems [226, 227]. It decomposes a multi-objective problem into a number of single objective sub-problems and recombines the results. Decomposition-based algorithms often achieve better diversity than Pareto dominance-based algorithms, are easier to integrate with local searching mechanisms [223, 228], and may cope better with problems having many objectives [229, 230, 231] or complicated Pareto fronts [232, 233]. Thus, decomposition-based EMO may be more suitable to feature selection than Pareto dominance-based EMO. MOEA/D (Multi-objective Evolutionary Algorithm based on Decomposition) [31], a recently developed framework for multi-objective optimization, is a representative of decomposition-based EMO algorithms, which has been applied to many multi-objective problems. It decomposes a multi-objective problem to many scalar sub-problems using a set of weight vectors, where each weight vector defines a scalar sub-problem whose optimal solution will be a Pareto optimal solution to the original (full) problem. In this chapter, we will improve MOEA/D to deal with characteristics of multi-objective feature selection.

6.1.1 Chapter Goal

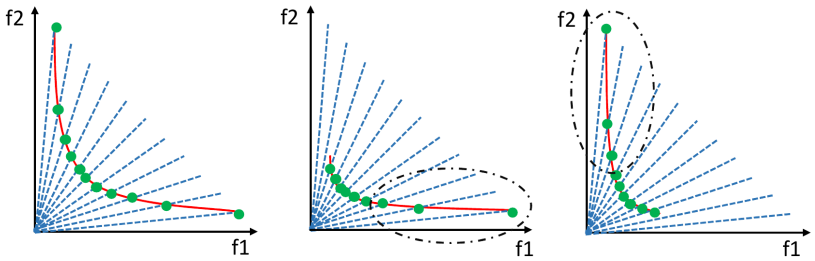
The overall goal of this chapter is to develop a new strategy for MOEA/D to decompose a multi-objective feature selection problem with an expec-

tation of obtaining a set of non-dominated feature subsets, which has a wide range of numbers of features and achieves better classification performance than using all features. In the proposed decomposition strategy, instead of using multiple weight vectors, multiple reference points are used to define the sub-problems. Based on the new decomposition, static and dynamic reference point methods are developed to identify conflicting regions, which are then focused on by allocating more resources to achieve better Pareto fronts. Both static and dynamic multiple reference point-based algorithms, called MOEA/D-STAT and MOEA/D-DYN, respectively, are compared with a standard MOEA/D algorithm and three Pareto dominance-based algorithms on 12 benchmark datasets of varying difficulties. Specifically, we will investigate:

- whether MOEA/D-STAT and MOEA/D-DYN can evolve feature subsets, which can achieve better performance than using all features,
- whether decomposition with reference points helps MOEA/D to improve the solution's quality and obtain a better approximation of the Pareto front than the multiple weight vectors decomposition,
- whether the new decomposition strategy generates more diverse feature subsets with various numbers of features than the three representatives of Pareto dominance-based algorithms: NSGA-II, SPEA2, and OMOPSO, and
- whether the dynamic reference points strategy can successfully recognize the conflicting regions and further improve Pareto front's quality by allocating more resources to these regions.

6.2 Proposed Algorithms

The section starts by summarizing the characteristics of feature selection that illustrate difficulties when applying MOEA/D to feature selection. It



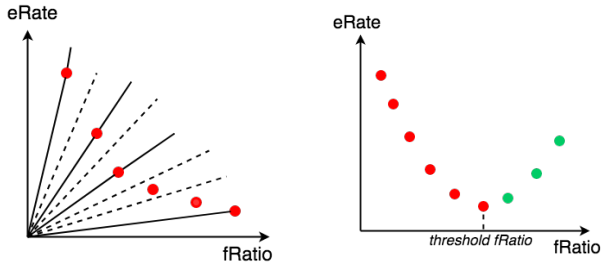
(a) Balance (b) Towards f_1 (c) Towards f_2

Figure 6.1: Examples of defining weights in MOEA/D

then shows how to use multiple reference points to decompose a feature selection problem, followed by two mechanisms (static and dynamic) to allocate the reference points. Finally, it introduces a repairing mechanism which helps to ensure that the decomposition's constraints are not violated and the population diversity is preserved.

6.2.1 Characteristics of Feature Selection

Feature selection is a multi-objective problem and the shape of its true Pareto front is unknown. Defining the weight vectors is essential since it strongly affects the performance of MOEA/D. However, it is not an easy task due to the dependence on the true Pareto front shape. A simple example is illustrated in Fig. 6.1, where the task is to minimize both objectives, f_1 and f_2, and the green dots show the best solutions for the weight vectors. When the Pareto front is biased towards f_1 (Fig. 6.1(b)) or f_2 (Fig. 6.1(c)), a set of evenly distributed weight vectors does not work well. Most solutions are obtained around the center of the Pareto front, and only a few solutions around the edge of the Pareto front are found. More weight vectors should be located near the edge to obtain more solutions there. Several interesting attempts have been made to adaptively update the weight vectors based on regions' densities to preserve the population diversity [234, 235]. However, most works require an additional computation step to adaptively adjust the weight vector set. This illustrates that it is not an



(a) Weight vector for FS (b) Not always conflicting

Figure 6.2: Characteristics of multi-objective feature selection

easy task to properly define the weight vectors in multi-objective feature selection.

In addition, feature selection has a highly discontinuous Pareto front. The task of feature selection is to reduce the classification error rate ($eRate$) while selecting a small portion of the original feature set ($fRatio$). $eRate$ measures the ratio between the number of wrongly classified instances and the total number of instances (m). $eRate$ is discrete and the interval between the adjacent values (i.e. the granularity) is $1/m$. Similarly, $fRatio$ is the ratio between the number of selected features and the total number of original features n. $fRatio$ is also discrete and the interval between its adjacent values is $1/n$. Therefore, the Pareto front of feature selection is highly discontinuous. If weight vectors are used to decompose feature selection, these vectors have to be carefully selected, otherwise, there will be vectors which do not correspond to any solution on the Pareto front as shown by the dashed line in Fig. 6.2(a). Furthermore, although both objectives are in the same range [0,1], they typically have different granularity due to the difference between $1/m$ and $1/n$. It has been shown that solving multi-objective problems where the objectives have different granularity usually results in imbalanced Pareto fronts [236].

The partially conflicting relationship between the two objectives in feature selection makes it an unusually challenging multi-objective problem. In feature selection, the classification performance is usually given a higher

priority. For example, if a feature set selects 10% more features than the other feature set but achieves 10% better accuracy, the first set is definitely preferred. Furthermore, the two objectives are not always in conflict. Specifically, removing irrelevant or redundant features from a feature set may improve the classification performance, which means that the two objectives are not conflicting in some regions. However, if all the features in a feature set are relevant and complementary, removing any feature reduces the classification performance. Only after removing all irrelevant/redundant features, the two objectives become conflicting. In other words, there might be a *threshold feature ratio* beyond which the two objectives are *mostly* harmonious. Fig. 6.2(b) illustrates the situation, in which each point is the best solution with the corresponding feature ratio. As can be seen in the figure, only the red points can form a Pareto front while all green points are dominated by the solution at the *threshold feature ratio*. It will be more effective for a multi-objective algorithm to allocate more computational efforts on regions with $fRatio$ below the *threshold*. However, the *threshold* is problem dependent and not easy to identify.

A basic assumption in decomposition-based EMO is that a wide variety of non-dominated solutions can be obtained over the Pareto front through a set of uniformly distributed weight vectors [105, 237, 238]. However, it is shown that some parts of a Pareto front can be more difficult to approximate than others [239]. It is natural to allocate resources (weight vectors) differently to different sub-problems with respect to their difficulties, which results in better efficiency [240]. A similar question appears in feature selection where the two objectives are not always conflicting on the whole objective space. Possibly, better Pareto fronts can be achieved by putting more efforts on the conflicting regions rather than evenly spending resources on both conflicting and non-conflicting regions.

Given the characteristics of feature selection, this chapter focuses on developing a new decomposition mechanism for feature selection, which can not only reduce the dependency on the Pareto front shape but also

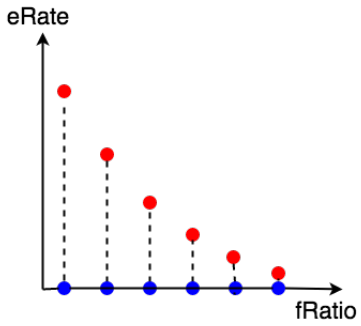


Figure 6.3: Multiple reference points in MOEA/D.

cope well with the front's high discontinuity and the complex relationship between its objectives. Particularly, in the following sections, a multiple reference point-based decomposition strategy is introduced to address the above characteristics of feature selection.

6.2.2 Decomposition with Multiple Reference Points

In standard MOEA/D, the effectiveness of the weight vector set depends on the shape of the true Pareto front which is unknown in feature selection. To reduce the effect of the Pareto front shape to produce more non-dominated feature subsets with different numbers of features, we use multiple reference points to decompose multi-objective feature selection problems instead of multiple weight vectors. Specifically, we allocate a set of R reference points on the $fRatio$ axis. A reference point placed at position *refRatio* on the $fRatio$ axis represents an idealized solution with an accuracy of 100% (i.e. 0% $eRate$) using exactly ($\lfloor refRatio \times n \rfloor$) features, where n is the total number of the original features. In the MOEA/D search, there will be one individual in the population for each reference point, just as there is one individual for each weight vector when the problem is decomposed using weight vectors. Fig. 6.3 shows a set of reference points marked by blue dots. Using multiple reference points, the multi-objective feature selection problem is decomposed into a sub-problem for each ref-

erence point. The solution of a sub-problem for a reference point at *refRatio* is the feature subset, whose size is at most ($n_{ref} = \lfloor refRatio \times n \rfloor$). Such a feature subset will be on the true Pareto front. The search space of each sub-problem is smaller than the original one since it is limited by the number of features defined by the corresponding reference point.

The fitness function of a candidate feature subset S to a sub-problem is designed as follows.

$$fitness_S = eRate_S + 100 \times max(|S| - n_{ref}, 0) + 0.01 \times fRatio_S \quad (6.1)$$

where $|S|$ is the number of selected features. The main task of the sub-problem is to minimize the classification error $eRate_S$, which is the first component. The second component is a penalty factor to ensure the condition that the number of selected features in S should not exceed n_{ref}. The last component shows a very weak preference for a smaller feature subset among feature subsets with the same classification error and different numbers of features. The fitness function actually gives a higher priority to reduce the classification error. The feature subset with a few more features but achieving significantly higher classification accuracy is still preferred as long as the number of selected features is not larger than n_{ref}.

A decomposition using weight vectors in a highly discontinuous space may lead to a sub-problem with no solutions, and may result in a very poor approximation of the Pareto front. In contrast, a decomposition using reference points leads to sub-problems that always have a solution from the Pareto front, and therefore should always give a good approximation of the true Pareto front. Because of the choice of the fitness function, this decomposition also handles the strong preference for classification performance in feature selection.

The idea of using multiple reference points in MOEA/D has already been examined in some studies [241, 242]. However, in those algorithms, the positions of the reference points need to be updated every generation

according to specific mechanisms. In our approach, the reference points are placed on the $fRatio$ axis prior to the evolutionary process. Moreover, there is no weight vector in the proposed algorithm. These two differences make our algorithm simpler than other multiple reference point-based EMO algorithms.

6.2.3 Reference Points Allocation

The previous section showed how multiple reference points can be used to effectively decompose feature selection despite its discontinuous Pareto front. This section describes how the reference points are allocated on the $fRatio$ axis. One way is to fix locations of the reference points at the beginning, which is called $static$ allocation. A more advanced strategy is to *dynamically* modify the locations, which is capable to detect conflicting/non-conflicting regions.

Static allocation

In the static allocation, the reference points are uniformly placed on the $fRatio$ axis and do not change during the search. Specifically, given R reference points, the position of the i^{th} reference point is $(i/R, 0)$. Notice that there is no reference point at the location $(0, 0)$ since it defines an empty feature subset.

In MOEA/D, a neighborhood is an important characteristic, which allows the transfer of information to improve the candidate solutions. For each sub-problem, its neighbors are sub-problems whose reference points are close to this sub-problem's reference point. For example, when the number of neighbors is 3, the neighborhood of $(3/R, 0)$ includes $(2/R, 0)$, $(3/R, 0)$ and $(4/R, 0)$. In general, we expect that the solutions of neighboring sub-problems will be similar, which is an important requirement of MOEA/D.

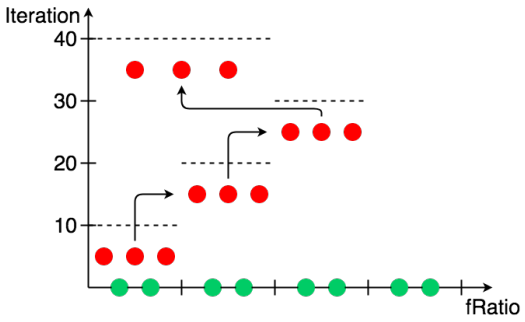


Figure 6.4: Dynamic reference points example: $fixed$ points are green, $moving$ points are red, dashed line shows the interval that $moving$ points are located in the corresponding iterations.

Dynamic allocation

In feature selection, the two objectives are not always in conflict, and in each of such regions, there can be only one solution from the true Pareto front. Evenly distributing all reference points on the entire domain of the $fRatio$ axis might limit the performance of MOEA/D since some reference points are wasted in non-conflicting regions. Therefore, we propose a dynamic mechanism that firstly identifies the conflicting and non-conflicting region, and then allocates more reference points to the conflicting regions.

In order to achieve the above goal, the $fRatio$ axis is divided into I intervals, all of the same length, $1/I$. We assume that there will be one interval containing the *threshold feature ratio*, beyond which the two objectives are *mostly* not conflicting (Fig. 6.2(b)). The R reference points are divided into F $fixed$ points and M $moving$ points ($R = F + M$). The F fixed points are evenly located across the I intervals, shown by the green points in Fig. 6.4. In the beginning, the M moving points are all located on the first interval, and the locating mechanism spreads the moving points while avoiding overlapping between the two types of reference points as much as possible. After a certain number of iterations defined by the division between the maximum number of iterations and the number of intervals, the moving points are re-allocated on the next interval. For example,

in Fig. 6.4, in the first 10 iterations, the three moving points are located on the first interval. In the next 10 iterations, the moving points are re-allocated to the second interval and so on. Here, the $10^{th}, 20^{th}...$ iterations are called *boundary iterations*, since on these iterations the moving points are re-allocated.

The re-allocation process is continued until the algorithm detects that the two objectives are potentially not conflicting anymore. As can be seen in Fig. 6.2(b), most solutions in the potentially non-conflicting regions (green) are dominated by a solution in the conflicting region (red). Therefore, to determine whether the two objectives are still conflicting in the i^{th} interval, the solution with the lowest classification error in the interval is compared with all solutions from the previous interval. If the solution from the i^{th} interval is dominated by a solution in the previous interval, the algorithm assumes the two objectives are not conflicting in any further interval from the i^{th} one. The moving points are then evenly allocated on all the intervals prior to the i^{th} one and their locations are not changed until the evolutionary process is finished. An example is given in Fig. 6.4, where after allocating moving points on the third interval, the algorithm finds that the solution with the best accuracy obtained by reference points in the third interval is dominated by one of the solutions from the second interval. This is an indication that in the regions from the third interval, the two objectives may not conflict. Thus the algorithm allocates all moving points on the first and second intervals.

In the evolutionary process, the moving points are re-allocated many times. However, Giagkiozis et al. [243] showed that $dynamic$ mechanisms are not always good since they may cause divergence in the population. To avoid the divergence but still preserve the population's diversity, the re-allocation process has to be done carefully. Firstly, the moving points are re-allocated so that there will be the least overlap with the fixed points because a diverse allocation usually leads to a diverse solution on the Pareto front. Secondly, when reallocating a reference point to a new value on the

$fRatio$ axis, the algorithm attempts to preserve as much information from the solutions found for the sub-problem at the previous location of the reference point. Therefore, the algorithm initializes the reference point with a feature subset as close as possible to the feature subset from the previous solution. Since the new location requires a different number of features, the feature subset from the previous solution must be "repaired", which will be discussed in the following section.

It should be noted that the dynamic mechanism does not have to ensure that the *threshold* interval is found exactly. It just needs to estimate possible regions in which the two objectives are mostly conflicting and puts more effort (reference points) on these regions. There are still some fixed reference points locating in possible non-conflicting regions just in case the estimation is not good enough. In addition, these fixed points on non-conflicting regions usually have large n_{ref}, which may allow different features to be introduced into solutions for neighboring reference points with smaller n_{ref} values. This helps to prevent premature convergence of the sub-problems in the conflicting regions.

6.2.4 Repairing Mechanism

All evolutionary algorithms create new candidate solutions out of old solutions. If it is possible for the new candidates to be invalid (by not satisfying constraints), then the searching mechanism is in danger of wasting a lot of search time on exploring infeasible regions of the search space. One option is to identify and remove any invalid candidates, but this may lose valuable information contained in the candidates. An alternative option is to "repair" an invalid candidate by transforming it into a similar valid solution, which has the advantage of retaining information in the candidate, but may be expensive if the repair mechanism is not efficient.

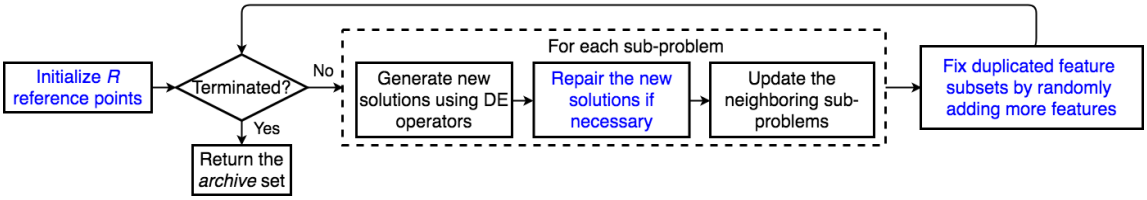
For sub-problems in the proposed MOEA/D-based feature selection algorithm, repair is particularly important because each sub-problem corresponds to a small part of the search space - the sub-space of features sets

whose size is close to but not more than the n_{ref} of the reference point - and it is difficult to ensure that new candidates are always within the subspace. When the searching mechanism creates a candidate feature set S that is larger than n_{ref}, the repairing mechanism must remove ($|S| - n_{ref}$) features in order to make it valid. The mechanism chooses the ($|S| - n_{ref}$) features with the lowest individual classification accuracies (which is precalculated at the start of the algorithm). A potential problem with this approach is that it may remove features that are strongly complementary to other features, even though they are individually weak. However, this is not a big problem since the information about complementary features will usually be retained in the neighboring sub-problems with larger n_{ref} values, and the searching mechanism will be able to re-select the removed features using information from the neighboring subproblems.

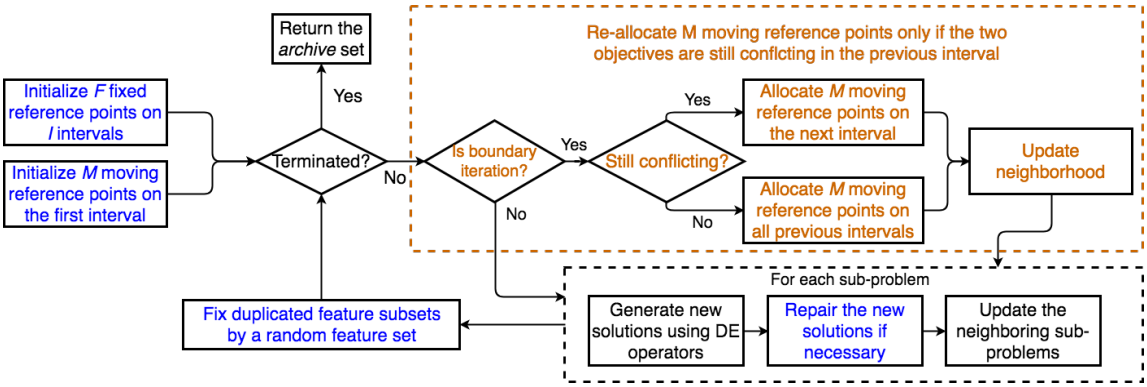
Re-allocating reference points in the dynamic mechanism is even more prone to creating invalid candidates since re-allocating a reference point means changing its n_{ref}. If a reference point is re-allocated to a smaller n_{ref} value, its current candidate feature subset will most likely be too large for the new n_{ref}, and features will be removed by the same mechanism described above. If a reference point is re-allocated to a larger n_{ref}, its candidate feature set is still valid but may be much smaller than the new n_{ref}, which is highly problematic because it is likely to be similar to the candidates of subproblems with smaller reference points; and therefore will reduce the population diversity and limit the ability to explore new feature combinations. Therefore, the repair process will rank all unselected features based on their individual classification accuracies and sequentially add them to the candidate feature set until its size reaches n_{ref}.

6.2.5 Fixing Duplicated Feature Subsets

One problem of the proposed decomposition approach is that feature subsets for reference points with smaller n_{ref} values can also become solu-



(a) Static multiple reference point-based strategy (MOEA/D-STAT).



(b) Dynamic multiple reference point-based strategy (MOEA/D-DYN).

Figure 6.5: Overall multiple reference point-based MOEA/D algorithms.

tions for reference points with larger n_{ref} values. The duplicated feature subsets might cause a low diversity and premature convergence. In order to avoid the situation, all duplicated sets from the larger reference points are repaired. Since in the static strategy, the reference points density on an interval is not high, randomly adding unselected features to the duplicated subsets should be sufficient. However, due to the dynamic allocation of moving reference points, the reference points on a particular interval is denser, so the duplicated feature subset in the dynamic strategy is replaced by a newly random feature subset.

6.2.6 Overall Proposed Algorithms

Fig. 6.5 shows an overview of the two proposed multiple reference point-based MOEA/D algorithms for feature selection. The static one is illus-

trated in Fig. 6.5(a), where the blue parts are the essential differences in comparison with the standard MOEA/D algorithm. Fig. 6.5(b) presents the dynamic multiple reference points. The difference between the dynamic mechanism and the static one is the moving reference points re-allocation, marked by the orange color. Note that the re-allocation is only performed when the algorithm has not identified the *threshold* interval yet. Once the *threshold* interval is found, the M moving reference points are allocated to the conflicting intervals and no further re-allocation is needed. In this chapter, MOEA/D uses the differential evolutionary (DE) crossover operator and polynomial mutation operators to generate new candidate solutions, which is an efficient approach to preserve the population diversity for complicated Pareto fronts [31].

The pseudo-code of the static multiple reference point-based MOEA/D for feature selection (MOEA/D-STAT) is shown in Algorithm 3. In this chapter, each individual is represented by a vector of real numbers. The vector length is equal to the total number of original features. Each entry corresponds to an original feature and its value determines whether or not the corresponding feature is selected. Specifically, the feature is chosen if and only if the entry's value is greater than a threshold θ. During the evolutionary process, σ is the probability that a sub-problem selects its T neighbor sub-problems to update its solution.

In this chapter, the Tchebycheff approach [31] is used as a representative of standard MOEA/D to compare with the proposed multiple reference point decomposition, since it usually achieves better results than the Weighted Sum approach [31] and it does not need to specify a penalty factor like the Boundary Intersection approach. In addition, the Tchebycheff approach has good theoretical properties [244].

Algorithm 3 : Pesudo-code of MOEA/D-STAT

begin

Calculate the classification accuracy of each feature on the training set;

Initialize R reference points: $refPoint_i = (i \times \frac{1}{R}, 0)$ where $i = 1, ..., R$;

Compute the Euclidean distance between any two reference points;

Find the set of T neighboring reference points of each reference point;

Each i^{th} sub-problem's neighboring set is denoted B_i;

Randomly initialize the population $P = (p_1, p_2, ..., p_R)$ where p_i is the candidate solution of the i^{th} sub-problem;

while maximum iteration is not reached **do**

for i=1,...,R **do**

$$Ne = \begin{cases} B(i) \text{ if } rand < \sigma \\ P \text{ otherwise} \end{cases}$$

Randomly select two solutions p_n and p_m from Ne;

Apply DE crossover to generate a solution y_c. The j^{th} entry of y_c is calculated as:

$$y_{cj} = \begin{cases} p_{ij} + F \times (p_{ij} - p_{kj}) & \text{with probability } CR \\ p_{ij} & \text{with probability } 1 - CR \end{cases}$$

Apply polynomial mutation to generate new solution y_m from the solution y_c;

Repair y_m if it selects more than n_{ref} features;

Update solutions of neighboring sub-problems if y_m is better than the solutions of sub-problems in terms of the fitness values calculated by Eq. (6.1);

end for

Repair the duplicated feature subsets;

Update the archive set;

end while

Output the archive set;

end

6.3 Experiment Design

6.3.1 Benchmark Techniques

The proposed algorithms, MOEA/D-STAT and MOEA/D-DYN, are compared with standard MOEA/D, NSGA-II [224], SPEA2 [102] and OMOPSO [104] on 12 datasets selected from the UCI machine learning repository [32]. The datasets are chosen so that they have different numbers of features, classes and instances, which can be seen in Table 1.1. In all algorithms, the candidate solutions are evaluated by K-nearest neighbor (KNN) where K=5, which ensures that KNN might avoid outliers while still having a good efficiency. Each algorithm is executed for 50 independent runs. In each run, the datasets are divided into training and test sets with the proportions of 70% and 30%, respectively. During the training process, KNN with 10-fold cross-validation is applied to calculate the classification error rate on the training set to avoid feature selection bias. The evolved feature subsets are then evaluated on the test set to obtain their testing accuracies. These settings are commonly used in feature selection [186, 245].

In order to examine the performance of the six multi-objective algorithms, the hypervolume indicator [109] and inverted general distance (IGD) indicator [246] are used. In each run, an algorithm obtains two Pareto fronts, which are the training Pareto front and the testing Pareto front. The two fronts contain two sets of non-dominated feature subsets evaluated on the training and test sets. Therefore, after 50 runs, each algorithm has two sets of metric values based on the training and test sets, respectively. In order to calculate the two indicators, it is necessary to know the true Pareto front, but it is not known in feature selection. Therefore, the true Pareto front is approximated by the non-dominated solutions obtained from the union of all solutions generated by the six algorithms in the 50 independent runs. Here, the hypervolume values of a Pareto front is calculated by its inverted front, which is implemented in the JMetal

package [247]. Therefore, the larger the hypervolume value, the better the algorithm. A significance test, Wilcoxon test with its significance level set to 0.05, is used to compare the performance between MOEA/D-STAT, MOEA/D-DYN, and the benchmark algorithms.

For each algorithm the Pareto front corresponds to the median hypervolume value is obtained, which is called a $median$ front. Note that although the $median$ front can give a good visualization, the indicator values are a more reliable measure to compare different algorithms. The reason is that the indicator values are calculated based on all the solutions produced by each algorithm in the 50 independent runs, but the median front in figures only show the median non-dominated solutions, which is obtained from a single run. We visualized the median fronts to provide a visual intuition of the patterns.

6.3.2 Parameter settings

Choosing a proper parameter setting for MOEA/D is a difficult task since it is problem-dependent. After doing several experiments with some suggestions from [31], the parameters are set as below. The number of neighbors T is set to $R/10$, which is much smaller than the population size to preserve diversity. However, the smallest value of T is 4 to ensure the diversity between neighboring sub-problems. The maximum number of solutions that are replaced by a newly generated subset is set to 1 since it is recommended that this number should be much smaller than T [31]. Both MOEA/D algorithms use DE crossover and polynomial mutation, where the crossover rate is 0.6 and the mutation rate is $1/n$. In the DE crossover, the scaling factor F is 0.7, which lies in the recommended range [0.6,0.8]. The probability of selecting parents from the neighbor sub-problems, σ, is 0.85. The parameter settings of NSGA-II, SPEA2, and OMOPSO are set to the recommended setting from their original papers, which are default settings in the JMetal package [247].

The dynamic strategy has two main parameters: M - the number of moving reference points, I - the number of intervals. Based on experiments, M is set to $0.4 * R$, which ensures the significant effect of moving reference points while maintaining enough fixed reference points to explore all intervals. If the number of features is less than 20, then the number of intervals is set to 9. Otherwise, the number of intervals is set to 4. On datasets having less than 20 features, since the search space corresponding to each interval it not large, it is fine to have 9 intervals, which ensures a fine-grained intervals leading to a more accurate estimation of non-conflicting regions. However, on datasets with large numbers of features, the search space corresponding to each interval is much larger, which requires more efforts (time and reference points) to be well explored. Since the fewer intervals results in larger numbers of iterations in which the moving points explore an interval, the number of intervals on datasets with large numbers of features is set to 4, a small value.

The number of nearest neighbors in KNN is set to 5 to avoid noise instances while still maintaining its efficiency. For all algorithms, the maximum number of iterations is 200. The population size is set to the number of features due to the exponential increase of the search space size with respect to the number of features. However, the population size is bounded by 200 to avoid a high computational cost. The threshold θ is set to 0.6 so that the algorithms start with slightly small numbers of features.

6.4 Results

The IGD values of the six algorithms on the training and test sets are shown in Tables 6.1 and 6.2, respectively. Tables 6.3 and 6.4 show the average hypervolume values of each algorithm on the training and test sets, respectively. The two signs beside the average value of the four benchmark algorithms show results of the significance test comparing between them and the two proposed algorithms, MOEA/D-STAT and MOEA/D-DYN,

respectively. "↑"/ "↓" / "∘" mean that MOEA/D-STAT or MOEA/D-DYN is significantly better/worse than or similar to the corresponding benchmark algorithm. For MOEA/D-STAT, the single sign on its column shows the comparison between it and MOEA/D-DYN.

The $median$ fronts of the algorithms on the training and test sets are shown in Figs. 6.6 and 6.7, respectively. In each sub-figure, the two numbers inside the brackets show the total number of original features and the training or testing error when using all features. The horizontal and vertical axes represent $fRatio$ and $eRate$, respectively. Six datasets are selected as representatives of small (Australian, Vehicle), medium (Musk1, Arrhythmia) and large (Madelon, Multiple Features) datasets. The trends of Pareto fronts are similar on the other datasets.

6.4.1 Comparison with Using All Features

As can be seen from Figs. 6.6 and 6.7, on all datasets, the feature subsets evolved by both MOEA/D-STAT and MOEA/D-DYN contain at most 60% original features, and select at least three feature subsets, which are better than using all features on both training and test sets. Especially, on Arrhythmia and Madelon, all subsets selected by the two algorithms achieve better classification performance than using all features while selecting less than 10% original features.

The results suggest that on all datasets, applying multiple reference points to MOEA/D-based feature selection can select a small number of features while still achieving better performance than using all features.

6.4.2 MOEA/D-STAT vs Other EMO Methods

On the training set, as shown in Table 6.1, in terms of the IGD indicator, in most cases all other algorithms are significantly worse than the static multiple-reference points strategy. Only on the Hillvalley, NSGAII and SPEA2 achieve better IGD values, while OMOPSO and standard MOEA/D have the same IGD values as MOEA/D-STAT on at most three out of

Table 6.1: IGD on training sets.

Dataset	NSGAII	SPEA2	OMOPSO	MOEA/D	MOEA/D-STAT	MOEA/D-DYN
Wine	0.049±0.010 (↓ ↓)	0.036±0.011 (↓ ↓)	0.027±0.008 (↓ ↓)	0.036±0.011 (↓ ↓)	**0.018±0.013** (○)	0.023±0.011
Australian	0.029±0.015 (↓ ↓)	0.030±0.013 (↓ ↓)	0.010±0.010 (↓ ↓)	0.024±0.016 (↓ ↓)	0.003±0.007 (○)	**0.002±0.005**
Vehicle	0.021±0.011 (↓ ↓)	0.026±0.010 (↓ ↓)	0.015±0.010 (↓ ↓)	0.020±0.012 (↓ ↓)	**0.004±0.003** (↑)	0.006±0.004
German	0.056±0.015 (↓ ↓)	0.051±0.017 (↓ ↓)	0.045±0.017 (↓ ↓)	0.037±0.021 (↓ ↓)	**0.023±0.018** (○)	0.025±0.021
WBCD	0.012±0.011 (↓ ↓)	0.012±0.011 (↓ ↓)	0.007±0.010 (○ ↓)	0.015±0.010 (↓ ↓)	0.009±0.009 (↓)	**0.000±0.002**
Ionosphere	0.009±0.008 (↓ ↓)	0.008±0.008 (↓ ↓)	0.002±0.002 (○ ↓)	0.007±0.005 (↓ ↓)	0.001±0.003 (↓)	**0.000±0.000**
Sonar	0.016±0.003 (↓ ↓)	0.015±0.003 (↓ ↓)	0.015±0.003 (↓ ↓)	0.014±0.003 (↓ ↓)	0.010±0.002 (↓)	**0.009±0.003**
Hillvalley	0.005±0.002 (↑ ↓)	0.006±0.002 (↑ ↓)	0.007±0.003 (○ ↓)	0.006±0.001 (○ ↓)	0.006±0.001 (↓)	**0.005±0.001**
Musk1	0.008±0.001 (↓ ↓)	0.007±0.002 (○ ○)	0.010±0.002 (↓ ↓)	0.007±0.001 (○ ○)	**0.007±0.001** (↑)	0.007±0.001
Arrhythmia	0.003±0.001 (↓ ↓)	0.002±0.001 (↓ ↓)	0.003±0.001 (↓ ↓)	0.002±0.000 (↓ ↓)	0.002±0.000 (↓)	**0.002±0.000**
Madelon	0.024±0.001 (↓ ↓)	0.023±0.001 (↓ ↓)	0.018±0.005 (↓ ↓)	0.013±0.003 (↓ ↓)	0.007±0.001 (↓)	**0.004±0.001**
MultipleFeatures	0.008±0.001 (↓ ↓)	0.007±0.001 (↓ ↓)	0.010±0.003 (↓ ↓)	0.004±0.001 (↓ ↓)	0.003±0.000 (↓)	**0.001±0.000**

Table 6.2: IGD on test sets.

Dataset	NSGAII	SPEA2	OMOPSO	MOEA/D	MOEA/D-STAT	MOEA/D-DYN
Wine	0.081±0.020 (↓ ↓)	0.014±0.015 (○ ↓)	0.009±0.002 (○ ○)	0.018±0.014 (↓ ↓)	0.009±0.006 (○)	**0.008±0.000**
Australian	0.044±0.031 (↓ ↓)	0.049±0.034 (↓ ↓)	0.024±0.015 (↓ ↓)	0.037±0.029 (↓ ↓)	**0.016±0.007** (○)	0.018±0.004
Vehicle	0.023±0.011 (↓ ○)	0.028±0.015 (↓ ○)	0.025±0.012 (↓ ○)	0.026±0.014 (↓ ○)	**0.018±0.010** (↑)	0.023±0.011
German	0.075±0.016 (○ ○)	0.077±0.017 (○ ○)	0.080±0.018 (↓ ↓)	0.074±0.022 (○ ○)	0.070±0.028 (○)	**0.069±0.025**
WBCD	0.007±0.009 (○ ↓)	0.008±0.011 (○ ↓)	0.003±0.003 (↑ ○)	0.008±0.010 (○ ↓)	0.007±0.009 (↓)	**0.003±0.002**
Ionosphere	**0.024±0.006 (↑ ↑)**	0.027±0.005 (○ ↑)	0.027±0.004 (○ ↑)	**0.024±0.006 (↑ ↑)**	0.028±0.003 (○)	0.029±0.000
Sonar	0.032±0.009 (○ ○)	0.029±0.006 (○ ○)	**0.028±0.007** (↑ ↑)	0.029±0.007 (○ ○)	0.031±0.005 (○)	0.031±0.005
Hillvalley	0.011±0.004 (↑ ○)	**0.011±0.002** (↑ ↑)	0.013±0.003 (↑ ○)	0.013±0.004 (↑ ○)	0.015±0.004 (↓)	0.013±0.004
Musk1	0.021±0.003 (↓ ↓)	0.020±0.003 (↓ ↓)	0.022±0.004 (↓ ↓)	0.018±0.003 (↓ ↓)	**0.015±0.004** (○)	**0.015±0.004**
Arrhythmia	0.005±0.001 (↓ ↓)	0.005±0.001 (○ ↓)	0.006±0.002 (↓ ↓)	**0.004±0.001** (○ ○)	0.005±0.001 (↓)	**0.004±0.001**
Madelon	0.048±0.001 (↓ ↓)	0.047±0.003 (↓ ↓)	0.036±0.011 (↓ ↓)	0.019±0.006 (↓ ↓)	0.014±0.002 (↓)	**0.008±0.001**
MultipleFeatures	0.011±0.001 (↓ ↓)	0.010±0.001 (↓ ↓)	0.014±0.003 (↓ ↓)	0.005±0.001 (↓ ↓)	0.005±0.001 (↓)	**0.001±0.000**

Table 6.3: Hypervolume on training sets

Dataset	NSGAII	SPEA2	OMOPSO	MOEA/D	MOEA/D-STAT	MOEA/D-DYN
Wine	0.751±0.056 (↓ ↓)	0.870±0.020 (↓ ↓)	0.876±0.001 (↓ ↓)	0.872±0.005 (↓ ↓)	**0.877±0.001** (○)	**0.877±0.001**
Australian	0.778±0.015 (↓ ↓)	0.782±0.008 (↓ ↓)	0.794±0.003 (○ ↓)	0.783±0.019 (↓ ↓)	0.794±0.002 (↓)	**0.795±0.000**
Vehicle	0.795±0.009 (↓ ↓)	0.794±0.010 (↓ ↓)	0.801±0.002 (○ ↓)	0.796±0.006 (↓ ↓)	0.801±0.001 (↓)	**0.802±0.001**
German	0.709±0.013 (↓ ↓)	0.707±0.016 (↓ ↓)	0.717±0.004 (○ ↓)	0.713±0.006 (↓ ↓)	0.718±0.004 (↓)	**0.719±0.003**
WBCD	0.916±0.009 (↓ ↓)	0.917±0.006 (↓ ↓)	0.919±0.001 (○ ↓)	0.918±0.002 (↓ ↓)	0.920±0.001 (↓)	**0.920±0.000**
Ionosphere	0.899±0.014 (↓ ↓)	0.900±0.017 (↓ ↓)	0.910±0.005 (↓ ↓)	0.901±0.010 (↓ ↓)	0.912±0.003 (↓)	**0.912±0.000**
Sonar	0.871±0.014 (↓ ↓)	0.867±0.013 (↓ ↓)	0.867±0.012 (↓ ↓)	0.869±0.012 (↓ ↓)	0.887±0.007 (○)	**0.889±0.008**
Hillvalley	0.617±0.007 (↓ ↓)	0.616±0.004 (↓ ↓)	0.611±0.007 (↓ ↓)	0.614±0.007 (↓ ↓)	0.620±0.004 (↓)	**0.625±0.003**
Musk1	0.919±0.010 (↓ ↓)	0.924±0.007 (↓ ↓)	0.898±0.014 (↓ ↓)	0.929±0.005 (↓ ↓)	**0.933±0.004** (○)	0.932±0.004
Arrhythmia	0.940±0.006 (↓ ↓)	0.949±0.005 (↓ ↓)	0.940±0.012 (↓ ↓)	0.955±0.002 (↓ ↓)	**0.957±0.001** (○)	**0.957±0.001**
Madelon	0.874±0.011 (↓ ↓)	0.883±0.009 (↓ ↓)	0.863±0.018 (↓ ↓)	0.849±0.011 (↓ ↓)	0.891±0.004 (↓)	**0.896±0.003**
MultipleFeatures	0.951±0.007 (↓ ↓)	0.960±0.008 (↓ ↓)	0.933±0.016 (↓ ↓)	0.974±0.006 (↓ ↓)	0.991±0.000 (↓)	**0.994±0.000**

Table 6.4: Hypervolume on test sets.

Dataset	NSGAII	SPEA2	OMOPSO	MOEA/D	MOEA/D-STAT	MOEA/D-DYN
Wine	0.754±0.058 (↓ ↓)	0.894±0.029 (↓ ↓)	0.904±0.003 (○ ○)	0.890±0.019 (↓ ↓)	0.903±0.006 (○)	**0.904±0.000**
Australian	0.747±0.061 (↓ ↓)	0.739±0.065 (↓ ↓)	0.781±0.022 (↓ ↓)	0.760±0.055 (↓ ↓)	**0.791±0.006** (○)	0.790±0.004
Vehicle	0.791±0.011 (↓ ↓)	0.788±0.012 (↓ ↓)	0.797±0.004 (↑ ○)	0.793±0.009 (○ ↓)	0.795±0.004 (↓)	**0.798±0.003**
German	0.669±0.022 (↓ ↓)	0.671±0.018 (↓ ↓)	0.678±0.010 (○ ○)	0.673±0.014 (↓ ↓)	0.680±0.007 (○)	**0.680±0.006**
WBCD	0.909±0.012 (○ ↓)	0.908±0.014 (○ ↓)	0.914±0.001 (↑ ○)	0.908±0.012 (○ ↓)	0.912±0.005 (↓)	**0.914±0.000**
Ionosphere	**0.852±0.019** (↑ ↑)	0.845±0.015 (○ ○)	0.844±0.011 (○ ○)	0.851±0.016 (↑ ↑)	0.842±0.009 (○)	0.839±0.000
Sonar	0.774±0.031 (↓ ↓)	0.782±0.022 (↓ ↓)	0.790±0.027 (○ ○)	0.790±0.027 (○ ○)	**0.798±0.021** (○)	0.793±0.022
Hillvalley	0.595±0.013 (↑ ○)	**0.598±0.010** (↑ ○)	0.589±0.012 (○ ↓)	0.593±0.012 (○ ○)	0.590±0.010 (↓)	**0.598±0.011**
Musk1	0.846±0.019 (↓ ↓)	0.857±0.015 (↓ ↓)	0.834±0.025 (↓ ↓)	0.860±0.013 (↓ ↓)	0.868±0.010 (○)	**0.872±0.010**
Arrhythmia	0.934±0.007 (↓ ↓)	0.943±0.005 (↓ ↓)	0.935±0.012 (↓ ↓)	0.951±0.002 (↓ ↓)	**0.952±0.002** (○)	**0.952±0.002**
Madelon	0.860±0.011 (↓ ↓)	0.869±0.009 (↓ ↓)	0.857±0.016 (↓ ↓)	0.849±0.011 (↓ ↓)	0.883±0.004 (↓)	**0.886±0.004**
MultipleFeatures	0.947±0.007 (↓ ↓)	0.956±0.008 (↓ ↓)	0.929±0.016 (↓ ↓)	0.971±0.006 (↓ ↓)	0.987±0.001 (↓)	**0.990±0.001**

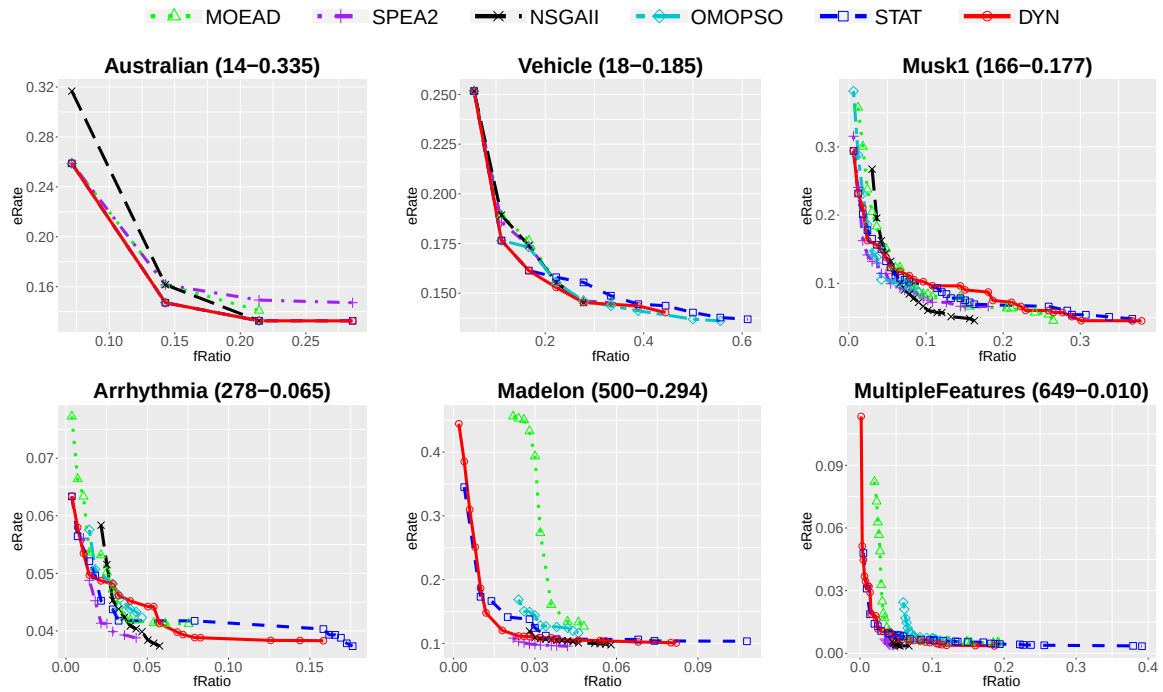


Figure 6.6: Median fronts on training sets.

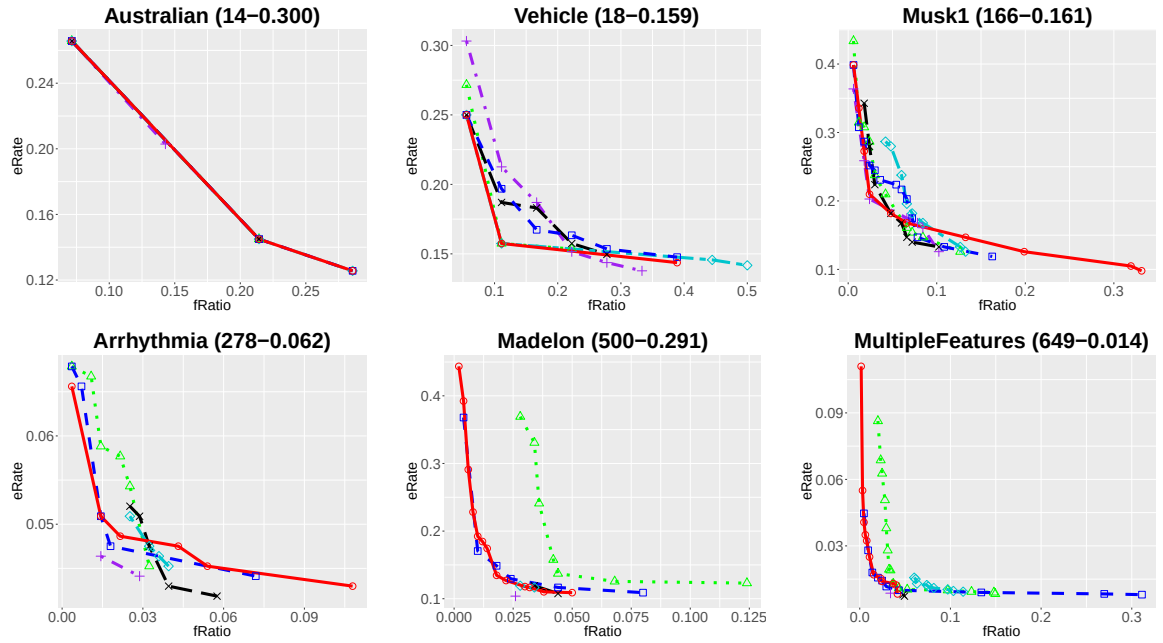


Figure 6.7: Median fronts on test sets.

the 12 datasets. In terms of hypervolume values as shown in Table 6.3, MOEA/D-STAT is significantly better than NSGAII, SPEA2 and standard MOEA/D on all datasets, as shown in Table 6.3. Among the benchmark algorithms, SPEA2 achieves the best hypervolume values on the medium and large datasets while OMOPSO has the best performance on the small datasets (similar performance with MOEA/D-STAT on four small datasets). But when there is a large number of features, MOEA/D-STAT significantly outperforms all other benchmark algorithms. The possible reasons can be seen in Fig. 6.6. On the small datasets, MOEA/D-STAT evolves the similar shapes with the four benchmark algorithms, but for the same number of features, MOEA/D-STAT tends to achieve lower classification errors. It is mainly because the decomposition along with feature ratios makes the search space of each sub-problem much smaller than the original search space. Although the upper bound feature ratio limits the number of features, MOEA/D-STAT still allows to replace worse features in the current subset by better features through communications with its neighboring sub-problems. On the larger datasets such as MultipleFeatures, the fronts become significantly different. Dominance-based algorithms quickly lose their diversities and their Pareto fronts focus mostly on the middle of the objective search space. OMOPSO is the worst dominance-based algorithms while SPEA2 and NSGAII have very similar Pareto fronts. On the other hand, decomposition-based algorithms achieve much more diverse and evenly distributed Pareto-front. Even standard MOEA/D's solutions have many more different feature ratios than NGSAII and SPEAII. MOEA/D-STAT also evolves as diverse Pareto fronts as standard MOEA/D. However, given the same number of features, MOEA/D-STAT always achieve better classification error. This is the effect of the fitness function in the new decomposition strategy, where a higher priority is given to the classification error. Therefore, in MOEA/D-STAT, there is much more pressure on the classification error than in standard MOEA/D.

On the test sets, as can be seen in Tables 6.2 and 6.4, on at most only

two out of the 12 datasets, MOEA/D-STAT is worse than the three benchmark algorithms, except for OMOPSO, which is significantly worse than MOEA/D-STAT on eight datasets. On the five largest datasets, MOEA/D-STAT mostly outperforms the four benchmark algorithms. Fig. 6.7 shows that the median fronts evolved by MOEA/D-STAT are usually more diverse than the ones of dominance-based algorithms. Specifically, on Madelon datasets, the median front of MOEA/D-STAT contains 13 solutions with feature ratios ranging from 0.002 to 0.05. The three dominance-based algorithms have only one or two solutions on their median fronts and most classification errors achieved by the dominance-based algorithms are attained by MOEA/D-STAT.

The experimental results show that using multiple reference points generates better Pareto fronts than using multiple weight vectors. Since the fitness function in the proposed decomposition focuses more on reducing the classification error, its classification performance is significantly better than standard MOEA/D. The new decomposition not only preserves the higher diversity over dominance-based algorithms but also improves the diversity over using multiple weight vectors since it ensures that each sub-problem, defined by a reference point, corresponds to exactly one solution on the true Pareto front.

6.4.3 MOEA/D-DYN vs Others

In Tables 6.1 - 6.4, the second sign in the brackets shows the significant test results, which compares between the four benchmark algorithms with MOEA/D-DYN. On the training set, in terms of the hypervolume indicator, NSGAII, SPEA2, OMOPSO and MOEA/D are significantly worse than MOEA/D-DYN on *all* datasets. MOEA/D-STAT achieves similar hypervolume as MOEA/D-DYN on four datasets while being significantly worse on all other eight datasets. In terms of IGD, on most cases the other algorithms are significantly worse than MOEA/D-DYN, except for

that NSGAII, SPEA2 and MOEA/D-STAT outperform MOEA/D-DYN on only one out of the 12 datasets. Similarly, on the test sets, MOEA/D-DYN is worse than the other algorithms on at most two datasets. The superior to MOEA/D-STAT shows that the dynamic mechanism in MOEA/D-DYN does not affect the algorithm's convergence, which preserves the high performance of the new decomposition using multiple reference points.

Now we will focus more on analyzing the effect of the dynamic mechanism. As shown in Tables 6.1-6.4, MOEA/D-DYN achieves significantly better IGD/hypervolume values than MOEA/D-STAT. The significant improvement is a result of improvement in both classification performance and Pareto-fornt diversity, which can be seen in Fig. 6.6. On the small datasets such as Australian and Vehicle, MOEA/D-DYN's fronts have the same length as MOEA/D-STAT but given the same feature ratio, MOEA/D-DYN's classification error is always lower. On medium datasets, MOEA/D-DYN's fronts become shorter because the solution with the lowest accuracy already dominates all other MOEA/D-STAT's solutions, which select more features. This pattern is clearly shown on Arrhythmia in Fig. 6.6. On the two large datasets, Madelon and Multiple Features, the fronts evolved by MOEA/D-DYN is even much shorter than MOEA/D-STAT's ones. However, shorter fronts do not mean MOEA/D-DYN's solution sets are less diverse than the solutions found by MOEA/D-STAT. Let take the median fronts on Multiple Features as an example. MOEA/D-DYN's median front contains 13 feature subsets, which have feature ratios varying in the range [0.002, 0.042]. Although MOEA/D-STAT's feature ratios have a longer range, [0.004, 0.311], its median front has only 11 feature subsets. Despite selecting more features, MOEA/D-STAT's best feature subset in terms of the classification error is still worse than that of MOEA/D-DYN. It can be seen that the dynamic mechanism does not waste resources (reference points) on non-conflicting regions. It puts more effort on the conflicting regions, which results in more evenly distributed Pareto fronts with better classification performance given the same number of features.

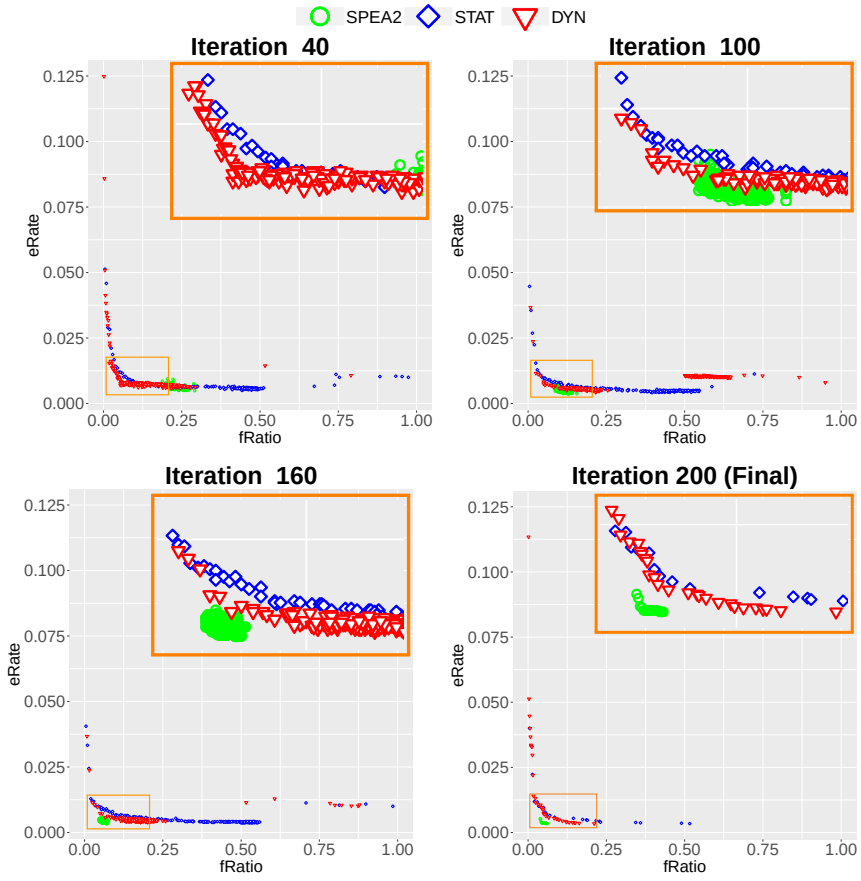


Figure 6.8: Evolutionary processes of the 1^{st} run on MultipleFeatures.

6.4.4 Further Analysis on the Evolutionary Processes

In this section, the search behaviors of different algorithms are examined through their evolutionary processes. SPEA2 is selected as a representative of dominance-based algorithms since it achieves the best performance among the dominance-based algorithms. The largest dataset, MultipleFeatures, is selected to show the differences between the three algorithms clearly.

The three evolutionary processes are shown in Fig. 6.8. There are 4 sub-figures, which corresponds to the populations at the $40^{th}, 100^{th}, 160^{th}$ and 200^{th}, the final iteration, respectively. Note that all algorithms start

from the same initialization, but it is not shown in the figure to save space. Since there is too much overlapping in the center of Pareto fronts, we zoom in these parts and put the zoomed figure on the top right of each sub-figure. It can be seen that, at the 40^{th} iteration, SPEA2 quickly loses its diversity due to the dominance ranking and it mainly searches on a very small area of the objective space. In the following iterations, its diversity is gradually worse, which finally results in a low-diverse Pareto front. On the other hand, two multiple reference point-based MOEA/D algorithms, MOEA/D-STAT and MOEA/D-DYN, maintain their diversities through the whole process. However, their search behaviors are quite different. It can be seen that since MOEA/D-STAT evenly distributed its reference points on the $fRatio$ axis, its population spreads on the whole axis. However, the solutions within the small $fRatio$ (less than 0.5) is denser since some sub-problems with a large n_{ref} values can take subsets with low $fRatio$ as their solutions. For MOEA/D-DYN, it first starts focusing on the first interval which has the feature ratio ranging in [0, 0.25]. Therefore, in the first figure, most solutions are in this range. In the following boundary iterations, MOEA/D-DYN shifts its focus to the next interval. As shown in the second figure, at the 100^{th} iteration, MOEA/D-DYN starts focusing on the third interval by allocating more reference points there. It seems that after a number of iterations, it finds that the third interval is a $threshold$ interval, beyond which the two objectives are possibly no longer conflicting. Therefore, MOEA/D-DYN allocates all moving reference points on the first and second intervals as shown in the fourth figure, without further reallocation. Note that MOEA/D-DYN still leaves some reference points on the other intervals in case the $threshold\ interval$ detection is not accurate. In the final figures, it is shown that the solutions of MOEA/D-DYN on the first range is much more diverse than solutions evolved by MOEA/D-STAT. In addition, MOEA/D-DYN can achieve as good classification accuracies as MOEA/D-STAT but with smaller numbers of features.

6.5 Chapter Summary

In this paper, a new decomposition for MOEA/D is proposed to solve feature selection problems. Instead of using multiple weight vectors, feature selection is decomposed by a set of reference points allocated along the feature ratio axis. The new decomposition is designed to deal with the highly discontinuity in Pareto fronts of feature selection problems. Secondly, a dynamic reference point strategy is proposed to detect and allocate more resource to the conflicting regions. The experimental results show that the two multiple reference point algorithms can evolve more diverse Pareto fronts than the four benchmark algorithms, including NSGAII, SPEA2, OMOPSO and standard MOEA/D. The multiple reference point decomposition also assists MOEA/D to achieve better classification accuracy than using weight vectors since there is more search pressure on improving the classification performance. The dynamic mechanism allows MOEA/D to focus more on the conflicting regions, which results in more diverse Pareto fronts with lower classification errors than the static reference points.

This chapter finds out that decomposition-based EMO, more specifically MOEA/D, can solve multi-objective feature selection more efficiently and effectively than Pareto dominance-based EMO. The strategy of decomposing a multi-objective problem into many scalar sub-problems is particularly suitable for multi-objective feature selection since it can be decomposed along one of feature selection's objectives which is the number of features. Thus a highly discontinuous Pareto front in feature selection becomes an advantage when MOEA/D is used to achieve feature selection. This decomposition strategy can be applied to other multi-objective problems which have highly discontinuous Pareto front like feature selection.

The complicated relationship between objectives in feature selection can also be handled using the proposed decomposition mechanism. Firstly,

the fitness function of each sub-problem gives higher priority to the classification performance. Secondly, the dynamic mechanism can automatically estimate in which regions the objectives are in conflict. Based on that, more resources are allocated to conflicting regions to further improve the quality of obtained feature subsets.

A limitation of this work is that the multiple reference point-based algorithms spend computational time on repairing duplicated feature subsets, which requires to re-estimate the repaired solutions. In the future, we will investigate a more sophisticated evolutionary mechanism to avoid producing duplicated solution leading to better efficiency. It also can be seen that although the multiple reference point decomposition achieves more diverse fronts than dominance-based algorithms such as SPEA2, sometimes solutions evolved by SPEA2 have a higher classification performance. If more search pressure is putting on regions of those solutions, the Pareto fronts of MOEA/D can be further improved. However, these regions depend on datasets and it is not an easy task to identify them.

From Chapter 3 to Chapter 6, we have worked on improving the performance of single-objective and multi-objective feature selection on both evaluation and searching mechanisms, which are two main components of feature selection. In the next chapter, we will work on using feature selection to achieve transfer learning which is one of the challenging tasks in machine learning, which is transfer learning.

Chapter 7

Feature Selection for Transfer Learning

7.1 Introduction

Transfer learning is a challenging task in machine learning, which aims to utilize acquired knowledge from a similar/related available labeled data (source domain) to improve learning performance on the target data (target domain) [36]. If the source and target domains have the same feature space, the transfer learning task can be further specified as *domain adaptation* [36], which is the focus of this chapter.

There are many approaches to achieve domain adaptation (as discussed in section 2.1.2 of Chapter 2); feature-based approaches are one of the most popular approaches. Many recent feature-based domain adaptation methods [248, 249, 61, 193, 250] attempt to build a new common latent feature space onto which both source and target data can be projected, and traditional machine learning can be used to train a classifier to classify the projected target data. However, these methods usually have to assume models to measure differences between data distributions in different domains. Furthermore, the dimensionality of the latent feature space must be pre-defined. Due to creating new high-level features for the latent feature

space, the meaning of the features and the interpretation of the data in the projected space is reduced. Therefore, instead of building a new latent feature space with new high-level features, some other works [59, 60] select a subset of the original features where the differences between source and target data distributions are small. This has an advantage that it preserves the meaning of the original features, instead of creating new features with unknown meanings. Therefore, this chapter focuses on selecting original features that are domain-invariant. This is known as feature selection for domain adaptation.

Although it has been shown that selecting domain-invariant original features across different domains has good results, domain-invariance is not sufficient. It is also important that the selected features have high discriminative ability (relevant features). This has not been paid enough attention in existing feature selection-based domain adaptation approaches. In addition, most existing feature-based domain adaptation approaches make assumptions about *marginal distributions* (related to features) and/or *conditional distributions* (related to the class label) to simplify their models and make them easier to solve using numerical optimization techniques. In this chapter, we propose a new fitness function, which guides PSO to select a good domain-invariant feature subset with an expectation of reducing differences in both marginal and conditional distributions while maintaining good accuracies on the source and target domains. The fitness function is designed with respect to the characteristics of the k-nearest neighbor (KNN) classification algorithm in order to minimize the number of model assumptions and lead to better accuracies on the target domain.

7.1.1 Chapter Goal

The overall goal of this chapter is to develop a new fitness function for PSO to achieve feature selection for domain adaptation. The fitness function aims to select domain-invariant features containing relevant information

about the class label, thereby resulting in high classification accuracies on the target domain. The three main components in the fitness function aim to reduce the classification error on the source domain, the difference of the marginal distributions between the two domains, and the difference of the conditional distributions between the two domains, respectively. Depending on whether there are labeled instances available in the target domain, the fitness function can be flexibly changed from a semi-supervised to an unsupervised form. The proposed fitness function is examined on three real-world benchmark problems and compared with four state-of-the-art traditional domain adaptation algorithms. Specifically, we will investigate:

- whether the proposed fitness function can assist PSO to select a good subset of features, which can achieve better performance than using all features on the target domain,
- whether the proposed PSO-based algorithm with the new fitness function can evolve common feature spaces, which achieve higher classification accuracy than the four traditional feature-based domain adaptation approaches, and
- whether the semi-supervised form can use class information on the target domain to outperform the unsupervised form.

7.2 Proposed Algorithm

In this section, the proposed PSO-based feature selection approach for domain adaptation is described. The main contribution is a new fitness function which allows the PSO-based feature selection algorithm to work in both cases: the class label information on the target domain is available, i.e. semi-supervised, or not available, i.e. unsupervised. To be convenient, the source dataset is named Src, the target un-labeled and labeled datasets are called $TarU$ and $TarL$, respectively. Figure 7.1 gives

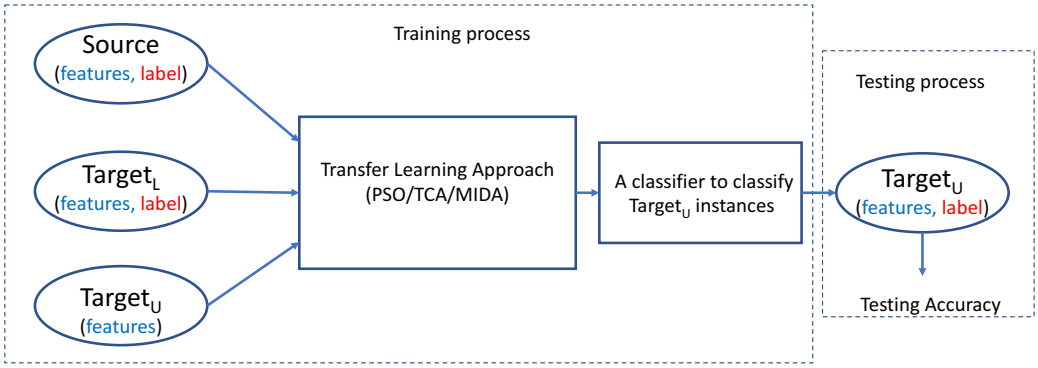


Figure 7.1: An overall view of transfer learning.

an overall view of this work. As can be seen from the figure, a transfer learning approach can take all the source instances (features, label), all the available labeled-target instances (features, label), and feature values from unlabeled-target instances to build a classifier that can classify the unlabeled-target instances. This is very different from traditional machine learning which is not allowed to use the feature values from the unlabeled-target instances.

7.2.1 New Fitness Function

In feature selection for domain adaptation, in order to achieve high classification performance on the target domain, the selected features must satisfy the following requirements: *1*) having a good discriminative ability on both domains, *2*) minimizing the difference between conditional distributions, and *3*) minimizing the difference between marginal distributions on source and target domains. Therefore, the fitness function has three components corresponding to the three above conditions, which can be seen in Eq. (7.1).

$$Fitness = sw \times srcErr + tw \times tarErr + stw \times diffST \tag{7.1}$$

where $srcErr$ and $tarErr$ are classification errors on the source and target data, which ensure the selected features to have a good discriminability

on both domains (requirement 1). Furthermore, $tarErr$ is obtained based on a classifier trained by the source data. Therefore, minimizing $tarErr$ leads to a smaller difference between conditional distributions on the two domains (requirement 2). The last condition is achieved through $diffST$, which measures how different the two marginal distributions are (requirement 3). The three terms will be explained in more details in the following sections. *sw*, *tw*, and *stw* are used to control the contributions of these three components and they sum up to 1, i.e. $sw + tw + stw$=1. The weight values can show the relationship between source and target domains, which will be illustrated in the parameter setting section. The three terms *scrErr*, *tarErr*, and $diffST$ are discussed in the following subsections.

7.2.2 Discriminability on the Source Domain: *srcErr*

In Eq. (7.1), $srcErr$ is to ensure that the selected features have a high discriminative ability in the source domain. $srcErr$ is measured by the classification error rate. Many feature-based domain adaptation algorithms aim to minimize the differences between source and target domains while ignoring this important property in the source domain. The features obtained from these algorithms might not be useful if they cannot preserve the discriminative ability on the source domain and thereby on the target domain, since the two domains become more similar under the selected features.

In order to ensure that $srcErr$ is not biased, $srcErr$ is calculated by applying 3-fold cross-validation on the source dataset. Particularly, each fold plays the role of a test set one time while the other two folds are combined to form a training set. Eq. (7.2) shows how classification error rate is obtained on each fold.

$$ErrorRate = \frac{FP + FN}{TP + TN + FP + FN} \tag{7.2}$$

where TP, TN, FP and FN stand for true positive, true negative, false positive and false negative, respectively. The average value of three error

rates on the three folds is assigned to $srcErr$.

7.2.3 Discriminability on The Target Domain: *tarErr*

In order to achieve a good discriminability on the target domain, we have to consider two possible situations: there are only a small number of labeled instances or there is no labeled instance, which form semi-supervised or unsupervised learning tasks, respectively.

Semi-supervised learning

If there are a small set of labeled instances available in the target domain, called $TarL$, $tarErr$ can be calculated as the classification error rate on $TarL$ with Src being used as the training set. The reason is that if $tarErr$ is low, the two sets $TarL$ and Src are likely to have similar conditional distributions. Since $TarL$ and $TarU$ contain labeled and unlabeled instances drawn from the target domain, they should have the same conditional distribution. Therefore, minimizing $tarErr$ leads to reducing the distribution's difference between Src and $TarU$ as well. The $tarErr$ in the semi-supervised learning is named $tarErr_l$.

Unsupervised Learning

In this case, we do not have labeled instances on the target domain, so the question is how to measure the discriminability without using labels. Based on the idea that the two closest instances usually belong to the same class, we can *estimate* the classification error on the target domain by using labeled source domain instances, given a set of selected features. In particular, suppose that x_{t_1} and x_{t_2} are two closest unlabeled instances on the target domain ($TarU$), they are very likely to have the same class label. Since the class labels of x_{t_1} and x_{t_2} are not available, we can *estimate* their class labels based on their two closest instances from the source domain, x_{s_1} and x_{s_2}. If x_{s_1} and x_{s_2} are in the same class, x_{t_1} and x_{t_2} are also in

Algorithm 4 : Calculate $tarErr_u$

$correct$ =0
for each instance x_{t_i} in $TarU$ **do**
  find its closest instance in the source domain, x_{s_i}
  find its closest instance in the target domain, x_{t_j}
  find the closest instance of x_{t_j} in the source domain, x_{s_j}
  if x_{s_i} and x_{s_j} are in the same class **then**
    $correct$ = $correct$ +1
  end if
end for
$tarErr_u$ = 1 - $\frac{correct}{|TarU|}$

the same class, which means the selected features are good for grouping similar instances into the same class. Otherwise, x_{t_1} and x_{t_2} are in different classes indicating the poor discriminability of the selected features. The $tarErr$ in the unsupervised learning, called $tarErr_u$, is the division between the number of closest target instances being estimated in *different* classes and the total number of instances in $TarU$. Note that since the number of closest instance pairs equal to the number of instances in $TarU$, $tarErr_u$ is in the range [0, 1]. Details on calculating $tarErr_u$ are shown in Algorithm 4.

On both semi-supervised and unsupervised learning cases, the target of $tarErr$ is to ensure that if the selected features have a good discriminative ability on the source domain, they should also have a high discriminability on the target domain. Conceptually, this is similar to the idea of making conditional distributions similar across domains in existing feature-based domain adaptation approaches. However, the use of $tarErr$ does not require any model assumption about the conditional distribution and it works closely with the KNN classification algorithm, which is expected to result in a high classification performance.

7.2.4 Difference Between Marginal Distributions: $diffST$

The last term, $diffST$, in Eq. (7.1) aims to minimize the difference between the marginal distributions. Maximum Mean Discrepancy (MMD) [192] is used to measure the difference between the two marginal distributions, as shown in Eq. (7.3). This metric is widely used in many feature-based approaches.

$$D(Src, TarU) = \left\| \frac{1}{|Src|} \sum_{i=1}^{|Src|} \phi(Src_i) - \frac{1}{|TarU|} \sum_{i=1}^{|TarU|} \phi(TarU_i) \right\|_H \tag{7.3}$$

where $\phi(x) : X \to H$, H is a universal reproducing kernel Hilbert space. $|Src|$ and $|TarU|$ are the number of instances on the source domain and the number of unlabeled instances on the target domain, respectively. Note that in both semi-supervised and unsupervised learning, $TarU$ is always used in the Eq. (7.3) since the final task is to well classify instances from $TarU$. Using the kernel trick, i.e. $k(z_i, z_j^T) = \phi(z_i)\phi(z_j^T)$, where k is a positive definite kernel [61], Eq. (7.3) can be rewritten as:

$$\begin{aligned} D(Src, TarU) = \Big(& \frac{1}{|Src|^2} \sum_{i=1}^{|Src|} \sum_{j=1}^{|Src|} k(Src_i, Src_j) \\ & + \frac{1}{|TarU|^2} \sum_{i=1}^{|TarU|} \sum_{j=1}^{|TarU|} k(TarU_i, TarU_j) \\ & - \frac{2}{|Src||TarU|} \sum_{i=1}^{|Src|} \sum_{j=1}^{|TarU|} k(Src_i, TarU_j) \Big)^{1/2} \end{aligned} \tag{7.4}$$

According to [251], Gaussian Radial Basis Function [RBF, $k(x, y) = exp(||x - y||^2/2\sigma^2)$] is able to detect more types of dependence than linear or polynomial kernels, where σ is the kernel width. Thus RBF is used in this chapter, and its kernel width (σ) is automatically selected for each case based on the "median trick" [252].

7.2.5 Overall Algorithm

In this chapter, the sticky binary PSO proposed in Chapter 4 is used as a searching mechanism to generate candidate feature subsets. Each entry in a particle's position corresponds to one original feature. The entry's value indicates whether the corresponding feature is selected or not. Particularly, the value of 1 means that the feature is selected; while the value of 0 means that the feature is discarded. This chapter is the first attempt to perform feature selection for domain adaptation using PSO to automatically choose a number of features while considering feature interactions and discriminability on both domains.

The overall structure of the proposed system is shown in Fig. 7.2. The main contribution of this chapter is the PSO-based feature selection algorithm, which is marked in blue. In general, source (Src), unlabeled target ($TarU$) and possibly labeled target ($TarL$) data are used to evaluate particles using the proposed fitness function, Eq. (7.1). Based on the final feature subset selected by PSO, both *Src* and *TarU* are projected on the common feature space to form two new data, Src' and $TarU'$, which should share the same data distributions. The KNN classification algorithm (k=1) uses Src' as the training set to classify instances in *TarU'* to obtain the classification performance on the target domain. Depends on whether $TarL$ is available in the target domain, either $tarErr_l$ or $tarErr_u$ is used in Eq. (7.1).

We name the two PSO-based algorithms using $tarErr_l$ and $tarErr_u$ as SemPSO and UnPSO, respectively.

7.3 Experiment Design

The proposed two algorithms are compared with using all features, two well-known unsupervised traditional feature-based domain adaptation algorithms, Transfer Component Analysis (TCA) [61], Maximum Indepen-

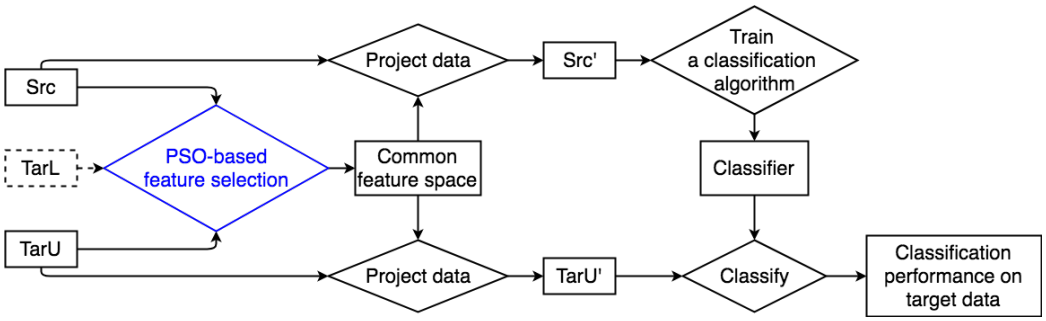


Figure 7.2: PSO-based Feature Selection for Domain Adaptation

dence Domain Adaptation (MIDA) [62], and their extended semi-supervised algorithms, STCA [61], SMIDA [62]. The overall systems of the four benchmark traditional algorithms are similar to the one shown in Fig. 7.2, except for the step of building the common feature space.

7.3.1 Benchmark Datasets

All the algorithms are examined on three well-known real-world problems, Gas Sensor [79], Handwritten Digits [60] and Object Recognition [253, 248], which are shown in Table 7.1, where #C and #F represent the number of classes and features. Each problem contains many cases, which have the same number of classes and features, but might have different numbers of instances in the source (Src) and target domains ($TarU, TarL$).

The gas sensor array drift datasets are collected by Vergara et al. [79] using 16 gas sensors over 36 months. The task is to classify instances into six different kinds of gas. The whole datasets are divided into 10 batches according to the acquisition time. The 1^{st} batch is used as the source dataset and each batch from the 2^{nd} to the 10^{th} ones are used as the target dataset, which forms 9 domain adaptation cases.

USPS and MNIST [60] are two handwritten digit datasets, which share 10 classes of digits. The USPS dataset is collected by scanning envelops from US Postal Service while MNIST is taken from mixed American Census Bureau employees and American high school students, so they have

Table 7.1: Domain adaptation problems.

Problem	Cases	#C	#F	|Src|	|TarU|	|TarL|
Gas Sensor	1-2	6	129	178	746	498
	1-3	6	129	178	951	635
	1-4	6	129	178	97	64
	1-5	6	129	178	118	79
	1-6	6	129	178	1380	920
	1-7	6	129	178	2168	1445
	1-8	6	129	178	176	118
	1-9	6	129	178	282	188
	1-10	6	129	178	2160	1440
Handwritten Digits	MNIST-USPS	10	257	800	1080	720
	USPS-MNIST	10	257	720	1200	800
Object Recognition	A-C	10	801	384	674	449
	A-D	10	801	384	94	63
	A-W	10	801	384	177	118
	C-A	10	801	449	574	384
	C-D	10	801	449	94	63
	C-W	10	801	449	177	118
	D-A	10	801	63	574	384
	D-C	10	801	63	674	449
	D-W	10	801	63	177	118
	W-A	10	801	118	574	384
	W-C	10	801	118	674	449
	W-D	10	801	118	94	63

very different distributions. In this problem, there are two domain adaptation cases, in which either MNIST or USPS is the source dataset and the other one is the target dataset.

The last problem is to recognize 10 objects from four different image sources including Caltech-256 (C) [253], Amazon (A), Webcam (W) and DSLR (D) [248]. To form a domain adaptation case in this problem, we select one image source as the source domain and another image source as the target domain. Therefore, there are 12 cases for the object recognition problem.

In general, there are 23 domain adaptation cases, which have different numbers of classes, features or different numbers of instances in the source and target domains.

7.3.2 Parameter Settings

Parameters of the four traditional algorithms (TCA, STCA, MIDA, and SMIDA), including their kernel and kernel widths, are tuned for the best accuracy using a heuristic search [62] on each case. The parameters of SBPSO are set according to Chapter 4. Each PSO-based algorithm is run 30 independent times on each case.

To tune the three weights in Eq. (7.1), three cases with the lowest classification accuracy from each problem are selected as representatives, which are "1-8", "USPS-MNIST", and "C-W". Different values of the three weights are examined by running SemPSO one time on each selected case. The evolved feature subsets are used to obtain two projected datasets of Src and $TarL$, called Src' and $TarL'$. The best setting for each problem is selected according to the best and the average classification accuracy on $TarL'$. Particularly, on Gas Sensor, sw, tw, and stw are set to 0.1, 0.9 and 0.0, respectively. On Handwritten Digits and Object Recognition, the three values are (0.1, 0.7, 0.2) and (0.0, 0.1, 0.9), correspondingly. Based on the parameters, it can be seen that different problems have different relationships between source and target domains. For example, on Gas Sensor, tw is large which means that the main difference between the two domains is the conditional distribution. Thus tuning the weights may reveal the relationship between domains, which cannot be easily achieved by most traditional approaches. The setting of UnPSO mainly follows SemPSO, except for Object Recognition, where sw is set to 0. Since $TarL$ is not available in UnPSO, $srcErr$ seems to be more reliable than $tarErr$, so sw is set to 0.1 instead of 0.0.

7.4 Results and Discussions

The results on all algorithms are shown in Table 7.2. In the table, #F and Acc represent for the number of selected features and the average classifi-

cation performance. The numbers of features selected by PSO-approaches are used as the pre-defined number of features for four traditional methods to ensure a relatively fair comparison. In the table, the best accuracies are marked in bold while the second best ones are underlined. A significance Wilcoxon test with significance level set to 0.05 is used to compare semi-supervised, unsupervised approaches to examine the effect of the proposed fitness function in both learning cases. Particularly, SemPSO is compared with Full/STCA/SMIDA in Table 7.3(a). Table 7.3(b) shows the comparisons between UnPSO and Full/TCA/MIDA. In each table cell, the three numbers sequentially represent the number of cases that the PSO-based approaches are significantly better, similar or worse than the other benchmark algorithms.

7.4.1 SemPSO/UnPSO vs Using All Features

As can be seen from Table 7.2, in all cases, SemPSO achieves significantly better performance than using all features. For example on Gas Sensor, SemPSO usually improves 20% over the original feature set, especially on the 1-8 case, the accuracy of SemPSO is almost three times better. In the 1-6 case, the accuracy of using all features are already quite high, which means that the two domains are very similar and most features are domain-invariant. However, SemPSO still manages to improve the accuracy by 5%. The possible reason is the classification accuracies in Eq. (7.1) assist PSO to remove irrelevant or redundant features from both domains.

Similar to SemPSO, UnPSO also achieves good performance despite lacking information about the class label in the target domain. Table 7.3(b) shows that UnPSO is worse than the original feature sets on only two cases while being significantly better on 17 out of the 23 cases. Although on 1-10 and D-A, UnPSO's accuracies are at most 0.9% less than using all features, it only selects less than half number of the features. On the other hand, the largest improvement by UnPSO is on W-C, where UnPSO evolves a

Table 7.2: Overall results on 23 domain adaptation cases.

Cases	Full		Unsupervised				Semi-supervised			
			TCA	MIDA	UnPSO		STCA	SMIDA	SemPSO	
	#F	Acc	Acc	Acc	#F	Acc	Acc	Acc	#F	Acc
1-2	128	69.57	63.40	61.80	58.63	76.71	72.39	68.50	49.33	**90.90**
1-3	128	70.24	60.04	63.41	57.00	72.66	72.24	63.83	44.30	**94.58**
1-4	128	61.86	52.58	56.70	62.40	61.55	68.04	58.76	55.23	**81.34**
1-5	128	70.34	49.15	75.42	53.70	71.89	66.95	75.42	56.73	**76.44**
1-6	128	89.64	79.06	80.14	59.00	89.56	84.35	80.58	52.47	**94.53**
1-7	128	53.60	57.84	56.83	55.20	58.09	63.65	55.81	45.73	**71.54**
1-8	128	26.70	12.50	29.55	60.33	32.41	9.09	13.64	41.27	**79.47**
1-9	128	46.45	21.28	18.44	57.93	53.01	29.79	**71.63**	47.67	67.86
1-10	128	49.12	49.54	47.31	58.83	48.19	57.73	45.28	36.77	**68.35**
MNIST-USPS	256	59.63	35.65	34.91	113.33	65.90	55.65	35.93	104.23	**72.54**
USPS-MNIST	256	23.83	20.83	21.17	74.13	49.82	10.67	19.92	97.33	**56.14**
A-C	800	22.55	27.89	28.19	414.83	27.24	19.44	**31.45**	410.20	30.53
A-D	800	18.09	22.34	23.40	409.17	26.74	17.02	26.60	392.30	**29.61**
A-W	800	22.03	26.55	27.12	412.90	28.31	20.90	32.20	401.73	**35.37**
C-A	800	24.39	30.84	30.49	396.23	28.82	24.56	15.33	394.03	**33.05**
C-D	800	22.34	6.38	**26.60**	390.47	25.53	23.40	25.53	383.30	26.06
C-W	800	16.95	20.90	20.34	396.03	25.12	20.34	23.73	389.03	**28.44**
D-A	800	22.65	17.94	17.42	397.63	22.10	11.50	23.34	394.00	**31.40**
D-C	800	23.74	24.78	23.44	379.83	22.38	12.76	26.26	393.80	**29.06**
D-W	800	41.24	10.17	18.08	396.10	47.18	42.94	18.08	405.07	**51.94**
W-A	800	21.95	9.58	24.91	403.67	21.62	10.63	24.22	401.43	**30.21**
W-D	800	44.68	13.83	14.89	407.57	**52.27**	7.45	18.09	403.17	50.32
W-C	800	8.00	14.24	14.09	412.17	21.39	14.39	10.83	391.27	**27.72**

Table 7.3: SemPSO or UnPSO being better/similar/worse using significance tests.

Full	STCA	SMIDA
23/0/0	23/0/0	20/1/2

(a) SemPSO vs semi-supervised methods

Full	TCA	MIDA
17/4/2	18/1/4	17/2/4

(b) UnPSO vs unsupervised methods

set of features which are almost three times more accurate than using all features.

The experimental results show that PSO guided by the proposed fitness function can automatically reduce half of the numbers of features and achieve better classification performance than using all features. The fitness function not only selects domain-invariant features but also extracts relevant ones to improve the classification accuracy.

7.4.2 SemPSO vs STCA/SMIDA

As can be seen in Table 7.3(a), SemPSO well utilizes the class label information on the target domain to significantly improve the classification performance. In all cases, SemPSO significantly outperforms STCA. In comparison with SMIDA, SemPSO is significantly better on 20 out of the 23 cases. On the 1-6 case, where the two domains are very similar, both STCA and SMIDA build new latent feature spaces, which perform worse than the original features. In this case, the important information of original features is discarded by the two traditional methods. On the other hand, SemPSO aims to select relevant original features on both domains, so it can improve the performance over using all features. The two traditional methods aim to maximize the dependence between the features and labels on the target domain. However, the dependency is implicitly optimized through the Hilbert-Schmidt Independence Criterion (HSIC) [254]. Meanwhile, in the proposed fitness function, the classification performance explicitly presents the dependence between the features and class labels, which leads to higher accuracies of the target model. In comparison with UnPSO, except for W-D, SemPSO achieves better classification performance on all other cases. The results show that the actual classification error on *TarL* works better than estimating the classification error on *TarU*.

7.4.3 UnPSO vs TCA/MIDA

The significance test results between UnPSO and TCA/MIDA are shown in Table 7.3(b). As can be seen from the table, UnPSO is similar to the two traditional algorithms only on at most two cases while being significantly better on at least 17 cases. On the Gas Sensor problem, UnPSO usually achieves 10% accuracy better than at least one of the two traditional methods. Especially on the difficult case 1-9, UnPSO is at least three times more accurate than TCA and MIDA. UnPSO also outperforms the two traditional methods on the two handwritten digital cases. Only on the 12 object recognition cases, TCA and MIDA can achieve comparable performance in comparison with UnPSO. UnPSO is similar or better than the other two methods on only nine out of the 12 cases. The possible reason is the estimation process of $tarErr_u$ mainly bases on Euclidean distances, which may not work well on such high-dimensional datasets (800 features).

7.4.4 Overall Comparisons

As can be seen in Table 7.2, in general, SemPSO achieves the best classification accuracy on 19 out of the 23 cases while being ranked as the second-best algorithm on the other four cases. UnPSO evolves the best common feature set on one case and acquires the second best accuracy on nine cases. The achievement of UnPSO is much better than the best traditional algorithm, SMIDA, which obtains the best or second best performance on six cases. Note that in the overall comparisons, UnPSO is also compared with semi-supervised algorithms like SMIDA, which assume some instances are labeled in the target domain. Therefore, the outperformance of UnPSO to other semi-supervised traditional algorithms suggests that the estimation of target classification error based on the source domain can improve the discriminative ability on the target domain and reduce the differences between their conditional distributions, which are

assumed to be the same in the four traditional approaches. The classification error rate on the source domain also plays an important role in the fitness function. Normally, traditional feature-based adaptation approaches focus only on producing domain-invariant features, while the source classification performance ensures that the selected features have high discriminative abilities on both domains when they become similar.

7.5 Chapter Summary

In this chapter, a novel fitness function is developed to assist PSO to automatically select a subset of original features, which can improve classification performance in domain adaptation problems. The proposed fitness function aims to select relevant and domain-invariant features across different domains. The fitness function can flexibly adapt with unsupervised or semi-supervised domain adaptation, depends on the availability of labels on the target domain. Based on that, two PSO-based feature selection algorithms for domain adaptation are proposed and examined on three well-known real-world problems containing 23 domain adaptation cases, in total. The proposed PSO-based algorithms, called UnPSO and SemPSO, are compared with using all features and four traditional feature-based domain adaptation algorithms, TCA, STCA, MIDA, and SMIDA. The experimental results show that both UnPSO and SemPSO outperform using all features and the other algorithms on almost all datasets. Although UnPSO does not use any labeled instances in the target domain, it still outperforms semi-supervised benchmark algorithms, such as STCA and SMIDA, on many domain adaptation cases.

The contribution of this chapter is to design a new fitness function for PSO to achieve domain adaptation by feature selection. This is the first work applying an EC algorithm, more specifically PSO, to achieve feature-based domain adaptation. In the proposed fitness function, the classification performance is used to explicitly consider discriminating abilities

on both domains, feature interactions and minimize model assumptions about differences between source and target domains. By doing that, PSO can select better feature subsets than relying on model assumptions which may ignore the feature interactions. Furthermore, the proposed fitness function can flexibly adapt to the availability of labeled instances in the target domain, which is usually more difficult to achieve by the traditional feature-based domain adaptation.

In the future, we will further investigate their potential to achieve even better performance. For example, the difference between two marginal distributions is calculated based on the MMD model, which may not perform well on the high-dimensional dataset. We will work on estimating the difference without any model assumption. The three weights in the fitness function can be adaptively changed, by analyzing the relationships between source and target domains. In addition, in case of unsupervised learning, distance-based measures are used to estimate class labels for unlabeled instances. It is known that distance-based measures may not scale well with respect to the number of features, so new measures need to be investigated.

Chapter 8

Conclusions

This thesis has focused on developing EC-based feature selection methods in classification. The overall goal was to investigate and improve the capability of EC for feature selection to reduce the number of features while maintaining or even improving the classification performance compared with using all the original features. To achieve the above goal, a number of EC-based feature selection approaches have been proposed to automatically select small feature subsets with similar or better classification performances than using all features. The proposed methods focus on different aspects of feature selection such as the number of objectives (single-objective/multi-objective), the fitness function (filter/wrapper), and the searching mechanism. Experiments have been conducted to examine and compare the proposed methods with existing methods on a range of real-world datasets of varying difficulty. The results show that the proposed algorithms can enhance the capability of EC-based feature selection to effectively select small subsets of informative features for classification.

The remainder of this chapter presents conclusions for each individual objective of this thesis, summarizes the main findings from each chapter, and then suggests several potential research directions for future work.

8.1 Achieved Objectives and Main Conclusions

This thesis has demonstrated that EC, more specifically PSO and MOEA/D, can effectively and efficiently address feature selection in single and multi-objective ways, respectively. The EC-based feature selection approach can also be applied to achieve transfer learning, one of the most challenging tasks in machine learning.

In this section, each of the five achieved objectives is summarized, followed by the main conclusions drawn from each objective.

8.1.1 PSO and Mutual Information Estimation for Feature Selection

This first objective was to design a new fitness function in PSO for single objective filter-based feature selection (Chapter 3). In the new fitness function, mutual information is used to measure the relevance and redundancy of a candidate feature subset, and is calculated using an estimation method to avoid limitations of the traditional counting approach. Experimental results show that the new fitness function can assist PSO to reduce the number of features while achieving comparative classification performance in comparison with using all features. More importantly, the estimation approach results in better feature subsets than the traditional counting approach on both continuous and discrete datasets. This approach is the first PSO and mutual information estimation-based feature selection approach.

Feature interactions

Mutual information is a good measure to detect interactions between features. This thesis demonstrates that the estimation approach can calculate the mutual information for detecting feature interactions better than the counting approach. Given a set of features, both approaches try to build

probability distributions on the feature values by examining the instances. However, the counting approach simply counts instances with each possible value of the features to derive probability distributions for the features. For numeric features, this requires a large number of instances to calculate mutual information accurately. On the other hand, the estimation approach is based on distances between the feature values in the instances to derive the probability distribution of the feature, which exploits additional information about ordering in continuous and ordered discrete datasets. Therefore, the estimation based feature selection algorithms can detect redundant and irrelevant features which cannot be detected using the counting approach.

In addition, the estimation approach can work directly on both continuous and ordered discrete datasets. In contrast, on continuous datasets, the counting approach requires a discretization process which loses important information from the original data. A recommendation of this thesis is to use the estimation approach on these kinds of datasets, where feasible.

Computational time

In terms of the computational cost, the estimation approach is more expensive than the counting approach. The main reason is that the counting approach goes through all instances to count; so its complexity is $O(N)$, given N is the number of instances. The estimation approach has to calculate between instances; in general its complexity is at least $O(N^2)$ with a higher constant factor than the counting approach. Therefore, the estimation approach may be infeasible on large datasets.

8.1.2 Sticky Binary PSO

This thesis introduces a novel binary PSO (BPSO) algorithm which can effectively address binary problems, such as feature selection (Chapter 4). In the new BPSO algorithm, the velocity is redefined as the flipping prob-

ability and the momentum is redefined as the tendency to stick with the current position. The two newly defined concepts help to describe movements of particles in binary search spaces more accurately. Based on that, a dynamic mechanism is proposed to better balance exploration and exploitation during the evolutionary process of BPSO. Experiments on two well-known binary problems (knapsack and feature selection) show that the new BPSO algorithm evolves similar or better solutions than other well-known binary EC algorithms on almost all datasets. In addition, the new BPSO algorithm is more efficient than the other algorithms. The dynamic mechanism improves the performance of the new BPSO algorithm since it helps to better control the trade-off between exploration and exploitation.

This thesis shows that in order to achieve good performances on binary search spaces, BPSO has to be designed to cope with characteristics of these search spaces. In addition, given a limited computational resource to solve a problem with a large and complex search space, the trade-off between exploration and exploitation needs to be considered carefully to acquire good solutions.

Movements of particles in binary search spaces

This thesis shows that the performance of BPSO can be significantly improved if the movements of particles in a binary search space are described more accurately. The main reason is that in a binary search space, particles move by flipping their position's entries, which has no direction in contrast to movements in a continuous search space. The limitations of existing BPSO algorithms are a result of inappropriately applying the velocity and momentum concepts from continuous PSO to BPSO. Therefore, redefining velocity as a flipping probability and momentum as a stickiness property is suitable for binary search spaces, which improves the performance of BPSO.

Exploration and exploitation in a large and complex search space

This thesis demonstrates that controlling the trade-off between exploration and exploitation can significantly influence the performance of binary PSO on binary problems, specifically feature selection.

The thesis confirms that it is better to focus more on exploration at the beginning, and then gradually shift the focus to exploitation at the end, which is generally accepted. The main reason is exploring more promising regions first avoids local optima, then the exploitation can focus on the promising regions. A common way is to do *all* exploration first, then *all* exploitation. This thesis shows that this is not always the best way.

When the search space is large (due mainly to a large number of decision variables/features) and complex (due mainly to complex interactions between features), it is better to take an alternating approach which alternates between exploration and exploitation than the sequential approach, performing all the exploration followed by all the exploitation. The exploration steps aim to discover promising regions in the search space; the exploitation steps aim to find the best point within a promising region. In the sequential approach, PSO may forget some discovered promising regions which might contain an optimal solution. The alternating approach can partially avoid that by exploiting discovered promising regions as they are found. Furthermore, even if PSO does not lose any promising regions, in the exploitation step, PSO usually splits its computational resources evenly on different promising regions. In the sequential approach, the exploration may result in a large number of promising regions, so there will be only a little computational resource allocated to each region. This may lead to these regions not being well exploited, which does not result in good solutions. The alternating approach exploits the promising regions as they are found, so for each iteration, the number of promising regions exploited by the alternating approach is usually smaller than that of the sequential approach. Therefore, the alternating approach puts more computational resources in each region, which might result in better solu-

tions. Most importantly, the exploitation process usually results in better quality solutions than the exploration process. Therefore, interleaving of exploration and exploitation usually leads to better solutions during the evolutionary process, which is more likely to result in better final solutions than the sequential approach, given the same computational cost.

By combining a mechanism to alternate between exploration and exploitation with a dynamic mechanism to increase the amount of exploitation towards the end, the proposed BPSO algorithm can evolve significantly better feature subsets, especially on datasets with large numbers of features.

8.1.3 Surrogate Models for Wrapper-based Feature Selection

This thesis presented a surrogate model in PSO for single objective wrapper-based feature selection (Chapter 5). The main goal was to reduce the computational cost of wrapper-based feature selection approaches while maintaining or even improving the classification performance. In the surrogate model, a surrogate training set is built by selecting a small number of informative instances from the original training set. The surrogate training assists PSO to efficiently locate promising regions in the search space, then the located regions are further explored using the original training set. A local search was also proposed to improve the quality of current $gbest$, based on the surrogate training set and information obtained from $gbest$ in previous iterations. Furthermore, a dynamic surrogate model was proposed to automatically select a suitable surrogate training set during the PSO evolutionary process. The experimental results show that the new surrogate models help PSO to select smaller subsets of features with similar or better classification accuracies than using the original training set. More importantly, using the surrogate training set can significantly reduce the computational cost of PSO-based wrapper-based feature selection al-

gorithms.

Surrogate training set

This thesis demonstrates that using the surrogate training set can partially avoid overfitting and reduce the number of selected features. However, it is important to select informative instances and remove noisy instances from the original training set to form the surrogate training set. The main purpose is to maintain important information which is the class boundaries in this case. Removing noisy instances helps the search to select important features and avoid features which capture characteristics of noisy instances. It is also important to select enough instances to avoid underfitting. Furthermore, the consistency between the surrogate training set and the original training set has to be maintained during the evolutionary process in order to make the surrogate fitness landscape similar to the original fitness landscape. This ensures that the promising regions obtained using the surrogate fitness function are worth exploring using the original fitness function.

Local search in a surrogate model

Although a surrogate model using a surrogate training set can significantly reduce the computational cost, it is still an approximation approach and its performance is usually similar to the original model (without using the surrogate model). It was found that a local search based on the surrogate training set can improve the performance of using the surrogate model without increasing the computational cost too much. The local search improves the quality of current *gbest* using features selected by *gbest* in previous iterations. In PSO, $gbest$ captures essential historical information about the evolutionary search, but the local search approach shows that there is an advantage in keeping more historical information about the evolutionary search, because this additional historical information can

improve the performance of PSO for feature selection.

8.1.4 Multi-objective Wrapper-based Feature Selection

This thesis presents a multi-objective wrapper-based feature selection approach using MOEA/D (Chapter 6). The new approach decomposes multi-objective feature selection into several single-objective sub-problems by using a set of multiple reference points instead of a set of weight vectors as in standard MOEA/D. Each reference point corresponds to a number of features and the task of the corresponding sub-problem is to find the best feature subset containing at most the number of features. The reference points can be allocated statically with fixed locations or dynamically with their locations being updated during the evolutionary process. The dynamic allocation aims to deal with the partial conflict between the two objectives of feature selection. Experimental results show that both versions of the new algorithm can evolve more diverse Pareto fronts than three well-known Pareto dominance-based algorithms and standard MOEA/D. The dynamic allocation successfully detects conflicting regions, which results in more diverse Pareto fronts with better classification performance than the static allocation.

Highly discontinuous Pareto fronts

This thesis shows how to use a new decomposition in order to apply MOEA/D to a problem with a highly discontinuous Pareto front. Handling Pareto fronts with complex shapes is a challenging task for standard MOEA/D since its performance heavily depends on its set of weight vectors which in turn depends on the shape of the Pareto front. The decomposition based on multiple reference points allocated on the feature ratio (i.e. number of selected features over the total number of features) axis makes MOEA/D able to cope with the discontinuity of Pareto fronts in multi-objective feature selection. This decomposition is expected to be effective

in solving other multi-objective problems which have highly discontinuous Pareto fronts like feature selection.

Unequally important objectives

This thesis demonstrates that MOEA/D is a good choice for feature selection. Although Pareto dominance-based EMO algorithms can cope with highly discontinuous Pareto front, they usually assume that the two objectives have the same importance. In feature selection, the classification performance usually has a higher priority than the number of features. Since each sub-problem in MOEA/D is usually a single objective problem, this characteristic can be embedded in the fitness function for each sub-problem. The same would be true for any other multi-objective where the objectives do not have equal importance.

Objectives that only partially conflict

Multi-objective approaches usually assume that the objectives are in conflict. In multi-objective feature selection, the two objectives are not always in conflict. The regions where the objectives are not conflicting have only one best solution, but the conflicting regions may have multiple non-dominated solutions. Therefore the conflicting regions should be focused on more than the non-conflicting regions. Chapter 6 presented the first work using MOEA/D that identifies conflicting regions in feature selection. By allocating more reference points, it can allocate more computational resources to conflicting regions, which results in a diverse set of non-dominated feature subsets with better classification performance.

8.1.5 Feature Selection for Transfer Learning

This thesis introduces a PSO-based feature selection algorithm for feature-based transfer learning. The new PSO-based algorithm has a novel fitness

function which explicitly uses the classification performance to reduce assumptions required to model the differences between the source and target domains while still selecting relevant and domain-invariant features across different domains. The new fitness function is effective whether or not class labels are available in the target domain. The new algorithm can successfully evolve small feature subsets which achieve better classification performance on the target domain than using all features. Given the same number of features, the new algorithm can evolve feature subsets with lower classification error than four well-known feature-based transfer learning algorithms.

This thesis demonstrates that PSO-based feature selection can be successfully applied to achieve transfer learning, which has not been done before. In feature-based transfer learning, the task is to select features that are not only relevant but also domain-invariant, which minimize the differences between data distributions (marginal and conditional distributions) on different domains. Therefore, the fitness function for feature selection-based transfer learning is more complex than the fitness function for feature selection in classification. It is shown that instead of using any model assumptions, classification accuracy can be used in the fitness function to simultaneously reduce the difference between conditional distributions and maintain discriminating ability on target domains. The new fitness function can take advantages of class labels in the target domain but does not require them, which is an advantage over traditional feature-based transfer learning.

8.2 Future Work

This section provides some possible directions for future work.

8.2.1 Mutual Information Estimation-based Feature Selection

It has been shown that mutual information estimation works well on feature selection. However, its computational cost is high. One cause of the high cost is a large number of instances. Therefore, one way to address the problem would be to develop an instance selection algorithm for mutual information estimation to reduce the number of instances. However, removing too many instances may affect the performance of mutual information estimation and therefore the ability to detect relevant and redundant features. A second cause for the high cost is that it is time-consuming to calculate pair-wise distances between instances and sort them. Therefore, another way to address the problem would be to develop a new mutual information estimator that is not based on distances. An additional reason for this approach is that it is known that distance measures may not work well on datasets with high dimensionality.

8.2.2 Combining Feature Construction and Feature Selection

Feature construction and feature selection are the two main approaches for feature reduction. However, most existing feature reduction work focuses on either feature selection or feature construction. Tran et al. [255] showed that the combination of new high-level features and the original features appearing in the high-level features achieves a better classification performance than using only selected features or only constructed features. This suggests that it is promising to do feature construction and feature selection together so that we can take the advantages of both methods. This was partially done by Tran et al. [255] but only the constructed features were evolved during the evolutionary process while the selected features were chosen at the end, which meant that the interactions between constructed and selected features were ignored. It would be interesting to

develop a new representation and updating mechanisms to combine feature construction and feature selection. We conducted some initial work on this problem, which used a simple combination of vector-based feature selection and tree-based feature construction within a single evolutionary process (mixing GA-like operators and GP-like operators) [256]. Future work could explore how to take into account interactions between the two components.

8.2.3 MOEA/D for Feature Selection

This thesis proposed the first multi-objective wrapper-based feature selection approach using MOEA/D. It has been shown that MOEA/D can evolve a more diverse set of non-dominated feature subsets than Pareto dominance-based algorithms. However, given the same number of features, the Pareto dominance-based algorithms still achieve better classification performance. In addition, the computational cost of the proposed algorithm is still high due to the repairing process which requires additional evaluations. Therefore, future work could investigate more sophisticated updating mechanisms which produce feasible feature subsets and put more emphasis on improving the classification performance.

8.2.4 Feature-based Transfer Learning

This thesis proposed the first work applying PSO-based feature selection to achieve transfer learning. However, this was a very initial work and more works need to be done in future to investigate more deeply the ability of EC-based feature reduction to achieve transfer learning. One direction would be to apply GP to build new latent feature spaces between different domains. This could be further extended by building a new feature space containing a subset of the original features and new high-level features.

Bibliography

[1] R. Bellman, *Dynamic programming*. Courier Corporation, 2013.

[2] E. Keogh and A. Mueen, *Curse of Dimensionality*, pp. 314–315. Boston, MA: Springer US, 2017.

[3] H. Zhao, A. P. Sinha, and W. Ge, "Effects of feature construction on classification performance: An empirical study in bank failure prediction," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2633–2644, 2009.

[4] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys (CSUR)*, vol. 50, no. 6, p. 94, 2017.

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

[6] A. A. Albrecht, "Stochastic local search for the feature set problem, with applications to microarray data," *Applied Mathematics and Computation*, vol. 183, no. 2, pp. 1148–1164, 2006.

[7] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2011.

[8] K. Neshatian and M. Zhang, "Dimensionality reduction in face detection: A genetic programming approach," in *International Confer-*

ence on Image and Vision Computing New Zealand, pp. 391–396, IEEE, 2009.

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

[10] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[11] W. Dai, Q. Yang, G.-R. Xue, and Y. Yu, "Boosting for transfer learning," in *Proceedings of the 24th International Conference on Machine learning*, pp. 193–200, ACM, 2007.

[12] J. Jiang and C. Zhai, "Instance weighting for domain adaptation in nlp," in *ACL*, vol. 7, pp. 264–271, 2007.

[13] E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *Advances in Neural Information Processing Systems*, pp. 153–160, 2007.

[14] A. Schwaighofer, V. Tresp, and K. Yu, "Learning gaussian process kernels via hierarchical bayes," in *Advances in Neural Information Processing Systems*, pp. 1209–1216, 2004.

[15] I.-H. Jhuo, D. Liu, D. Lee, S.-F. Chang, *et al.*, "Robust visual domain adaptation with low-rank reconstruction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2168–2175, 2012.

[16] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, and M. Salzmann, "Unsupervised domain adaptation by domain invariant projection,"

in *IEEE International Conference on Computer Vision (ICCV)*, pp. 769–776, 2013.

[17] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars, "Unsupervised visual domain adaptation using subspace alignment," in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2960–2967, 2013.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[19] J. Ahmad, F. Javed, and M. Hayat, "Intelligent computational model for classification of sub-golgi protein using oversampling and fisher feature selection methods," *Artificial intelligence in Medicine*, vol. 78, pp. 14–22, 2017.

[20] M. Dash, H. Liu, and H. Motoda, "Consistency based feature selection," in *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pp. 98–109, Springer, 2000.

[21] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.

[22] A. Senawi, H.-L. Wei, and S. A. Billings, "A new maximum relevance-minimum multicollinearity (mrmmc) method for feature selection and ranking," *Pattern Recognition*, vol. 67, pp. 47–61, 2017.

[23] I. Kononenko, "On biases in estimating multi-valued attributes," in *IJCAI*, vol. 95, pp. 1034–1040, Citeseer, 1995.

[24] F. Li, D. Miao, and W. Pedrycz, "Granular multi-label feature selection based on mutual information," *Pattern Recognition*, vol. 67, pp. 410–423, 2017.

[25] W. Gao, L. Hu, and P. Zhang, "Class-specific mutual information variation for feature selection," *Pattern Recognition*, vol. 79, pp. 328–339, 2018.

[26] C. Granger and J.-L. Lin, "Using the mutual information coefficient to identify lags in nonlinear models," *Journal of time series analysis*, vol. 15, no. 4, pp. 371–384, 1994.

[27] G. A. Darbellay, "An estimator of the mutual information based on a criterion for conditional independence," *Computational Statistics & Data Analysis*, vol. 32, no. 1, pp. 1–17, 1999.

[28] C. Freeman, D. Kulić, and O. Basir, "An evaluation of classifier-specific filter measure performance for feature selection," *Pattern Recognition*, vol. 48, no. 5, pp. 1812–1826, 2015.

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

[30] J. Kennedy, R. Eberhart, *et al.*, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, 1995.

[31] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[32] M. Lichman, "UCI machine learning repository," 2013.

[33] Y. Kodratoff, *Introduction to machine learning*. Morgan Kaufmann, 2014.

[34] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine learning: An Artificial Intelligence Approach*. Springer Science & Business Media, 2013.

[35] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[36] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[37] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of Electronic Imaging*, vol. 16, no. 4, p. 049901, 2007.

[38] L. Olshen, C. J. Stone, *et al.*, "Classification and regression trees," *Wadsworth International Group*, vol. 93, no. 99, p. 101, 1984.

[39] B. Zheng, S. W. Yoon, and S. S. Lam, "Breast cancer diagnosis based on feature extraction using a hybrid of k-means and support vector machine algorithms," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1476–1482, 2014.

[40] J.-H. Kim, "Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap," *Computational Statistics & Data Analysis*, vol. 53, no. 11, pp. 3735–3745, 2009.

[41] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, "Prediction error estimation: a comparison of resampling methods," *Bioinformatics*, vol. 21, no. 15, pp. 3301–3307, 2005.

[42] T. Dietterich, "Overfitting and undercomputing in machine learning," *ACM Computing Surveys (CSUR)*, vol. 27, no. 3, pp. 326–327, 1995.

[43] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[44] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[45] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[46] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[47] G. V. Kass, "An exploratory technique for investigating large quantities of categorical data," *Applied Statistics*, pp. 119–127, 1980.

[48] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[49] B. Schölkopf and C. J. Burges, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[50] M. A. Hearst, S. T. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[51] J. Valyon and G. Horváth, "A weighted generalized ls-SVM," *Periodica Polytechnica Electrical Engineering*, vol. 47, no. 3-4, pp. 229–252, 2003.

[52] P. Langley, W. Iba, and K. Thompson, "An analysis of bayesian classifiers," in *AAAI*, vol. 90, pp. 223–228, 1992.

[53] J. Lu, V. Behbood, P. Hao, H. Zuo, S. Xue, and G. Zhang, "Transfer learning using computational intelligence: A survey," *Knowledge-Based Systems*, vol. 80, pp. 14–23, 2015.

[54] D. Pardoe and P. Stone, "Boosting for regression transfer," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 863–870, Omnipress, 2010.

[55] W. Pan, H. Zhong, C. Xu, and Z. Ming, "Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks," *Knowledge-Based Systems*, vol. 73, pp. 173–180, 2015.

[56] G.-R. Xue, W. Dai, Q. Yang, and Y. Yu, "Topic-bridged PLSA for cross-domain text classification," in *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 627–634, 2008.

[57] J. Gao, W. Fan, J. Jiang, and J. Han, "Knowledge transfer via multiple model local structure mapping," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 283–291, 2008.

[58] T. Evgeniou and M. Pontil, "Regularized multi–task learning," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 109–117, 2004.

[59] S. Uguroglu and J. Carbonell, "Feature selection for transfer learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 430–442, Springer, 2011.

[60] J. Tahmoresnezhad and S. Hashemi, "An efficient yet effective random partitioning and feature weighting approach for transfer learning," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 30, no. 02, p. 1651003, 2016.

[61] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[62] K. Yan, L. Kou, and D. Zhang, "Learning domain-invariant subspace using domain features and independence maximization," *IEEE Transactions on Cybernetics*, vol. 48, no. 1, pp. 288–299, 2018.

[63] D. Koller and M. Sahami, "Toward optimal feature selection," Technical Report 1996-77, Stanford InfoLab, February 1996.

[64] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature subset selection," *IEEE Transactions on Computers*, vol. 100, no. 9, pp. 917–922, 1977.

[65] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, vol. 2, pp. 129–134, 1992.

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

[67] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[68] 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.

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

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

[71] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.

[72] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 13, no. 5, pp. 971–989, 2016.

[73] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 02, pp. 185–205, 2005.

[74] G. H. John, R. Kohavi, K. Pfleger, *et al.*, "Irrelevant features and the subset selection problem," in *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 121–129, 1994.

[75] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[76] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*, pp. 199–213, Springer, 1998.

[77] Y.-J. Hu, "Constructive induction: covering attribute spectrum," in *Feature Extraction, Construction and Selection*, pp. 257–272, Springer, 1998.

[78] U. Kamath, K. De Jong, and A. Shehu, "Effective automated feature construction and selection for classification of biological sequences," *PloS one*, vol. 9, no. 7, p. e99982, 2014.

[79] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[80] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.

[81] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.

[82] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial intelligence through simulated evolution," 1966.

[83] J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.

[84] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1470–1477, 1999.

[85] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces [r]," *Berkeley: ICSI*, 1995.

[86] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann, *et al.*, "What is a learning classifier system?," in *International Workshop on Learning Classifier Systems*, pp. 3–32, Springer, 1999.

[87] J. E. Hunt and D. E. Cooke, "Learning using an artificial immune system," *Journal of Network and Computer Applications*, vol. 19, no. 2, pp. 189–212, 1996.

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

[89] K. Sarath and V. Ravi, "Association rule mining using binary particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1832–1840, 2013.

[90] M. A. Taha and D. I. A. al Nadi, "Spectrum sensing for cognitive radio using binary particle swarm optimization," *Wireless Personal Communications*, vol. 72, no. 4, pp. 2143–2153, 2013.

[91] J. C.-W. Lin, L. Yang, P. Fournier-Viger, T.-P. Hong, and M. Voznak, "A binary PSO approach to mine high-utility itemsets," *Soft Computing*, pp. 1–19, 2016.

[92] S. Mirjalili and A. Lewis, "S-shaped versus V-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1–14, 2013.

[93] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Systems*, vol. 64, pp. 22–31, 2014.

[94] J. Yang, H. Zhang, Y. Ling, C. Pan, and W. Sun, "Task allocation for wireless sensor network using modified binary particle swarm optimization," *IEEE Sensors Journal*, vol. 14, no. 3, pp. 882–892, 2014.

[95] A. H. El-Maleh, A. T. Sheikh, and S. M. Sait, "Binary particle swarm optimization (BPSO) based state assignment for area minimization of sequential circuits," *Applied soft computing*, vol. 13, no. 12, pp. 4832–4840, 2013.

[96] T. Zhai and Z. He, "Instance selection for time series classification based on immune binary particle swarm optimization," *Knowledge-Based Systems*, vol. 49, pp. 106–115, 2013.

[97] J. Liu, Y. Mei, and X. Li, "An analysis of the inertia weight parameter for binary particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 666–681, 2016.

[98] B. Tan, H. Huang, H. Ma, and M. Zhang, *Binary PSO for Web Service Location-Allocation*, pp. 366–377. Cham: Springer International Publishing, 2017.

[99] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm Intelligence*, pp. 43–85, Springer, 2008.

[100] C. A. C. Coello, "Evolutionary multiobjective optimization," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 5, pp. 444–447, 2011.

[101] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," *Lecture notes in computer science*, vol. 1917, pp. 849–858, 2000.

[102] E. Zitzler, M. Laumanns, L. Thiele, *et al.*, "SPEA2: Improving the strength pareto evolutionary algorithm," in *Eurogen*, vol. 3242, pp. 95–100, 2001.

[103] M. Reyes-Sierra, C. C. Coello, *et al.*, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[104] M. R. Sierra and C. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and e-dominance," in *Evolutionary Multi-criterion Optimization*, vol. 3410, pp. 505–519, Springer, 2005.

[105] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 440–462, 2017.

[106] H. Ishibuchi, Y. Setoguchi, H. Masuda, and Y. Nojima, "Performance of decomposition-based many-objective algorithms strongly depends on pareto front shapes," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 169–190, 2017.

[107] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, 2014.

[108] C. Zhang, K. C. Tan, L. H. Lee, and L. Gao, "Adjust weight vectors in MOEA/D for bi-objective optimization problems with discontinuous pareto fronts," *Soft Computing*, pp. 1–16, 2017.

[109] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: Optimal-distributions and the choice of the reference point," in *Proceedings of the ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, pp. 87–102, 2009.

[110] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[111] E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.

[112] L. Alfonso, A. Lobbrecht, and R. Price, "Optimization of water level monitoring network in polder systems using information theory," *Water Resources Research*, vol. 46, no. 12, 2010.

[113] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[114] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, pp. 1065–1076, 1962.

[115] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical review E*, vol. 69, no. 6, p. 066138, 2004.

[116] J. T. Lizier, "JIDT: An information-theoretic toolkit for studying the dynamics of complex systems," *arXiv preprint arXiv:1408.3270*, 2014.

[117] S. D. Stearns, "On selecting features for pattern classifiers.," in *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976)*, (Coronado, CA), pp. 71–75, 1976.

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

[119] S. Nakariyakul and D. P. Casasent, "An improvement on floating search algorithms for feature subset selection ," *Pattern Recognition*, vol. 42, no. 9, pp. 1932–1940, 2009.

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

[121] J. Lee and D.-W. Kim, "Feature selection for multi-label classification using multivariate mutual information," *Pattern Recognition Letters*, vol. 34, no. 3, pp. 349–357, 2013.

[122] J. Lee and D.-W. Kim, "Mutual information-based multi-label feature selection using interaction information," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2013–2025, 2015.

[123] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," in *Proceedings of The International Conference on Machine learning*, p. 78, ACM, 2004.

[124] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2, 1-norms minimization," in *Advances in Neural Information Processing Systems*, pp. 1813–1821, 2010.

[125] N. Kwak, "Principal component analysis based on l1-norm maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 9, pp. 1672–1680, 2008.

[126] J. Tang, S. Alelyani, and H. Liu, "Feature selection for classification: A review," *Data Classification: Algorithms and Applications*, p. 37, 2014.

[127] J. Gui, Z. Sun, S. Ji, D. Tao, and T. Tan, "Feature selection based on structured sparsity: A comprehensive study," *IEEE Transactions on*

Neural Networks and Learning Systems, vol. 28, no. 7, pp. 1490–1507, 2017.

[128] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Applied Soft Computing*, vol. 18, pp. 261–276, 2014.

[129] B. Xue, M. Zhang, and W. N. Browne, "New fitness functions in binary particle swarm optimisation for feature selection," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2012.

[130] B. Xue, M. Zhang, and W. N. Browne, "Novel initialisation and updating mechanisms in pso for feature selection in classification," in *Applications of Evolutionary Computation*, (Berlin, Heidelberg), pp. 428–438, Springer Berlin Heidelberg, 2013.

[131] S. M. Vieira, L. F. Mendonça, G. J. Farinha, and J. M. Sousa, "Modified binary PSO for feature selection using svm applied to mortality prediction of septic patients," *Applied Soft Computing*, vol. 13, no. 8, pp. 3494–3504, 2013.

[132] 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.

[133] S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon, "Modified binary particle swarm optimization," *Progress in Natural Science*, vol. 18, no. 9, pp. 1161–1166, 2008.

[134] C.-L. Huang and C.-J. Wang, "A GA-based feature selection and parameters optimization for support vector machines," *Expert Systems with applications*, vol. 31, no. 2, pp. 231–240, 2006.

[135] S.-W. Lin, K.-C. Ying, S.-C. Chen, and Z.-J. Lee, "Particle swarm optimization for parameter determination and feature selection of sup-

port vector machines," *Expert Systems with Applications*, vol. 35, no. 4, pp. 1817–1824, 2008.

[136] A. Boubezoul and S. Paris, "Application of global optimization methods to model and feature selection," *Pattern Recognition*, vol. 45, no. 10, pp. 3676–3686, 2012.

[137] M. C. Lane, B. Xue, I. Liu, and M. Zhang, "Particle swarm optimisation and statistical clustering for feature selection," in *AI 2013: Advances in Artificial Intelligence*, pp. 214–220, Springer, 2013.

[138] M. C. Lane, B. Xue, I. Liu, and M. Zhang, "Gaussian based particle swarm optimisation and statistical clustering for feature selection," in *Evolutionary Computation in Combinatorial Optimisation*, pp. 133–144, Springer, 2014.

[139] H. B. Nguyen, B. Xue, I. Liu, and M. Zhang, "PSO and statistical clustering for feature selection: a new representation," in *Simulated Evolution and Learning*, pp. 569–581, Springer, 2014.

[140] H. B. Nguyen, B. Xue, I. Liu, P. Andreae, and M. Zhang, "Gaussian transformation based representation in particle swarm optimisation for feature selection," in *Applications of Evolutionary Computation*, pp. 541–553, Springer, 2015.

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

[142] B. Tran, B. Xue, and M. Zhang, "Improved PSO for feature selection on high-dimensional datasets," in *Simulated Evolution and Learning*, pp. 503–515, Springer, 2014.

[143] K. Mistry, L. Zhang, S. C. Neoh, C. P. Lim, and B. Fielding, "A micro-GA embedded PSO feature selection approach to intelligent facial

emotion recognition," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1496–1509, 2017.

[144] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015.

[145] S. Gu, R. Cheng, and Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Computing*, vol. 22, no. 3, pp. 811–822, 2018.

[146] F. Wang and J. Liang, "An efficient feature selection algorithm for hybrid data," *Neurocomputing*, vol. 193, pp. 33–41, 2016.

[147] 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.

[148] L. Cervante, B. Xue, L. Shang, and M. Zhang, "Binary particle swarm optimisation and rough set theory for dimension reduction in classification," in *IEEE Congress on Evolutionary Computation*, pp. 2428–2435, 2013.

[149] C. Bae, W.-C. Yeh, Y. Y. Chung, and S.-L. Liu, "Feature selection with intelligent dynamic swarm and rough set," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7026–7032, 2010.

[150] L. Cervante, B. Xue, L. Shang, and M. Zhang, "A dimension reduction approach to classification based on particle swarm optimisation and rough set theory.," in *Australasian Conference on Artificial Intelligence*, pp. 313–325, Springer, 2012.

[151] B. Chakraborty and G. Chakraborty, "Fuzzy consistency measure with particle swarm optimization for feature selection," in *IEEE*

International Conference on Systems, Man, and Cybernetics, pp. 4311–4315, IEEE, 2013.

[152] L. Cervante, B. Xue, M. Zhang, and L. Shang, "Binary particle swarm optimisation for feature selection: A filter based approach," in *IEEE Congress on Evolutionary Computation*, IEEE, 2012.

[153] H. Nguyen, B. Xue, I. Liu, and M. Zhang, "Filter based backward elimination in wrapper based PSO for feature selection in classification," in *IEEE Congress on Evolutionary Computation*, pp. 3111–3118, 2014.

[154] I. Jain, V. K. Jain, and R. Jain, "Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification," *Applied Soft Computing*, vol. 62, pp. 203 – 215, 2018.

[155] L.-Y. Chuang, C.-S. Yang, K.-C. Wu, and C.-H. Yang, "Gene selection and classification using taguchi chaotic binary particle swarm optimization," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13367–13377, 2011.

[156] B. Xue, M. Zhang, and W. N. Browne, "Multi-objective particle swarm optimisation (PSO) for feature selection," in *Proceedings of the 14th Annual conference on Genetic and Evolutionary Computation*, pp. 81–88, ACM, 2012.

[157] Y. Li, S. Zhang, and X. Zeng, "Research of multi-population agent genetic algorithm for feature selection," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11570–11581, 2009.

[158] J. Derrac, S. García, and F. Herrera, "A first study on the use of coevolutionary algorithms for instance and feature selection," in *Hybrid Artificial Intelligence Systems*, pp. 557–564, Springer, 2009.

[159] S. Oreski and G. Oreski, "Genetic algorithm-based heuristic for feature selection in credit risk assessment," *Expert systems with applications*, vol. 41, no. 4, pp. 2052–2064, 2014.

[160] L. Wang, "A hybrid genetic algorithm–neural network strategy for simulation optimization," *Applied Mathematics and Computation*, vol. 170, no. 2, pp. 1329–1343, 2005.

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

[162] P. N. da Silva, A. Plastino, and A. A. Freitas, "A novel genetic algorithm for feature selection in hierarchical feature spaces," in *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 738–746, SIAM, 2018.

[163] A. A. Chaaraoui and F. Flórez-Revuelta, "Human action recognition optimization based on evolutionary feature subset selection," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, pp. 1229–1236, ACM, 2013.

[164] J.-H. Seo, Y. H. Lee, and Y.-H. Kim, "Feature selection for very short-term heavy rainfall prediction using evolutionary computation," *Advances in Meteorology*, vol. 2014, 2014.

[165] D. Liang, C.-F. Tsai, and H.-T. Wu, "The effect of feature selection on financial distress prediction," *Knowledge-Based Systems*, vol. 73, pp. 289–297, 2015.

[166] D. Paul, R. Su, M. Romain, V. Sébastien, V. Pierre, and G. Isabelle, "Feature selection for outcome prediction in oesophageal cancer using genetic algorithm and random forest classifier," *Computerized Medical Imaging and Graphics*, vol. 60, pp. 42–49, 2017.

[167] S. Jiang, K.-S. Chin, L. Wang, G. Qu, and K. L. Tsui, "Modified genetic algorithm-based feature selection combined with pre-trained deep neural network for demand forecasting in outpatient department," *Expert Systems with Applications*, vol. 82, pp. 216–230, 2017.

[168] A. Mukhopadhyay and U. Maulik, "An SVM-wrapped multiobjective evolutionary feature selection approach for identifying cancer-microRNA markers," *IEEE Transactions on Nanobioscience*, vol. 12, no. 4, pp. 275–281, 2013.

[169] C. J. Tan, C. P. Lim, and Y.-N. Cheah, "A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models," *Neurocomputing*, vol. 125, pp. 217–228, 2014.

[170] U. Singh and S. N. Singh, "Optimal feature selection via NSGA-II for power quality disturbances classification," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2017.

[171] Y. Zhu, J. Liang, J. Chen, and Z. Ming, "An improved NSGA-III algorithm for feature selection used in intrusion detection," *Knowledge-Based Systems*, vol. 116, pp. 74 – 85, 2017.

[172] R. Hunt, K. Neshatian, and M. Zhang, "A genetic programming approach to hyper-heuristic feature selection," in *Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 320–330, Springer, 2012.

[173] U. Bhowan and D. McCloskey, "Genetic programming for feature selection and question-answer ranking in IBM Watson," in *Genetic Programming*, pp. 153–166, Springer, 2015.

[174] D. Y. Harvey and M. D. Todd, "Automated feature design for numeric sequence classification by genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 474–489, 2015.

[175] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalisation of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[176] F. Viegas, L. Rocha, M. GonÃğalves, F. MourÃčo, G. SÃą, T. Salles, G. Andrade, and I. Sandin, "A genetic programming approach for feature selection in highly dimensional skewed data," *Neurocomputing*, vol. 273, pp. 554 – 569, 2018.

[177] R. N. Khushaba, A. Al-Ani, A. AlSukker, and A. Al-Jumaily, "A combined ant colony and differential evolution feature selection algorithm," in *Ant Colony Optimization and Swarm Intelligence*, pp. 1–12, Springer, 2008.

[178] R. N. Khushaba, A. Al-Ani, and A. Al-Jumaily, "Feature subset selection using differential evolution and a statistical repair mechanism," *Expert Systems with Applications*, vol. 38, no. 9, pp. 11515–11526, 2011.

[179] B. Xue, W. Fu, and M. Zhang, "Multi-objective feature selection in classification: A differential evolution approach," in *Simulated Evolution and Learning*, vol. 8886 of *Lecture Notes in Computer Science*, pp. 516–528, Springer International Publishing, 2014.

[180] E. Hancer, B. Xue, and M. Zhang, "Differential evolution for filter feature selection based on information theory and feature ranking," *Knowledge-Based Systems*, vol. 140, pp. 103 – 119, 2018.

[181] A. Al-Dujaili, M. R. Tanweer, and S. Suresh, "DE vs. PSO: A performance assessment for expensive problems," in *IEEE Symposium Series on Computational Intelligence*, pp. 1711–1718, 2015.

[182] L. D. Vignolo, D. H. Milone, and J. Scharcanski, "Feature selection for face recognition based on multi-objective evolutionary wrap-

pers," *Expert Systems with Applications*, vol. 40, no. 13, pp. 5077 – 5084, 2013.

[183] B. Huang, B. Buckley, and T.-M. Kechadi, "Multi-objective feature selection by using NSGA-II for customer churn prediction in telecommunications," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3638–3646, 2010.

[184] C. J. Tan, C. P. Lim, and Y. Cheah, "A multi-objective evolutionary algorithm-based ensemble optimizer for feature selection and classification with neural network models," *Neurocomputing*, vol. 125, pp. 217 – 228, 2014.

[185] E. De la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps," *Knowledge-Based Systems*, vol. 71, pp. 322–338, 2014.

[186] B. Xue, M. Zhang, and W. N. Browne, "Multi-objective particle swarm optimisation (PSO) for feature selection," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pp. 81–88, 2012.

[187] 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.

[188] H. B. Nguyen, B. Xue, I. Liu, P. Andreae, and M. Zhang, "New mechanism for archive maintenance in PSO-based multi-objective feature selection," *Soft Computing*, vol. 20, no. 10, pp. 3927–3946, 2016.

[189] B. Xue, W. Fu, and M. Zhang, "Multi-objective feature selection in classification: a differential evolution approach," in *Proceedings of*

the 10th International Conference on Simulated Evolution and Learning, pp. 516–528, Springer, 2014.

[190] S. Paul and S. Das, "Simultaneous feature selection and weighting–an evolutionary multi-objective optimization approach," *Pattern Recognition Letters*, vol. 65, pp. 51–59, 2015.

[191] S. J. Pan, J. T. Kwok, and Q. Yang, "Transfer learning via dimensionality reduction.," in *AAAI*, vol. 8, pp. 677–682, 2008.

[192] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, "Integrating structured biological data by kernel maximum mean discrepancy," *Bioinformatics*, vol. 22, no. 14, pp. e49–e57, 2006.

[193] Y. Shi and F. Sha, "Information-theoretical learning of discriminative clusters for unsupervised domain adaptation," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 1275–1282, 2012.

[194] Z. Cui, W. Li, D. Xu, S. Shan, X. Chen, and X. Li, "Flowing on riemannian manifold: Domain adaptation by shifting covariance," *IEEE Transactions on Cybernetics*, vol. 44, no. 12, pp. 2264–2273, 2014.

[195] J. Walters-Williams and Y. Li, "Estimation of mutual information: A survey," in *Rough Sets and Knowledge Technology*, pp. 389–396, Springer, 2009.

[196] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[197] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[198] M. Lungarella, T. Pegors, D. Bulwinkle, and O. Sporns, "Methods for quantifying the informational structure of sensory and motor data," *Neuroinformatics*, vol. 3, no. 3, pp. 243–262, 2005.

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

[200] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

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

[202] F. van den Bergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937 – 971, 2006.

[203] M. A. Khanesar, M. Teshnehlab, and M. A. Shoorehdeli, "A novel binary particle swarm optimization," in *Mediterranean Conference on Control Automation(MED).*, pp. 1–6, 2007.

[204] A. P. Engelbrecht and G. Pampara, "Binary differential evolution strategies," in *IEEE Congress on Evolutionary Computation*, pp. 1942–1947, 2007.

[205] H. bin Ouyang, L. qun Gao, S. Li, and X. yong Kong, "Improved global-best-guided particle swarm optimization with learning operation for global optimization problems," *Applied Soft Computing*, vol. 52, pp. 987 – 1008, 2017.

[206] M. Chih, C.-J. Lin, M.-S. Chern, and T.-Y. Ou, "Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem," *Applied Mathematical Modelling*, vol. 38, no. 4, pp. 1338 – 1350, 2014.

[207] L. F. Mingo López, N. Gómez Blas, and A. Arteta Albert, "Multidimensional knapsack problem optimization using a binary particle swarm model with genetic operations," *Soft Computing*, pp. 1–16, 2017.

[208] B. Haddar, M. Khemakhem, S. Hanafi, and C. Wilbaut, "A hybrid quantum particle swarm optimization for the multidimensional problem," *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 1 – 13, 2016.

[209] M. F. Tasgetiren, Q. K. Pan, D. Kizilay, and G. Suer, "A differential evolution algorithm with variable neighborhood search for multidimensional knapsack problem," in *IEEE Congress on Evolutionary Computation*, pp. 2797–2804, 2015.

[210] D. Libao, W. Sha, J. Chengyu, and H. Cong, "A hybrid mutation scheme-based discrete differential evolution algorithm for multidimensional knapsack problem," in *Sixth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*, pp. 1009–1014, 2016.

[211] A. Rezoug, D. Boughaci, and M. Badr-El-Den, *Memetic Algorithm for Solving the 0-1 Multidimensional Knapsack Problem*, pp. 298–304. Cham: Springer International Publishing, 2015.

[212] J. P. Martins, H. Longo, and A. C. Delbem, "On the effectiveness of genetic algorithms for the multidimensional knapsack problem," in *Proceedings of the Companion Publication of the Annual Conference on Genetic and Evolutionary Computation*, GECCO Comp '14, pp. 73–74, 2014.

[213] H. Ishibuchi, N. Akedo, and Y. Nojima, "Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems,"

IEEE Transactions on Evolutionary Computation, vol. 19, no. 2, pp. 264–283, 2015.

[214] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[215] F. Glover and G. A. Kochenberger, "Critical event tabu search for multidimensional knapsack problems," in *Meta-Heuristics*, pp. 407–427, Springer, 1996.

[216] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1362–1381, 2009.

[217] F. Nie, H. Huang, X. Cai, and C. H. Ding, "Efficient and robust feature selection via joint l2,1-norms minimization," in *Advances in Neural Information Processing Systems 23*, pp. 1813–1821, Curran Associates, Inc., 2010.

[218] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[219] F. Murtagh and P. Legendre, "Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion?," *Journal of Classification*, vol. 31, no. 3, pp. 274–295, 2014.

[220] D. R. Wilson and T. R. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 257–286, 2000.

[221] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[222] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61 – 70, 2011.

[223] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function-based multiobjective evolutionary algorithm," in *Evolutionary Multi-criterion Optimization*, pp. 438–452, Springer, 2009.

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

[225] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics. the knapsack problem case study," *European Journal of Operational Research*, vol. 158, no. 2, pp. 418–433, 2004.

[226] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 682–691, 2012.

[227] C. P. Almeida, R. A. Gonçalves, E. F. Goldbarg, M. C. Goldbarg, and M. R. Delgado, "An experimental analysis of evolutionary heuristics for the biobjective traveling purchaser problem," *Annals of Operations Research*, vol. 199, no. 1, pp. 305–341, 2012.

[228] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[229] M. Asafuddoula, T. Ray, and R. Sarker, "A decomposition-based evolutionary algorithm for many objective optimization," *IEEE*

Transactions on Evolutionary Computation, vol. 19, no. 3, pp. 445–460, 2015.

[230] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[231] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[232] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang, "Stable matching-based selection in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 6, pp. 909–923, 2014.

[233] H.-L. Liu, F. Gu, and Q. Zhang, "Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 450–455, 2014.

[234] F. Gu and Y.-M. Cheung, "Self-organizing map-based weight design for decomposition-based many-objective evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, 2017, DOI: 10.1109/TEVC.2017.2695579.

[235] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, 2016.

[236] H. Ishibuchi, M. Yamane, and Y. Nojima, "Difficulty in evolutionary multiobjective optimization of discrete objective functions with

different granularities," in *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 230–245, Springer, 2013.

[237] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints.," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[238] S. Jiang and S. Yang, "A strength pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, 2017.

[239] Q. Zhang, W. Liu, and H. Li, "The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances," in *IEEE Congress on Evolutionary Computation*, pp. 203–208, 2009.

[240] A. Zhou and Q. Zhang, "Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 52–64, 2016.

[241] Y. Yuan, H. Xu, and B. Wang, "An improved NSGA-III procedure for evolutionary many-objective optimization," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pp. 661–668, 2014.

[242] Q. Zhang, H. Li, D. Maringer, and E. Tsang, "MOEA/D with NBI-style tchebycheff approach for portfolio management," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[243] I. Giagkiozis, R. C. Purshouse, and P. J. Fleming, "Towards understanding the cost of adaptation in decomposition-based optimization algorithms," in *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 615–620, 2013.

[244] K. Miettinen, *Nonlinear multiobjective optimization*, vol. 12. Springer Science and Business Media, 2012.

[245] H. B. Nguyen, B. Xue, H. Ishibuchi, P. Andreae, and M. Zhang, "Multiple reference points MOEA/D for feature selection," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pp. 157–158, 2017.

[246] J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *TIK-Report*, vol. 214, 2006.

[247] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the jMetal multi-objective optimization framework," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, pp. 1093–1100, 2015.

[248] R. Gopalan, R. Li, and R. Chellappa, "Domain adaptation for object recognition: An unsupervised approach," in *IEEE International Conference on Computer Vision*, pp. 999–1006, 2011.

[249] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073, 2012.

[250] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, 2014.

[251] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization," *Journal of Machine Learning Research*, vol. 13, no. May, pp. 1393–1434, 2012.

[252] L. Song, B. Boots, S. M. Siddiqi, G. J. Gordon, and A. Smola, "Hilbert space embeddings of hidden markov models," 2010.

[253] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[254] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *International Conference on Algorithmic Learning Theory*, pp. 63–77, Springer, 2005.

[255] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, no. 1, pp. 3–15, 2015.

[256] H. B. Nguyen, B. Xue, and P. Andreae, "A hybrid GA-GP method for feature reduction in classification," in *Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 591–604, Springer, 2017.