Feature Grouping-based Feature Selection

Ling Zheng

Supervisors: Prof. Qiang Shen
Dr. Neil S. Mac Parthaláin

Ph.D. Thesis

Department of Computer Science
Institute of Mathematics, Physics and Computer Science
Aberystwyth University

March 23, 2017

Declaration and Statement

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. Where **correction services**[1] have been used, the extent and nature of the correction is clearly marked in a footnote(s).

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

[1] This refers to the extent to which the text has been corrected by others.

Abstract

Feature selection (FS) is a process which aims to select input domain features that are most informative for a given outcome. Unlike other dimensionality reduction techniques, feature selection methods preserve the underlying semantics or meaning of the original data following reduction. Typically, FS can be divided into four categories: filter, wrapper, hybrid-based and embedded approaches. Many strategies have been proposed for this task in an effort to identify more compact and better quality feature subsets. As various advanced techniques have emerged in the development of search mechanisms, it has become increasingly possible for quality feature subsets to be discovered efficiently without resorting to exhaustive search.

Harmony search is a music-inspired stochastic search method. This general technique can be used to support FS in conjunction with many available feature subset quality evaluation methods. The structural simplicity of this technique means that it is capable of reducing the overall complexity of the subset search. The naturally stochastic properties of this technique also help to reduce local optima for any resultant feature subset, whilst locating multiple, potential candidates for the final subset. However, it is not sufficiently flexible in adjusting the size of the parametric musician population, which directly affects the performance on feature subset size reduction. This weakness can be alleviated to a certain extent by an iterative refinement extension, but the fundamental issue remains. Stochastic mechanisms have not been explored to their maximum potential by the original work, as it does not employ a parameter of pitch adjustment rate due to its ineffective mapping of concepts.

To address the above problems, this thesis proposes a series of extensions. Firstly, a self-adjusting approach is proposed for the task of FS which involves a mechanism to further improve the performance of the existing harmony search-based method. This approach introduces three novel techniques: a restricted feature domain created for each individual musician contributing to the harmony improvisation in order to improve harmony diversity; a harmony memory consolidation which explores the possibility of exchanging/communicating information amongst musicians such that it can dynamically adjust the population of musicians in improvising new harmonies; and a pitch adjustment which exploits feature similarity measures to identify neighbouring features in order to fine-tune the newly discovered harmonies.

These novel developments are also supplemented by a further new proposal involving the application to a feature grouping-based approach proposed herein for FS, which works by searching for feature subsets across homogeneous feature groups rather than examining a massive number of possible combinations of features. This approach radically departs from the traditional FS techniques that work by incrementally adding/removing features from a candidate feature subset one feature at a time or randomly selecting feature combinations without considering the relationship(s) between features. As such, information such as inter-feature correlation may be retained and the residual redundancy in the returned feature subset minimised. Two different instantiations of an FS mechanism are derived from such a feature grouping-based framework: one based upon the straightforward ranking of features within the resultant feature grouping; and the other on the simplification for harmony search-based FS.

Feature grouping-based FS offers a self-adjusting approach to effectively and efficiently addressing many real-world problems which may have data dimensionality concerns and which requires semantic-preserving in data reduction. This thesis investigate the application of this approach in the area of intrusion detection, which must deal in a timely fashion with huge quantities of data extracted from network traffic or audit trails. This approach empirically demonstrates the efficacy of feature grouping-based FS in action.

Acknowledgements

I would like to express my uttermost gratitude to my supervisors: Prof. Qiang Shen and Dr. Neil S. Mac Parthaláin, for playing great mentors throughout my whole PhD research life with their motivation, enthusiasm, guidance, and positive harshness.

My sincere gratitude goes to my entire family: my parents Yunguan Zheng and Lezhen Lin, my filial sister Qingping Zheng, my sweet wife Binbin Zheng, and her parents Yuming Zheng and Weihui Ke. The completion of this Ph.D. would not have been possible without their kind support and encouragement.

I am grateful to Dr. Diao Ren for his original inspirations to this research.

I am thankful to Dr. Richard Jensen for his help in research resource and documents.

I would like to thank all my fellow researchers ever in the Advanced Reasoning Group, in particular, Dr. Longzhi Yang, Dr. Chengyuan Chen, Dr. Shangzhu Jin, Dr. Pan su, Dr. Nitin Kumar Naik, Dr. Yongfeng Zhang, and PhD candidats including Zhenpeng Li, and Tianhua Chen for the stimulating discussions, insight, and helpful advice.

I would like to express my deepest appreciation to the Department of Computer Science at Aberystwyth University for their generous financial support.

My sincere gratitude goes to the anonymous reviewers, journal editors, conference organisers and attendees involved (either directly or indirectly) with my submitted works, for their encouragement and valuable input in refining my ideas.

I am extremely grateful to all of the academic, administrative, technical, and support staff at the Department of Computer Science, Aberystwyth University, for their kind assistance at all stages of my PhD research.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

DATA is a magic word, which can become any form of information existing everywhere in our life. This, in conjunction with the great progress in the development of computer hardware technology (e.g., in particular, data collection equipment and storage media) leads to the wide availability of huge amounts of data. Unfortunately, information analysis of such a big "data bomb" still remains challenging. As a result, only a small fraction of data is used to any advantage. Therefore, means of automation, efficiency, and scalability are increasingly required to enable humans to extract valuable information from these fast expanding mountains of data.

Knowledge discovery from data (KDD) [172] is a data analysis scheme for discovering useful knowledge that is humanly comprehensible from naturally meaningless data. It has certain alternative names, including: pattern discovery/analysis [40], information harvesting [151], knowledge extraction [218], data mining [95], data archaeology [23], and data dredging [29]. Particularly, data mining also acts as an important sub-field in KDD. A systemic knowledge discovery involves an iterative sequence of steps as characterised in Fig. 1.1:

1. Data screening: This is the process of inspecting the data for errors, and involves techniques such as checking raw data, identifying outliers and dealing with missing data.

2. Data cleansing: The main task of data cleansing is to correct or remove data in datasets that is incorrect, incomplete, improperly formatted, or duplicated.

3. Data reduction: This step is focused on techniques which aim to remove redundant, irrelevant, or misleading domain features. This allows data mining methods to perform more efficiently by rendering their data inputs in simpler, more compact form.

4. Data Mining: Data mining is the most important technique in KDD; it attempts to uncover the truly interesting data patterns that are hidden in large datasets.

5. Interpretation: This final step is to interpret these extracted patterns into useful knowledge through techniques such as modelling.

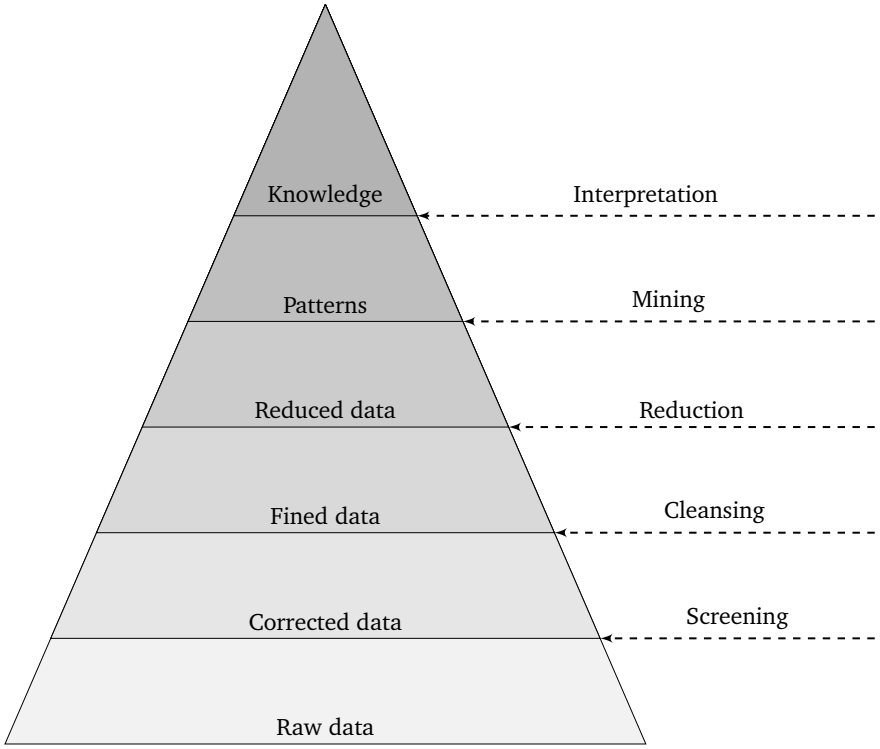

Figure 1.1: The knowledge discovery process

Steps 1-4 present essential techniques, albeit different for data preprocessing, where the data are prepared for mining. Particularly, techniques required in step 3 are exclusively used for solving problems which satisfy high dimensionality. These techniques can be divided into two categories: those that transform the underlying meaning of the data features and those that are semantics-preserving. Feature selection (FS) methods are involved in the latter category, in which subsets of the original features are selected in order to maximise a suitable evaluation function

in regards to information quality. As a fundamental step of knowledge discovery, FS not only helps data mining by improving the performance in terms of efficiency, but also preserves the human interpretability of the mined useful knowledge. The development of better FS methods is therefore the main subject of this thesis.

1.1 Feature Selection

Feature selection (FS) [139] is becoming an increasingly necessary step as the issue of dataset dimensionality becomes ever more pervasive for complex, real-world problems. Traditionally employed in areas such as data mining, pattern recognition, and machine learning, FS is now seeing widespread use [184]. This is because complex problems often contain large numbers of features, which may result in considerable computational overhead for data-driven knowledge discovery and decision-making tasks [232]. In particular, certain features may be irrelevant or redundant and offer no contribution when building robust predictive computational models. Some may involve a significant amount of noise or even be misleading, thereby adversely affecting the accuracy of a given model [109].

Combating the naïve assumption of "more features = more knowledge", which is the source of the high dimensionality problem, FS works by finding a minimal feature subset while preserving the underlying semantics of the data. It can be used to remove irrelevant, redundant, or noisy features. The advantages of FS techniques lie not only in indirectly alleviating the computational overhead for subsequent learning mechanisms, but also in offering more compact knowledge representation and a reduction in data storage requirements [91]. Reflecting these advantages, FS has become a popular technique for assisting tasks such as text processing, data classification and system control [143, 188, 189] while there are a wide range of real-world applications of FS [116, 139, 142] in prominent fields as illustrated in Fig. 1.2.

Broadly speaking, FS approaches in terms of evaluation methods can be divided into three different categories (or variants thereof): wrapper, filter, and hybrid methods [91]. Wrapper methods [92, 120, 121] are often used in conjunction with inductive learning algorithms (e.g., C4.5 [176]), where the classification accuracy of the learning mechanism is used as a metric of feature subset quality. Since classifiers typically require a re-training phase for newly added data, the computational overhead for large data can often be prohibitive for such methods. Filter methods on the

other hand, are simple and employ a predefined subset evaluation metric rather than a classifier learner to estimate the quality of any candidate feature subset. The use of evaluation metrics that are easy to compute makes feature filters computationally efficient, which can be computed independently of the subsequent learning task that exploits the selected feature subset. Nevertheless, high quality values for feature subset candidates obtained using filter evaluation metrics do not necessarily translate into good classification accuracies or indeed robust models, when the same features are used to train and test classifier learners. In an attempt to address both of the above mentioned problems, so-called 'hybrid' methods [244] have been proposed where elements of both wrapper and filter methods are integrated. As a further development, embedded FS methods [91], which perform feature selection using the objective function (e.g., least squares) of a learning algorithm to estimate the quality of the candidate subset without the need for re-training, have also been proposed.

1.2 Feature Grouping-based Feature Selection

Although existing work on FS has resulted in many powerful techniques, for most approaches important information with regard to the level of feature correlation may be ignored during the selection process. Methods that only iteratively include/exclude single individual features from a candidate subset of features are typical examples where this is the case. This loss of feature correlation may lead to a considerable level of redundancy in the resulting dataset [170]. FS algorithms based upon a high-level clustering framework can address this issue by grouping those redundant features together and then selecting the representative features from each group, in order to form a final set of selected features.

Initial clustering-related FS methods have been reported in the literature [100, 106, 190, 199]. The algorithmic framework for these methods primarily involves two steps: 1) identifying homogeneous groups of features using clustering techniques; and 2) performing subset selection on the resulting feature groups. Traditional clustering methods (e.g., k-means, c-means, hierarchical, and graph-theoretic clustering [104]) are used to obtain object clusters, where the similarity between objects are measured by distance functions or metrics (e.g., Euclidean distance). However, information behind features can be measured by information metrics such as mutual

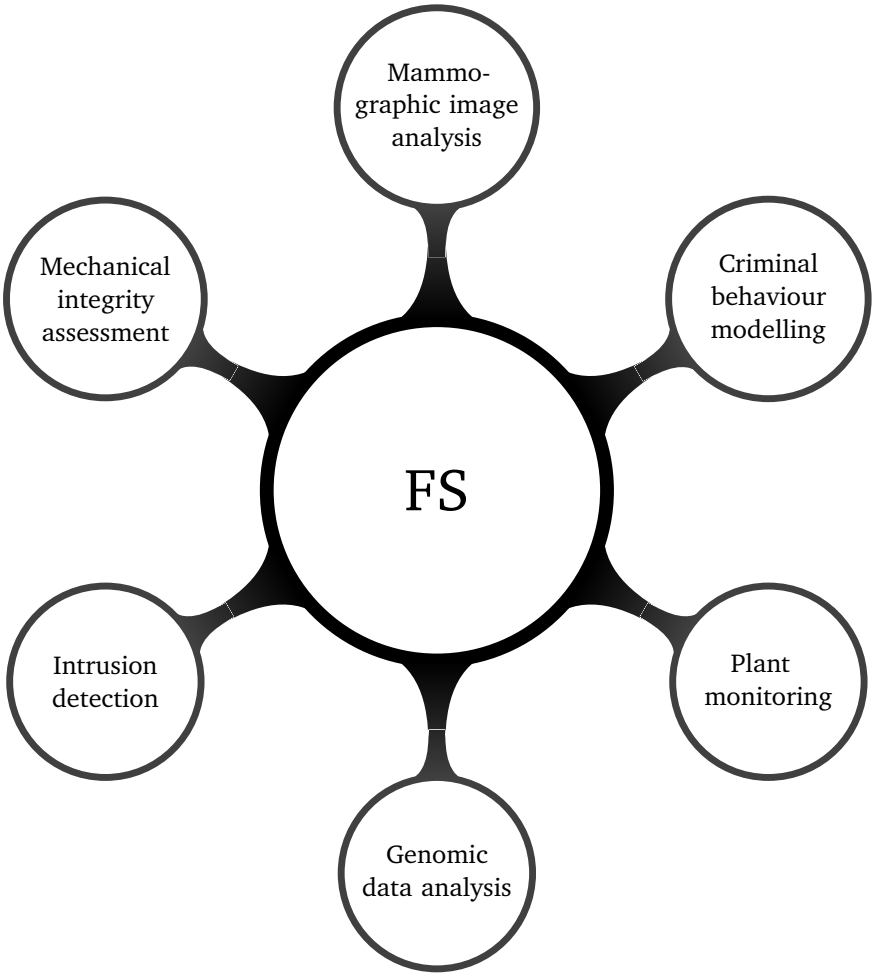

Figure 1.2: Real-world applications of FS

information [65], correlation coefficient [93], and fuzzy-rough set dependency [177]. This leads to the ability to cluster redundant features.

1.3 Feature Selection with Nature-Inspired Techniques

Leaving aside the learning mechanism, the search for the optimal feature subset is in fact a combinatorially hard problem [76]. An exhaustive search could be used in order to guarantee a global optimum, but this would also lead to an exponential increase in computational time-complexity. This means that exhaustive methods are often computationally intractable for feature subset search where the data is large. One of the most common approaches to addressing this drawback is to employ

greedy hill-climbing strategies, where single features (or groups of features) which result in the greatest increase in the quality of a candidate feature subset (i.e., the emerging subset of selected features) are greedily added to the candidate subset. However, many of these approaches, although efficient, can easily become trapped in local optima. Alternatives that employ metaheuristics may help to escape the local regions and return, or at least come close to returning, the global optimum. Examples of these include genetic algorithms (GA) [7], memetic algorithms (MA) [126], particle swarm optimisation (PSO) [219], harmony search (HS) [81], and other nature-inspired techniques.

Harmony search (HS) [80] is a recently developed meta-heuristic optimisation algorithm mimicking a musical improvisation phenomenon, during which each musician in an ensemble plays a note in order to discover a best overall harmony. HS has been very successful in addressing various engineering optimisation problems [4, 43, 69, 131, 202, 215, 237] and machine learning tasks [44, 146]. Several advantages over traditional optimisation techniques have also been demonstrated [80, 229].

HS imposes only limited mathematical requirements and is not sensitive to the initial parameter value settings. As a population-based approach, HS works by generating a new vector that encodes a candidate solution after considering the quality of existing tentative solutions. This is in contrast to the classical genetic algorithms that typically consider only two (parent) vectors in order to produce a new (child) vector. Due to its popularity, the original HS technique has been improved by methods that dynamically adjust its parameters [80, 145], making the algorithm more adaptive to the variance in variable value ranges. Work has also been carried out to analyse the evolution of the population variance over successive generations in HS, thereby drawing important conclusions regarding its exploratory power [43]. More variants have been developed in the literature (e.g., [78, 162, 229]) which attempt to improve its overall search capability.

An application of HS to FS (HSFS) has also been recently developed [53], which has demonstrated competitive FS outcomes. However, the original HSFS is too restrictive when adjusting the size of the parametric musician/variable population which directly affects the performance of the feature subset size reduction. This weakness is alleviated to a certain extent by an iterative refinement extension, but the fundamental issue remains. Stochastic mechanisms have not been explored to

their maximum potential by the original work, as it does not employ the parameter of pitch adjustment rate due to its ineffective mapping of concepts.

To address the original shortcomings of HSFS identified above, two different directional approaches to FS are proposed which attempt to deal with the challenges in further reducing the size of the feature subset while preserving the subset quality when compared with existing FS methods.

The first is a so-called self-adjusting method, which extends the original idea of HSFS. It includes three important improvements:

- The concept of a restricted feature domain is introduced in order to limit the locally explorable solution domains (of individual musicians), allowing more informative features to be located more quickly, whilst also reducing the run-time memory requirement of the algorithm. The more interesting aspect is that a restricted feature domain avoids the core where a large number of musicians select the same feature for the emerging subset. As such, it potentially improves the diversity of feature subsets.
- A harmony memory consolidation mechanism is developed, which allows musicians (that act as individual feature selectors in the algorithm) to exchange information on tentatively selected local features, and helps to identify and remove non-contributing musicians, while also allowing several new musicians to be recruited in a new iteration, since the musician group may not be sufficiently large. As a result, the size of the musician group can be dynamically adjusted during the search.
- A pitch adjustment strategy is presented which mimics the pitch adjustment behaviour of musicians. It is used by HSFS to fine tune the emerging feature subsets. In such a scheme, a feature may be substituted by its neighbour, which is determined via the use of a certain feature similarity measure.

The second directional approach extends a graph-based feature grouping approach [199], where two FS initialisations using the resulting feature groups are then derived. In particular, one is based upon music-inspired HS while the other is based upon the ranking of the obtained feature grouping (which may possibly lead to sub-optimisation). The main steps of these two FS methods are, basically, the same:

1. Construct a connected, undirected, and weighted graph by representing the features as vertices with the edges created through computing feature redundancy or collaboration with respect to the decision.
2. Generate a minimum spanning tree (MST) from this graph, where an MST is a graph representation of all the given features inter-connected such that redundant information on the edges of a given path ensures a global minimum.
3. Obtain feature groupings by breaking links (e.g., eliminating edge(s) with the minimum weight at a time) between features in the resultant MST.
4. Perform search methods across the feature grouping and evaluate the selected feature subset.
5. Repeat steps 3-4 until the algorithm terminates (e.g., no better feature subset can be discovered in the current grouping).

As well as applications to using the benchmark data, the novel techniques are also applied to a network security problem: intrusion detection [50]. The task of intrusion detection deals with malicious attacks (e.g., DOS attacks [225] and illegal access [212])on a computer network. With huge quantities of network data, such a task, in terms of building attack-predictable models, becomes computationally intractable. In particular, a dataset drawn from a large amount of network traffic may contain huge levels of redundant, irrelevant, or noisy information, which can be solved using the FS techniques. FS may, therefore, reduce this huge amount of data to a manageable size such that attack predictors can be built more efficiently while possibly being more predictive. Additionally, the approaches proposed in this thesis, which are supported with experimental evaluation, have demonstrated advantages over other existing FS methods.

1.4 Structure of Thesis

The rest of this thesis is structured as follows:

Chapter 2: Background

This chapter includes a systematic overview of existing FS techniques. It begins with an appraisal of the state-of-the-art FS methods in terms of evaluation functions

and search strategies. In contrast to these so-called "flat" FS methods that work by incrementally adding or removing individual features from a given feature subset, the following section investigates different directional FS methods, which are based on feature grouping techniques. As the approaches proposed in this thesis mainly involve HS [81] or its application to FS (HSFS) [53], this chapter includes a section devoted to discussing the basic principles of HS and HSFS.

Chapter 3: Self-Adjusting Harmony Search-based Feature Selection

This chapter presents a self-adjusting HSFS method which extends the original idea of HSFS. This technique introduces three significant strategies to improve the original HSFS. In particular, a new concept of a restricted feature domain is employed in order to limit the locally explorable solution domains (of individual musicians), which allows more useful features to be located more efficiently. A harmony memory consolidation mechanism is developed, aiming at dynamically adjusting the size of the musician group during the search. Furthermore, the pitch adjustment strategy that is used for fine tuning the emerging feature subsets is presented with a feature similarity measure such that a selected feature may be substituted by its neighbour.

Chapter 4: Feature Grouping-based Feature Selection using a Graph-theoretic Approach

In this chapter, a graph-based feature grouping framework extends the original idea proposed in [199], where three novel improvements are made. Firstly, in order to identify feature relationships, the framework includes a more powerful inter-feature measure, a three-way mutual information that is available to compute the level of redundancy and collaboration between features with respect to the decision. Secondly, the removal of irrelevant data at the earliest stage is no longer considered because datasets may display the XOR problem scenario or function where two irrelevant features combined can offer certain information to the decision. Thirdly, the framework uses a strategy of refining emerging feature groupings using the feature subsets obtained on these emerging groupings, encouraging the feature grouping process to be an internal step of FS algorithms rather than treating it as a preprocessing step that obtains a feature grouping prior to FS.

The framework itself involves a series of three primary steps. Firstly, a connected, undirected, and weighted graph is constructed by representing the features as vertices with the edges created by computing feature redundancy or collaboration with respect to the decision attribute. Secondly, an algorithm is devised to derive minimum spanning trees (MSTs) [201] from the constructed graph, where an MST is a graph representation of all given features inter-connected such that weights on the edges of a given path ensure a global minimum. Finally, feature groupings are obtained by breaking links between features in the resultant MST. This general framework can be implemented in a number of different ways to support feature selection. In this work, two particular instantiations are described, one based on the ranking of generated feature groups and the other based upon a music inspired metaheuristic (harmony search [81]).

Chapter 5: Feature Selection for Intrusion Detection

Network security has become increasingly important in today's advanced computer networks, where people are allowed to access information more easily, but which at the same time are vulnerable to a diverse range of malicious network activities such as DOS attack (e.g., crashing services and preventing legitimate requests), and illegal access (e.g., uploading malware and stealing of confidential information), which may potentially lead to severe real-life consequences such as financial loss for a bank, or the breach of military confidentiality of a nation-state. An intrusion detection system is a means for identifying and dealing with these network problems. It is, usually, a tool which requires building a predictive model using huge amounts of data that is drawn from network traffic. However, this data, which may contain irrelevant, redundant, and noisy information, is more likely to impact upon not only the speed with which predictors may be built, but also upon the accuracy of predictors. These considerations motivate the application of the FS methods proposed in this thesis to the task of intrusion detection, where building a predictive, interpretive and efficient model from data is a requirement. The experimental results show that the FS methods proposed in this thesis significantly reduce the dimensionality of KDD99 dataset [22] by several orders of magnitude while also dramatically improving the prediction accuracy of predictors built upon reduced data.

However, in the real-world intrusion detection problem, network traffics used for training are increasingly augmented and they are not static. Feature subsets

selected from a given training dataset may be out of date while new data entities came into the training dataset. Classifiers/intrusion analysers learned over such selected feature subsets may not deal with unknown data patterns well. Online FS methods [9], which perform data reduction on the fly when the training dataset changed or its volume increased, would be better alternatives.

Chapter 6: Conclusion

The thesis concludes with a summary of the key contributions as listed below,

- Two frameworks of FG-based FS are proposed.
- A new Graph-based FG method is carried out and applied to FS methods.
- Self-adjusting harmony search algorithm is proposed with three innovative mechanisms.
- All proposed FS methods in this thesis are applied to solve the task of intrusion detection.

Also, a discussion of short-term and long-term topics for further development.

Publications Arising from the Thesis

Publications are consulted in the realisation of the work presented in this thesis, containing both published papers,

- L. Zheng, R. Diao, and Q. Shen, Self-Adjusting Harmony Search-based Feature Selection, Soft Comput., vol. 19, no. 6, pp. 1567–1579, 2015.
- L. Zheng, R. Diao, and Q. Shen, Efficient Feature Selection using a Self-Adjusting Harmony Search Algorithm [240], Proceedings of the 13th UK Workshop on Computational Intelligence, 2013.

and as yet unpublished in peer-reviewed journals,

- L. Zheng, N. Mac Parthaláin, and Q. Shen, Feature Grouping and Selection using Fuzzy Linguistic Term-weighted Graph.

- L. Zheng and Q. Shen, Feature Grouping-based Feature Selection for Intrusion Detection.

Note that the latter two papers are finished in writing and currently they are under review by authors.

Appendices

Appendix A offers details of the benchmark datasets employed in this thesis.

Appendix B gives a summary of the acronyms used throughout this thesis.

Appendix C gives a summary of the symbols used throughout this thesis.

Chapter 2

Background

The growth of data both in terms of the number of features and the number of instances is a pervasive problem. The analysis of data with such high levels of dimensionality quickly becomes computationally intractable. Also, large scale data often contains irrelevant, redundant, and noisy features. FS techniques, which work by removing such features while preserving and even improving the interpretation of the underlying data, can be used to alleviate the problem caused by the "curse of dimensionality" [18]. FS can be treated as a preprocessing technique used to deal with the data after the process of data cleansing. The general framework of FS consists of two major components: feature subset evaluation and feature subset search. The flowchart shown in Fig. 2.1 presents the interaction between these two components as well as the stopping criteria, which require not only an appropriate convergence for proposed algorithms but also a guarantee in case of premature termination of the algorithms. Having generated a reduced feature set, data related to relevant features is used for further processing (e.g., the building of classification and/or clustering models). For those features filtered out, they are discarded.

The work in this thesis is focused on implementing FS based upon stochastic search strategies and a feature grouping framework. FS is first introduced in Section 2.1 with respect to feature subset evaluation and feature subset search. This is followed by Section 2.2, where the different feature grouping blueprints for FS are discussed and also the existing feature grouping methods related to FS or classification model building are reviewed. As for the nature-inspired metaheuristic,

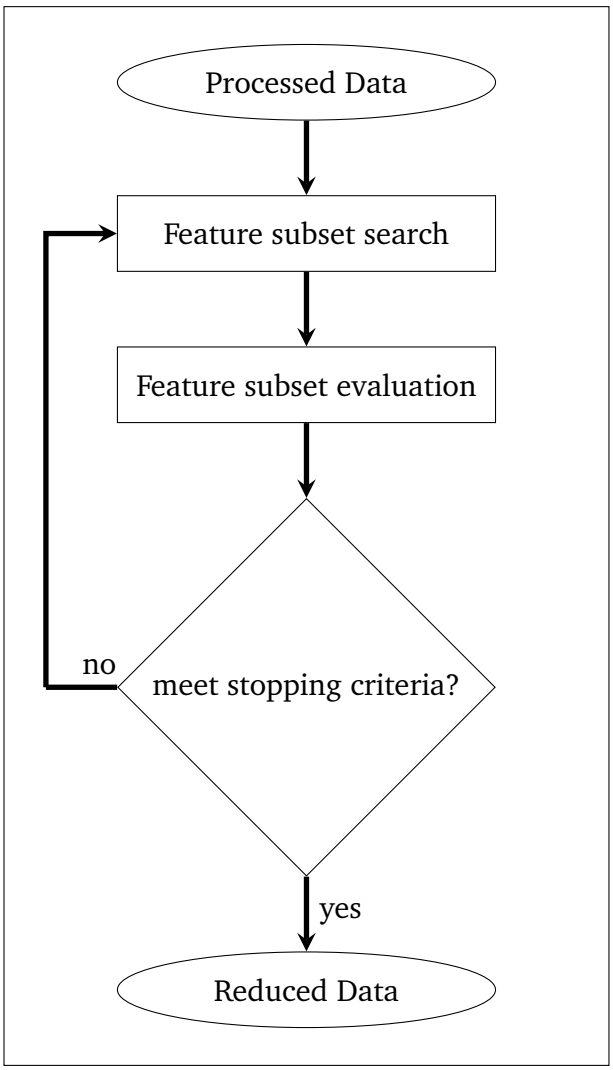

Figure 2.1: Flowchart and key components of FS

harmony search has been used and also improved for the task of FS. The principles of harmony search and its application to FS are elaborated in Section 2.3.

2.1 Feature Selection (FS)

An information system in the context of FS is a tuple $\langle X, Y \rangle$, where X is a non-empty set of finite objects (also referred to as the universe of discourse); and Y is a non-empty, finite set of features. For decision systems, $Y = \{A \cup Z\}$ where $A = \{a_1, \cdots, a_n\}$ is the set of conditional features, and n denotes the cardinality of A (note that features in A may be either continuous or discrete-valued), and Z is

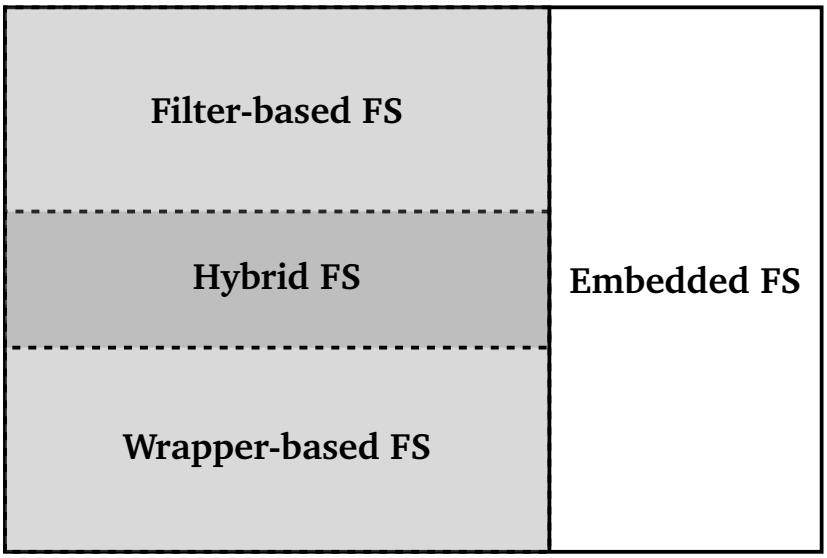

Figure 2.2: FS approaches are classified according to evaluation functions

the set of decision features. Given a dataset with n features, the task of FS is to find a subset $S \subseteq A$, which contains the most information as A, about Z while the cardinality of S is encouraged to be as small as possible.

2.1.1 FS Evaluation Functions

Various methods have been developed in the literature for the purpose of evaluating the quality of feature subsets. In general for such measures a numerical value $f(S)$ is generated for a given subset $S \subseteq A$, where A is all available conditional features in a dataset. The function $f : \mathbb{S} \rightarrow \mathbb{R}$ attempts to map a set of feature subsets onto a set of real numbers, which is often normalised in the interval $[0, 1]$. In this thesis, $f(\emptyset)$ is 0, indicating the poorest quality of selected feature subset. For any $S \in \mathbb{S}$, where $f(S)$ approaches a value of 1.0, indicates that S is a better feature subset. In particular, there may exist a set of equal quality feature subsets $\mathbb{S}' \subseteq \mathbb{S}$ in any dataset, when judged by the evaluation function. For the further specification, for any feature subset $S^p, S^q \in \mathbb{S}'$ and $S^p \neq S^q$, $f(S^p) = f(S^q)$. According to various evaluation functions, FS can usually be divided into four categories: filter, wrapper, hybrids, and embedded approaches, as illustrated in Fig. 2.2.

2.1.1.1 Filter Approaches

Filters are a collection of FS approaches that operate independently of the learning algorithm [91]. In these methods, features which are irrelevant and redundant

are removed prior to returning the resultant feature subset. Although a filter-based method is applicable for any subsequent learning algorithm, the resulting classification performance can vary depending on the feature subset returned by filter-based methods. Also, the high quality of feature subsets, which are obtained by filter-based methods, does not necessarily yield high classification accuracy when these feature subsets are used for training and testing classifier learners. As filter-based approaches to FS are very cheap (computationally), they are more popular than other approaches. The most widely employed filter-based approaches are detailed below.

Information Gain Entropy [122] is a very useful probabilistic model in information theory. It is often used for measuring the degree of uncertainty of information content through estimating the individual probabilities of its observed values. For given a set of information values observed by the feature a_x, $V_{a_x} = \{v^1_{a_x}, v^2_{a_x}, \cdots v^{|V_{a_x}|}_{a_x}\}$, the entropy of a_x can be calculated as follows:

$$H(a_x) = -\Sigma_{i=1}^{|V_{a_x}|} p(v^i_{a_x}) \log_2 p(v^i_{a_x}) \tag{2.1}$$

where $p()$ is the probability of a value taken by a feature. If the observed values of a_x are in fact partitioned according to another feature a_y, and the entropy of a_x with respect to the partitions induced by a_y is less than the entropy of a_x prior to partitioning, then there is a relationship between the two features a_x and a_y. The entropy of a_x after observing a_y, which is assumed to have a set of observed values $V_{a_y} = \{v^1_{a_y}, v^2_{a_y}, \cdots, v^{|V_{a_y}|}_{a_y}\}$, is defined as:

$$H(a_x \mid a_y) = -\Sigma_{j=1}^{|V_{a_y}|} p(v^j_{a_y}) \Sigma_{i=1}^{|V_{a_x}|} p(v^i_{a_x} \mid v^j_{a_y}) \log_2 p(v^i_{a_x} \mid v^j_{a_y}) \tag{2.2}$$

The above conditional entropy reflects uncertainty about a_x is reduced by providing a_y. Information gain [130] (or, alternatively, mutual information [65]) is given by

$$\begin{aligned} \text{information gain}(a_x, a_y) &= H(a_x) - H(a_x \mid a_y) \\ &= H(a_y) - H(a_y \mid a_y) \\ &= H(a_x) + H(a_y) - H(a_x, a_y) \end{aligned} \tag{2.3}$$

The higher value of this magnitude measured between features, the lower independency between them. However, its higher value measured between a feature and the decision illustrates this feature can provide more information for the decision.

As information gain is naturally symmetric, it is useful for measuring inter-feature correlation. Unfortunately, this measure is biased towards selecting features with higher information gain. The symmetrical uncertainty measure [231] is introduced to compensate for such bias. The values of information gain are normalised into the interval $[0, 1]$.

$$\text{symmetrical uncertainty}(a_x, a_y) = 2.0 \times [\frac{\text{information gain}(a_x, a_y)}{H(a_x) + H(a_y)}] \tag{2.4}$$

Information gain is a very common notion in the FS techniques, such as MRMR [170]. These FS algorithms attempt to select informative features by including relevant but non-redundant features or conversely excluding redundant and irrelevant features.

Interaction Gain Three-way mutual information, which is also known as interaction gain [234] is a metric which attempts to identify feature relationships, including collaboration and redundancy with respect to the decision. It is a special case for multivariate mutual information [209]. The general formation of multivariate mutual information is described as follows:

$$I(S \cup Z) = -\Sigma_{S' \subseteq S \cup Z}(-1)^{|S \cup Z| - |S'|} H(S') \tag{2.5}$$

where $S \subseteq A$ and the weight of interaction information is the sum over entropies for all possible subsets $S' \subseteq S \cup Z$. Z is a set of decision features. The entropy of a subset $H(S')$ is calculated as:

$$H(S') = -\Sigma_{a \in S'} H(a \mid (S' - \{a\})) \tag{2.6}$$

where $H(a \mid (S' - \{a\}))$ is the entropy of feature a conditioned by features except for a in S'. However, the inclusion of n features (rather than the simple binary case described above) makes it difficult to interpret the meaning of the resulting value. Therefore, n-way mutual information is difficult to adapt for the purposes of weighting the relationships between features [105].

The binary case of multivariate mutual information measures the correlation between any two features with respect to the decision. No assumptions are required when applied to feature selection techniques, such as those used in the maximum-relevance and minimum redundancy approach [170]. Also, it can be used to identify relationships between *subsets* of features that are similar, again with respect to the decision [16].

For any, two given features: $a_x, a_y \in A$ and the decision features Z with a pool of class labels $V_Z = \{v_Z^1, v_Z^2, \cdots, v_Z^{|V_Z|}\}$, supposing that a_x and a_y respectively have themselves, sets of observed values: $V_{a_x} = \{v_{a_x}^1, v_{a_x}^2, \cdots, v_{a_x}^{|V_{a_x}|}\}$ and $V_{a_y} = \{v_{a_y}^1, v_{a_y}^2, \cdots, v_{a_y}^{|V_{a_y}|}\}$, three-way mutual information can be computed as follows:

$$I(a_i, a_j, Z) = \Sigma_{l=1}^{|V_Z|}\Sigma_{i=1}^{|V_{a_x}|}\Sigma_{j=1}^{|V_{a_y}|} p(v_{a_x}^i, v_{a_y}^j, v_Z^l) \log \frac{p(v_{a_x}^i, v_{a_y}^j, v_Z^l)p(v_{a_x}^i)p(v_{a_y}^j)p(v_Z^l)}{p(v_{a_x}^i, v_{a_y}^j)p(v_{a_x}^i, v_Z^l)p(v_{a_y}^j, v_Z^l)} \tag{2.7}$$

Its values are bounded by the inequality:

$$-[H(a_x) + H(a_y)] \leq I(a_x, a_y, Z) \leq [H(a_x) + H(a_y)] \tag{2.8}$$

where $H(a_x)$ and $H(a_y)$ are the entropy of a_x and that of a_y respectively. In practical use, interaction gain is often normalised to the interval $[-1, 1]$ by the term $[(H(a_x) + H(a_y)]$. Denote the normalised $I(a_x, a_y, Z)$ as I_{xy}, then

$$I_{xy} = \frac{I(a_x, a_y, Z)}{H(a_x) + H(a_y)} \tag{2.9}$$

In common with conventional two-way mutual information, interaction gain also satisfies the symmetry property, which means that it is not influenced by the ordering of the features involved. Unlike two-way mutual information, however, three-way mutual information can have a positive, negative, or zero value. A positive interaction gain value implies collaboration between two features. Such inter-feature collaboration indicates that the two features together provide more information about the decision attribute than they do individually. The higher positive value, the stronger the collaboration. A negative interaction value implies that two features are redundant. In other words, the two features provide common information about the decision attribute. A low negative value, which tends towards -1.0, demonstrates high redundancy. A value of zero indicates that the inclusion of feature a_x (or a_y) has no impact on the relationship between a_y (or a_x) and Z. That is, a_x and a_y provide information about the decision attribute independently of one another.

Relief Relief is a class of feature weighting algorithms, including Relief [119], ReliefF [124], RReliefF [181] and their variations [34, 233]. The basic Relief algorithm only tackles classification problems with two classes although it is very

sensitive to feature interaction. ReliefF, which is an extension to Relief, improves upon the original Relief algorithm with its capacity to solve multiclass problems and deal with incomplete, noisy data. RReliefF is derived from ReliefF, and is applicable to continuous-valued class problems such as regression [102].

Given a dataset that has $|A|$ features and $|X|$ instances, Relief weights the statistical relevance of the decision features Z for each conditional feature. A relevance threshold τ in the interval $[0,1]$ determines whether the feature is selected. Detail of the Relief algorithm is shown in Algorithm 2.1.1.

$W = (0,0,\cdots,0)$: vector of feature weight that has $|A|$ elements.
$A = \{a_1, a_2, \cdots a_{|A|}\}$: set of conditional features
X: set of given instances
$S = \emptyset$: feature subset
for $i = 0$ **to** $|X|$ **do**
    Pick at random an instance $x \in X$
    Find a nearest instance x^{hit} for x in the same class
    Find a nearest instance x^{miss} for x in the different classes
    for $i = 0$ **to** $|A|$ **do**
        $W_{a_i} = W_{a_i} - \text{diff}(v^x_{a_i}, v^{x^{hit}}_{a_i})^2/|X| + \text{diff}(v^x_{a_i}, v^{x^{miss}}_{a_i})^2/|X|$
for $i = 0$ **to** $|A|$ **do**
    if $W_{a_i} > \tau$ **then**
        $S = S \cup \{a_i\}$
return S

Algorithm 2.1.1: Basic Relief for FS

Function $\text{diff}(v^{x_m}_{a_i}, v^{x_n}_{a_i})$ calculates the difference between the values of the feature a_i for two instances x_m and x_n. $v^{x_m}_{a_i}$ and $v^{x_n}_{a_i}$ represents two values taken by instances x_m and x_n respectively regarding feature a_i. When the values of the feature are nominal, this function can be defined as:

$$\text{diff}(v^{x_m}_{a_i}, v^{x_n}_{a_i}) = \begin{cases} 1 & \text{if } v^{x_m}_{a_i} \neq v^{x_n}_{a_i}, \\ 0 & \text{if } v^{x_m}_{a_i} = v^{x_n}_{a_i}. \end{cases} \tag{2.10}$$

When the values of features are numerical, this function can be defined as:

$$\text{diff}(v^{x_m}_{a_i}, v^{x_n}_{a_i}) = \frac{|v^{x_m}_{a_i} - v^{x_n}_{a_i}|}{\max(a_i) - \min(a_i)} \tag{2.11}$$

where $\max(a_i)$ and $\min(a_i)$ are the maximum value and the minimum value of feature a_i respectively. The denominator $\max(a_i) - \min(a_i)$ normalises the values of this function to the interval $[0, 1]$. This function can not only be used to compute feature weights but also be employed to locate the nearest neighbours for randomly-generated instances in Algorithm 2.1.1.

Unlike the basic Relief algorithm, ReliefF is not restricted to binary problems. For the process of finding the nearest neighbour, ReliefF has greater generalisation by searching k of nearest neighbours for randomly-generated instances instead of finding only one of the nearest neighbours in the original algorithm. Algorithm 2.1.2 presents an algorithmic description of the FS approach by using ReliefF. The user-defined parameter k controls the neighbourhood of the estimates and 10 is a value suggested [123].

$W = (0, 0, \cdots, 0)$: vector of feature weight that has $|A|$ elements.
X: set of given instances
$A = \{a_1, a_2, \cdots a_{|A|}\}$: set of conditional features
V_Z: set of class labels
$S = \emptyset$: feature subset
for $i = 0$ **to** $|X|$ **do**
    Pick at random an instance $x \in X$
    Find x k nearest instances $X_{v_Z^x} \subset X$ that have the same class label v_Z^x with x
    for $v_Z^n \in V_Z \setminus \{v_Z^x\}$ **do**
        Find x k nearest instances $X_{v_Z^n} \subset X$ that have the same class label v_Z^n with x
    for $i = 0$ **to** $|A|$ **do**
        $W_{a_i} = W_{a_i} - \Sigma_{x_i \in X_{v_Z^x}} \text{diff}(v_{a_i}^x, v_{a_i}^{x_i})^2 / (|X| \cdot |X_{v_Z^x}|) +$
        $\Sigma_{n=1}^{|V_Z \setminus \{v_Z^x\}|} [\frac{p(v_Z^n)}{1 - p(v_Z^x)} \cdot \Sigma_{x_i \in X_{v_Z^n}} \text{diff}(v_{a_i}^x, v_{a_i}^{x_i})^2 / (|X| \cdot |X_{v_Z^n}|)]$
for $i = 0$ **to** $|A|$ **do**
    if $W_{a_i} > \tau$ **then**
        $S = S \cup \{a_i\}$
return S

Algorithm 2.1.2: ReliefF for FS

In order to deal with incomplete data, missing values of features are treated probabilistically. For the case that either of the two given instances x_m (or x_n) has unknown values, the difference between the two instances of feature a_i is computed

as follows,

$$\text{diff}(v_{a_i}^{x_m}, v_{a_i}^{x_n}) = 1 - p(v_{a_i}^{x_m} \mid v_Z^{x_n}) \tag{2.12}$$

where $v_Z^{x_n}$ is the class label taken by instance x_n. For the case that both of the given instances have unknown values, the difference between two instances with respect to feature a_i is approximated using the relative frequencies of the values of feature a_i for all given classes.

$$\text{diff}(v_{a_i}^{x_m}, v_{a_i}^{x_n}) = 1 - \Sigma_{v \in V_{a_i}} (p(v \mid v_Z^{x_m}) \times p(v \mid v_Z^{x_n})) \tag{2.13}$$

where V_{a_i} is all of the values observed by feature a_i and again $v_Z^{x_m}$ is the class label taken by instance x_m.

In fact, the Relief algorithms can be theoretically explained using probability theory. The weighting of features (e.g., a feature a_i) in Relief is an approximation of the probabilistic difference:

$$\begin{aligned} \text{W}_{a_i} =& p(\text{different value of } a_i \mid \text{nearest instance of different class}) \\ & - p(\text{different value of } a_i \mid \text{nearest instance of same class}) \end{aligned} \tag{2.14}$$

By removing the context sensitivity imposed by the 'nearest instance' condition, features are treated as being independent of one another. Eqn. 2.14 can then be reformulated as:

$$\begin{aligned} \text{Relief}(a_i) =& p(\text{different value of } a_i \mid \text{different class}) \\ & - p(\text{different value of } a_i \mid \text{same class}) \end{aligned} \tag{2.15}$$

The weighting can be more formally described as:

$$\text{Relief}(a_i, Z) = \frac{Gini' \times \Sigma_{v \in V_{a_i}} p(v)^2}{(1 - \Sigma_{v_Z \in V_Z} p(v_Z)^2) \Sigma_{v_Z \in V_Z} p(v_Z)^2} \tag{2.16}$$

where V_Z is a set of class labels observed by decisional features Z and $Gini'$ is a modification of Gini-index [25], which is similar to information gain. Both of these are biased towards features with a large number of possible values. $Gini'$ becomes the measure shown below:

$$Gini' = [\Sigma_{v_Z \in V_Z} p(v_Z)(1 - p(v_Z))] - \Sigma_{v \in V_{a_i}} \left(\frac{p(v)^2}{\Sigma_{v \in V_{a_i}} p(v)^2} \Sigma_{v_Z \in V_Z} p(v_Z|v)(1 - p(v_Z|v)) \right) \tag{2.17}$$

Gini-index is differentiated from $Gini'$ by replacing $\frac{p(v)^2}{\Sigma_{v \in V_{a_i}} p(v)^2}$ with $p(v)$. In order to make Relief a symmetrical measure for any two given features, a_i and a_j, the measure is computed twice in which both features are treated as the decision feature once and the final result takes the average between them.

$$\text{Relief'}(a_i, a_j) = \frac{\text{Relief}(a_i, a_j) + \text{Relief}(a_j, a_i)}{2} \tag{2.18}$$

Correlation Correlation is a measure of the quality of feature subsets rather than an individual feature in correlation-based FS (CFS) [93] (note that it is different from the concept of classical linear correlation). Let S be a feature subset and Z be the set of decisional features, then the correlation between S and Z is calculated as:

$$\text{correlation}(S, Z) = \frac{\Sigma_{i=1}^{|S|} \text{correlation}(a_i, Z)}{\sqrt{|S| + \Sigma_{i=1, j \neq i}^{|S|} \Sigma_{j=1}^{|S|} \text{correlation}(a_i, a_j)}} \tag{2.19}$$

where correlation(a_i, Z) is the correlation between individual feature a_i and the class and correlation(a_i, a_j) is the correlation between any two features $a_i, a_j \in S$. The latter correlation is the so-called inter-correlation. Metrics including symmetrical uncertainty [231], symmetrical Relief [124], minimum description length (MDL) principle [96], and other techniques described later can also be used to calculate correlation(a_i, Z) and correlation(a_i, a_j).

In CFS, this correlation measure favours the selection of features that are highly correlated with the decision attribute and uncorrelated with each other. Irrelevant features are those that have no correlation with the decision attribute. They have more impact on the numerator. The removal of irrelevant features will result in better correlation between feature subsets and the decision attribute. The denominator controls inter-feature redundancy. High redundancy between features decreases the performance of feature subsets on this correlation measure whilst low redundancy between features increases the correlation measure. Therefore, when applying this measure to FS, features that are highly correlated with the decision attribute and less redundant with regard to each other emerge in the finally returned subset. However, the drawback of this measure is that the interaction between features, particularly those that are irrelevant, remains unconsidered. This is because a combination of irrelevant features may be highly correlated with the decision attribute.

Probabilistic Consistency The consistency measure [45] calculates the discriminability of a given feature subset $S \subseteq A$ with respect to the decision. For each feature $a_i \in S$ $(i = 1, 2, \cdots, |S|)$, assume that a_i has $|V_{a_i}|$ values. For continuous domains, this implies that feature values have to be discretised. A combination of values from all different features becomes a pattern, which is a part of an instance without the class label. The total number of patterns for S is the product of the quantity of value of all features in S, $\Pi_{i=1}^{k}|V_{a_i}|$. In practice, not all of the possible patterns have to be contained in a real-world dataset and the consistency measure is applied to relevant patterns already existing in the dataset. For all emergent patterns $\{N_S^j : j = 1, 2, \cdots, n\}$ of S, the concept of probabilistic consistency between S and the given class labels V_Z taken by decisional features Z is mathematically defined by

$$\text{consistency}(S, Z) = 1 - \sum_{j=1}^{n} \Big(\sum_{v_Z \in V_Z} p(N_S^j | v_Z) p(v_Z) - \sup_{v_Z \in V_Z} (p(N_S^j, v_Z)) \Big) \tag{2.20}$$

where $\Sigma_{v_Z \in V_Z} p(N_S^j | v_Z) p(v_Z)$ is the marginal probability of the pattern N_S^j over all the class labels in V_Z, computing the frequency of instances containing N_S^j while $p(N_S^j, v_Z)$ is the joint probability between the pattern N_S^j and a given class label v_Z, computing the frequency of instances that contain N_S^j and v_Z at the same time. Instances that contain the same pattern and the same class label are deemed to be consistent and instances that contain the same pattern but different class labels are treated as inconsistent with each other. The term $\sup_{v_Z \in V_Z}(p(N_S^j, v_Z))$ determines the most consistent instance(s) for an emergent pattern of S after taking account of all given class labels. The other instances that contain the same pattern as these most consistent instances are inconsistent with them because they possess different class labels. $\sum_{v_Z \in V_Z} p(N_S^j | v_Z) p(v_Z) - \sup_{v_Z \in V_Z}(p(N_S^j, v_Z))$ thus computes the inconsistency rate of an emergent pattern of S. Of course, the Eqn. 2.20 can be simplified as:

$$f(S, Z) = \sum_{j=1}^{n} \sup_{v_Z \in V_Z} (p(N_S^j, v_Z)) \tag{2.21}$$

as $\sum_{v_Z \in V_Z} p(N_S^j | v_Z) p(v_Z)$ sums to one for all emergent patterns of S.

Probabilistic consistency has been proven to be a monotonic measure in [10] and [45]. Given feature subsets $\{S_i : S_i \subseteq A \text{ and } i = 1, 2, \cdots, n\}$ and the decisional features Z, if $S_1 \subset S_2 \subset \cdots \subset S_n$, then $\text{consistency}(S_1, Z) \leq \text{consistency}(S_2, Z) \leq \cdots \leq \text{consistency}(S_n, Z)$. When applying this measure to the feature selection task, subsets which have higher consistency values may be returned as "optimal" feature

subsets. For the implementation of probabilistic consistency, a hashing mechanism can be used in order to improve its computational performance with linear time complexity [140].

Rough Sets A rough set is a formal approximation of a crisp set, which can be used for dealing with imperfect data. In an information system, rough set theory (RST) [169] can be used not only for decision making [147] but also to measure data dependencies [168]. The similarity between features or the correlation between features and the decision feature can be described by these data dependencies. However, rough sets are not used in this thesis but introduced for understanding its fuzzy extension. Although RST can be treated as a consistency measure, when compared with traditional crisp set-based consistency measures (e.g., probabilistic consistency [10]), it supplements their deficiency in handling uncertainty. The basic notions of RST are illustrated in Fig. 2.3.

The notion of the indiscernibility relation lies at the core of the RST. For any given subset $S \subset A$, the equivalence classes about S are identified as follows,

$$\text{IND}(S) = \{(x_i, x_j) \in X^2 \mid \forall a \in S, a(x_i) = a(x_j)\} \tag{2.22}$$

This means instances x_i and x_j are indiscernible when their values $a(x_i)$ and $a(x_j)$ described by any feature of S are the same. The equivalence classes of the S-indiscernibility relation are therefore denoted $[x]_S$. Each equivalence class is a subset of the universe of discourse X, all of which then form a partition of X.

For any subset $W \subseteq [x]_Z$ where $[x]_Z$ is a set of equivalence classes with respect to the decisional features Z, W is approximated by two constraints, which are known as the lower and upper approximations. The lower approximation $\underline{S}W$ that indicates information with certainty describes the instances of interest belonging to W:

$$\underline{S}W = \{x : [x]_S \subseteq W\} \tag{2.23}$$

The upper approximation $\overline{S}W$ extends the lower approximation. It describes the instances being included in both a class of indiscernible instances and W:

$$\overline{S}W = \{x : [x]_S \cap W \neq \emptyset\} \tag{2.24}$$

The boundary region of W with respect to S, which is identified by $\overline{S}W - \underline{S}W$, contains all the uncertain objects between $[x]_S$ and W. The positive region is the

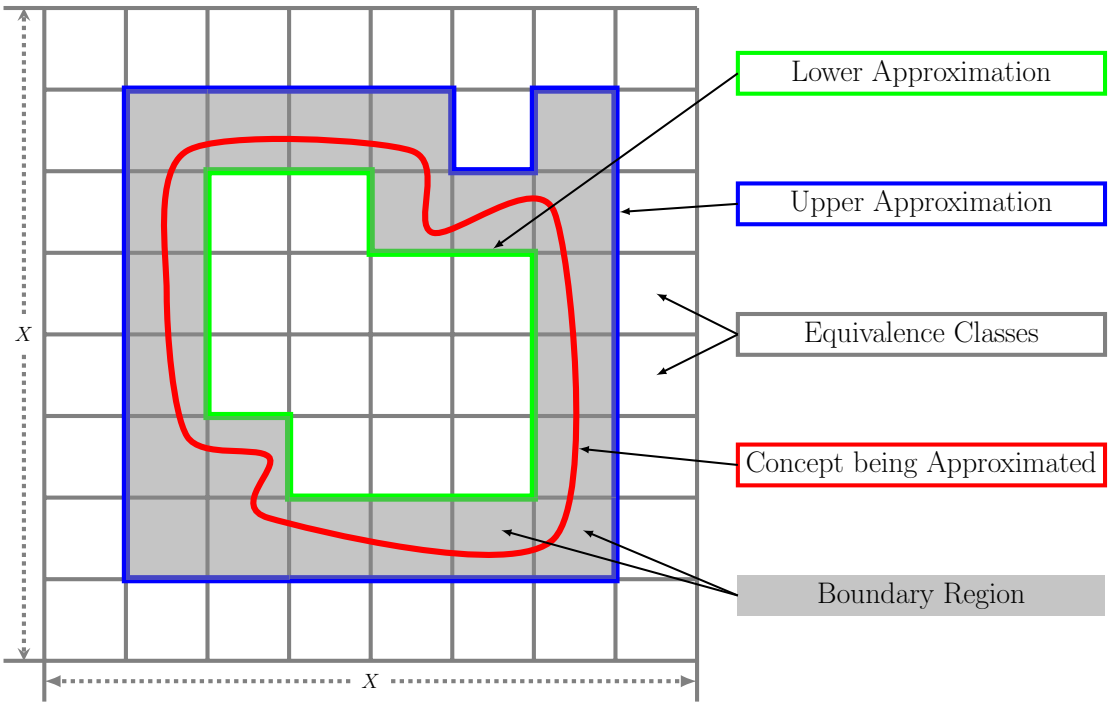

Figure 2.3: Basic notions of rough set

union of lower approximations for all elements of $[x]_Z$, $\bigcup_{W \in [x]_Z} \underline{S}W$, containing all certainty information about S.

RST has been widely used for developing FS approaches (e.g., [144], [168], [179], and [207]) as it requires no additional knowledge about the data domain. Some of the aforementioned techniques utilise feature subset measures derived from the notion of the positive region, which attempts to locate feature subsets with the maximum certainty about the decision attribute. Some of them are based on the lower approximation and the boundary region, which aim to select feature subsets with the minimum inconsistency about the decision attribute.

Fuzzy Rough Sets RST works well on discrete- or crisp-valued domains. However, for many real-world problems, the values of features are not necessarily nominal. Data discretisation is therefore needed in order to make RST computationally applicable to continuous, real value problems. In practice, data discretisation can result in information (e.g., data consistency) loss. For example, two close values observed by a real-valued feature in two different instances that have distinct class labels are in the same order of magnitude after the values of the feature have been discretised,

are considered the same value. As a result, these two values have lost their ability to distinguish between the classes. Fuzzy extensions of RST [164, 177], which are referred to as fuzzy-rough sets (FRS), are developed such that they can handle both discrete- and continuous-valued features independently of data discretisation.

The purpose of FRS is to approximate a fuzzy concept by two fuzzy sets: a fuzzy lower and a fuzzy upper approximation. In RST, instances either belong to the lower approximation with absolute certainty or not at all. In FRS, however, instances have a membership with each other ranged by the interval $[0, 1]$.

For FS, fuzzy-rough set theory can be used either to evaluate the degree of correlation between features and the decision attribute or to identify redundant information between features. Although the FRS-based FS approach developed in [108, 211] have shown to be highly useful, several problems still remain. Firstly, fuzzy sets have to be defined manually for features. Secondly, the fuzzy lower approximation might not be a subset of the fuzzy upper approximation. These issues inspire the development of novel FRS-based FS approaches in [111], where three new scenarios based on fuzzy similarity relations are developed upon notions of FRS which include:fuzzy lower approximation; the fuzzy boundary region; or the fuzzy discernibility matrix.

2.1.1.2 Wrapper Approaches

Unlike filter-based approaches, wrapper-based approaches [120, 121] utilise learning algorithms as the evaluation function for judging the quality of feature subsets. They aim to locate feature subsets that are most appropriate for a specific application. However, this type of approach has a very significant deficiency: problems with high dimensionality become computationally intractable because, for every evaluation of each located feature subset, a previously utilised learning model must be retrained upon the data described by those features in the feature subset, and classification performance is then returned as the quality of the feature subset. This situation is compounded further when learning models are complex.

There are many learning models the can be used for wrapper-based approaches, and these fall into three basic categories: classifiers, regression learners and clusterers. Many of the classifier and regression types (if not all) are often related to supervised learning [154] which attempts to infer models from labelled training

instances. Classifiers are regularly used for dealing with discrete-valued problems, whilst regression analysis is employed for tackling continuous-valued problems. A wide range of classifiers based on different mechanisms have been developed in the literature. These mechanisms mostly involve decision trees [26, 176], rule induction methods [6, 36, 159], Bayesian models [21, 48, 84], and state machines [84, 129]. As the use of different classifiers has an obvious bias in classification accuracy for the same data, diverse ensemble methods have also been developed in order to aggregate the opinions of different classifiers. The popular ensemble methods are, for example, bagging [56], boosting [186], stacking [224], and voting [156]. Most clustering methods (e.g., k-means clustering [136] and hierarchical clustering [113]) deal with unsupervised learning which can be used for building clustering models as the instances of a given dataset may be unlabelled [72].

2.1.1.3 Hybrid and Embedded Approaches

In an attempt to integrate the advanced elements of both filter and wrapper approaches, so-called 'hybrid' methods [101, 244] have been proposed where both evaluation functions and learning algorithms serve to evaluate the quality of feature subsets. In a general framework of these hybrids, evaluation functions are utilised to roughly screen feature subsets prior to applying learning algorithms. A smaller number of better feature subsets results, and to these learning algorithms are then applied to achieve feature subset refinement. As a further development, embedded methods which perform feature selection using the objective function (e.g., least squares) of a learning algorithm to estimate the quality of the candidate subset without the need for re-training have also been proposed [91]. In such methods, FS has been employed as a sub-process of learning algorithms whilst other approaches treat FS as an algorithm in isolation.

2.1.2 FS Search Strategies

Having provided diverse evaluation functions that are used for the purpose of measuring feature subset quality, this section introduces scenarios for exploring feature subsets. Given a dataset with n features, the task of FS is to find the best feature subset from 2^n possible combinations of features. An exhaustive search could be used in order to guarantee a global optimum, but this would also lead to an exponential increase in computational time-complexity. This means that exhaustive methods are often computationally intractable for feature subset search when the data is large.

To address this computational bottleneck, two broad categories of search technique have been developed for FS: greedy hill-climbers and metaheuristics. This section provides an overview of these techniques.

2.1.2.1 Greedy Hill-climbers

Greedy hill-climbers are often used for searching feature subsets in FS. They work by the incremental inclusion/exclusion of individual features from a candidate feature subset, with the aim of improving its quality. These techniques, albeit efficient in locating a feature subset, may lead to a locally optimal choice as the search sub-space they have visited may not contain the globally best solutions. In order to visit other solution regions, how is it possible to determine the most appropriate initial element of the feature subset? By way of example, a number of studies (e.g., [68]) have adopted a 'random-start' strategy to avoid the local optima. For more detail about this class of search methods, three popular strategies used to implement greedy hill-climbers for FS are listed below:

1. Stepwise forward selection: Hill-climbers in this scheme are initialised with an empty set of features. A feature that is judged to be the best of the original features is the very first added to the set. At each subsequent iteration, the best of the remainder of the original features is then added incrementally to the set.

2. Stepwise backward selection: Hill-climbers that employ this scheme are initialised with the full set of original features. Features judged the worst among the complete set of features remaining are iteratively removed.

3. Combination of forward selection and backward selection: Hill-climbers based on this scheme, which combines stepwise forward selection and backward elimination, are bi-directional search methods. At each step, these methods select the best feature while removing the worst from among the remaining features.

The stopping criteria for the methods may vary. The procedure may employ a threshold on the measure used to determine when to terminate the search process.

```
1  A = {a_1, a_2, ··· , a_|A|}: original feature set of given dataset
2  information gain(a_i, Z): information gain ratio of any feature a_i with respect
   to the decisional features Z
3  τ: threshold to determine the best feature (or the worst)
4  ã: current best feature in remaining features
5  S = ∅: candidate feature subset
6  while true do
7      ã =Random (A \ S)
8      for a_i ∈ A \ S do
9          if information gain(a_i, Z) > information gain(ã, Z) then
10             ã = a_i
11     if information gain(ã, Z) < τ then
12         return S
13     else
14         S = S ∪ {ã}
```

Algorithm 2.1.3: Stepwise forward selection for feature subsets

2.1.2.2 Metaheuristics

Metaheuristics are algorithmic frameworks of stochastic and nature-inspired search strategies. The region containing the best solution is more likely to be explored thanks to an element of randomness. However, there is no guarantee that the 'best' solution will be found; near-best solutions may often be obtained instead [88]. In order to better organise the reviewed approaches, metaheuristic algorithms can be divided into three categories that have been successful in the area of FS. Firstly, biologically-inspired approaches include the Genetic Algorithms (GA) [7], Genetic Programming [125], Memetic Algorithms [97], and the Clonal Selection Algorithm [46] from artificial immune systems. Secondly, physical, social and stochastic algorithms include Harmony Search (HS) [81], Simulated Annealing [192], Random Search [198], Scatter Search [87], and Tabu Search [85, 86]. Lastly, swarm systems include Artificial Bee Colony [114], Ant Colony Optimisation (ACO) [107], Firefly Algorithm [70] and Particle Swarm Optimisation (PSO) [219]. Of greater interest still, a number of recent studies have investigated the hybridisation of metaheuristic algorithms in order to discover (and improve upon) good candidate solutions. These hybrid methods include GA-PSO [135], ACO-GA [134], MA-PSO [137] and other possible combinations of metaheuristics.

Among the aforementioned algorithms, the most popular three are introduced in detail in the following section: GA, ACO and PSO. These three algorithms share

some common characteristics, most importantly is the fact that they are based upon a population of agents. When applying these techniques in order to solve the FS problem, each population stores a solution, representing a feature subset in a binary manner. Assuming there is a solution, taking the form "01010001", every single bit encodes a single feature: the value '1' means that the corresponding feature is selected while '0' indicates that the feature is not selected. This bit set therefore illustrates that the second, fourth, and eighth features are selected to form the feature subset, $\{a_2, a_4, a_8\}$.

FS can be a dual-objective optimisation problem. The quality of feature subsets is required to be maximised while the cardinality of those feature subsets must also be simultaneously minimised. To make a comparison between two given feature subsets: $S^{p^i} \in A$ and $S^{p^j} \in A$, an adopted scheme is formally described as follows,

$$S^{p^i} > S^{p^j} \Leftrightarrow f(S^{p^i}) > f(S^{p^j}) \vee (f(S^{p^i}) == f(S^{p^j}) \wedge |S^{p^i}| \vee |S^{p^j}|) \tag{2.25}$$

where these feature subsets are compared first in terms of the quality. The cardinality of the subsets is then used as a tie-breaker. Alternatively, in order to compare the quality of a pair of feature subsets via a single numerical difference, weighted aggregation (e.g., OWA [227]) may be applied by integrating the multiple objective functions. In this dual-objective case, the quality and the cardinality of a feature subset may be simply integrated using two weighting parameters α and β:

$$S^{p^i} > S^{p^j} \Leftrightarrow \alpha f(S^{p^i}) + \beta \frac{|A|}{|S^{p^i}|} > \alpha f(S^{p^j}) + \beta \frac{|A|}{|S^{p^j}|} \tag{2.26}$$

According to the problem at hand, these weighting parameters may be equal or biased. Of course, they can also be self-adaptive from one problem to another.

Since GA, ACO and PSO are all population-based, a number of the common representations in these three techniques can be formalised as follows:

- $p^i \in P$ — A population P of individuals p^i
- $S^{p^i} \in \mathbb{S}$ — Set of candidate feature subsets S^{p^i} maintained by p^i
- $\underset{\sim}{S}$ — Current worst subset
- $\tilde{S}$ — Current best subset

- $b_j^{S^{p^i}}$ A bit indicating selection state (0 or 1) of the j^{th} feature in S^{p^i}
- $f(S)$ The evaluated quality of any feature subset S
- λ Current iteration/generation
- λ_{max} Maximal iteration/generation
- r A random value/component
- B A temporary subset

Genetic Algorithms Genetic Algorithms (GA) [7] are inspired by observations of natural evolution. It works by passing the useful genetic information of parents to offspring through operating events such as crossover and mutation of chromosomes, a chromosome being a set of genes which are the carriers of genetic information. A considerable number of studies in the literature (e.g., [160, 228]) have argued for the usefulness and relevance of applying GA to FS. In these implementations, each gene is used to represent the binary state of a feature: '1' indicates that a respective feature is active and then selected; and '0' signals that the feature is not selected. That is, each chromosome acts as a feature subset.

The FS processes using GA are presented in Algorithm 2.1.4. Lines 5-7 are the initialisation stage, where the initial population P is formed by randomly generating feature subsets. The size of P is predefined and maintained in successive generations. The population reproduction module is depicted in lines 8-32. The assignment statement in line 10 attempts to propagate the current best feature subset to the next generation such that the useful features can pass from generation to generation, although these current best subsets may not be the best in the succeeding generation. For breeding each pair of a new population, two individual chromosomes are randomly selected from the current population and manipulated mainly using crossover and mutation operators. Lines 18-22 describe a scenario of operating crossover on two selected chromosomes. It first locates a crossover point along the length of the chromosome, and then exchanges the gene sequences in the same structure such that two new chromosome are produced. However, a gene naturally mutate. Therefore, a strategy of gene mutation for these two new chromosome is presented in lines 23-28. If the mutation event of a gene happens, the state of this gene will be set to the negation of its current state. Both crossover and mutation

events are respectively controlled by two threshold parameters: the crossover rate r_c and the mutation rate r_m. The reproduction process for a single generation is not completed until the size of the newly-produced population is equal to that of the current population. After that, the new population takes the place of the current population, where all feature subsets are evaluated and the current best subset is then updated. The algorithm terminates when the current iteration λ satisfies λ_{max} or the quality of the best and worst subsets evaluates to equal.

The GA-based FS algorithm does not change the flow of the internal processes of the original GA, which makes it simple to implement with slightly fewer notions translated such as mapping feature subsets into chromosomes. Being a randomised algorithm, however, the optimisation response time and the selected subset are not deterministic. And there is no guarantee that the best feature subset (if not the global best subset) can be found in a predefined amount of iterations. For tasks of on-line streaming FS, the effectiveness of GA will be less because of these drawbacks. Finding a reasonable setting for predefined parameters for a specified problem also becomes challenging since the problem domain of FS is hugely varied.

Ant Colony Optimisation The Ant Colony Optimisation (ACO) algorithm and its variants have been systematically introduced and investigated in a number of studies such as [57, 58, 60]. It is mainly used for solving difficult combinatorial optimisation problems (e.g., a popular travelling salesman [59], vehicle routing [17], and scheduling [152]). In particular, the task of FS can be considered as a combinatorial problem. Quite a number of studies [107, 115, 192] have been carried out in order to apply ACO to FS, and bear this observation out. The key idea of ACO is based on the foraging behaviour of ants, which is capable of locating the shortest path between colony and food source through biologically-mediated communication (e.g., pheromone).

In ACO-based FS algorithms, features are represented as nodes in a fully connected undirected graph, and a candidate feature subset S is therefore a path connecting the visited features. The ant movements on the graph are guided by two sets of hints: the heuristic information η and the pheromone values τ. η is a two-dimensional matrix, which is constructed prior to the ant search and then maintained until the algorithm terminates. The size of the matrix is $|A| \times |A|$, where A is the original input feature set. Unit $\eta_{ij} = \eta_{ji}$ stores the evaluated quality of the

$p^i \in P, i = 1$ **to** $|P|$: the initial population
S^{p^i}, S^{p^j}: existing feature subsets associated with p^i and p^j respectively
r_c: crossover rate
r_m: mutation rate
// Initialisation
for $i = 1$ **to** $|P|$ **do**
    for $j = 1$ **to** $|A|$ **do**
        $b_j^{S^{p^i}} =$ Random $(\{0, 1\})$
// Reproduction
$\lambda = 1$
while $(\lambda + +) < \lambda_{max} \wedge f(\underset{\sim}{S}) \neq f(\tilde{S})$ **do**
    $B^1 = B^2 = \tilde{S}$
    for $i = 3$ **to** $|P|$ **do**
        $B^i = S^{p^i}$ with a probability of $\frac{f(S^{p^i})}{\Sigma_{i=1}^{|P|} f(S^{p^i})}$
        $B^{i+1} = S^{p^j}$ with a probability of $\frac{f(S^{p^j})}{\Sigma_{i=1}^{|P|} f(S^{p^i})}$
        if $B^i == B^{i+1}$ **then**
            $r =$ Random $(\{1, 2, \cdots, |A|\})$
            $b_r^{B^i} = \neg b_r^{B^i}$
        else
            // Crossover
            $r =$ Random $([0, 1])$
            if $r < r_c$ **then**
                $r =$ Random $(\{1, \cdots, |A| - 1\})$
                for $k = 1$ **to** r **do**
                    $b_k^{B^{i+1}} = b_k^{S^{p^i}}$, $b_k^{B^i} = b_k^{S^{p^j}}$
            // Mutation
            for $k = 1$ **to** $|A|$ **do**
                $r =$ Random $([0, 1])$
                if $r < r_m$ **then**
                    $b_k^{B^i} = \neg b_k^{B^i}$
                if $r < r_m$ **then**
                    $b_k^{B^{i+1}} = \neg b_k^{B^{i+1}}$
        $i = i + 2$
    for $i = 1$ **to** $|p|$ **do**
        $S^{p^i} = B^i$
    Evaluate S^{p^i}, for all $p^i \in P$
    Update $\tilde{S}$

Algorithm 2.1.4: Genetic Algorithm for FS

feature subset $\{a_i, a_j\}$, and illustrates the correlation between a_i and a_j. τ is another matrix of the same size with η that stores the intensities of pheromone deposited by the ants. τ_{ij} indicates the intensity of a single path between a_i and a_j, which is often initialised with a constant value τ_0.

As seen in Algorithm 2.1.5, for every iteration, each ant starts from a random feature a_r, which is set as the current feature a_c of ants. The probability of the move from a_c to the next unvisited feature a_u is determined by

$$\text{prob}_u = \frac{\tau_{cu}^{\alpha} \eta_{cu}^{\beta}}{\Sigma_{a_u \text{not visited}} \tau_{cu}^{\alpha} \eta_{cu}^{\beta}} \tag{2.27}$$

where α and β are predefined weighting parameters. Unlike η, which is fixed after construction, τ is dynamically changeable during the activity of ants. The update of τ is based on two observations:

1. A certain proportion of pheromone could naturally evaporate such that the intensity of pheromone will be reduced over time.
2. Every single ant traversing a path will lay down its pheromone (suppose that all ants release the same amount of pheromone), which will increase the intensity of pheromone.

In addition to the mechanism of updating pheromone described in Algorithm 2.1.5, various pheromone-updating strategies are devised in a number of studies (e.g., [33, 107, 115]). A common approach to updating pheromone with respect to these two observations is then presented as follows:

$$\tau_{ij} = \rho \tau_{ij} + \Delta \tau_{ij} \tag{2.28}$$

where

$$\Delta \tau_{ij} = \Sigma_{i=1}^{|P|} \frac{f(S^{p^i})}{|S^{p^i}|} \tag{2.29}$$

when the path between two features: a_i and a_j has been traversed. Otherwise, $\Delta \tau_{ij}$ is zero. The parameter ρ is a decay constant, which is used to simulate the evaporation of pheromone. In particular, line 25 in Algorithm 2.1.5 presents a method that properly stops an ant and then completes the search for a feature subset, once the inclusion of further features cannot improve the quality of the current feature subset.

These intelligent ants are capable of discovering a desirable feature subset. However, building the predefined η requires $|A| \times |A|$ subset evaluations. That is, the ACO-based FS algorithm may become computationally impractical for large datasets. To address this problem, two methods can be considered: one is to use a more compact evaluation function in order to reduce the time cost of evaluating subsets; the other is to identify the core feature subset in advance such that only features except for the core subset are used to build η, which could reduce the dimensionality of η and shorten the subset evaluation times. Also, the influence of configuring ρ on subset selection remains to be further investigated. To improve the performance of subset selection, dynamic strategies, for example, can be used to adjust the value of ρ iteratively.

Particle Swarm Optimisation As a swarm intelligence implementation, the original particle swarm optimiser (PSO) [117] attempts to locate the optimal solution in search space by sending a population of intelligent particles P, which are capable of achieving information vantage points from which to determine the global best (gbest) and the past best (pbest). The global best is the best solution achieved so far by any particle in the population and the past best is the best solution obtained so far by an individual particle. Due to the popularity and simplicity of the PSO algorithm, many of its variants have been developed to significantly improve its search power. In [191], an additional parameter called inertia weight is introduced to the standard PSO algorithm and used to fine-tune the original velocity of particles. In [205], the standard particle swarm optimiser is improved by the addition of a neighbourhood operator by which particles are able to interact with neighbours. In [118], a binary version of the PSO algorithm is proposed such that PSO can deal with discrete binary variables.

When applied to FS [219] (see Algorithm 2.1.6), the velocity v_i of a given feature subset S^{p^i} representing the number of features to be changed is computed by:

$$v_i = wv_i + c_1 r_1 d(\tilde{S}, S^{p^i}) + c_2 r_2 d(\tilde{S}^{p^i}, S^{p^i}) \tag{2.30}$$

where w is the inertia weight used to linearly reduce the velocity of particles. The weights c_1 and c_2 in Eqn. 2.30 are the acceleration constants, which control the roaming distance of particles. The terms r_1 and r_2 are the random components: they embody the randomness of PSO in conjunction with stochastically initialising P. The function $d()$ is used to calculate the distance between two feature subsets. To

$p^i \in P, i = 1$ **to** $|P|$: the ant population
S^{p^i}: current path (feature subset) traversed by ant p^i
$\eta_{ij} = \eta_{ji}, i, j = 1$ **to** $|A|$: heuristic information
$\tau_{ij} = \tau_{ji}, i, j = 1$ **to** $|A|$: pheromone intensities
ρ: pheromone evaporation rate
τ_0: constant pheromone
τ_{sum}: total amount of pheromone over τ and used as a nominator
`// Initialisation`
for $i = 1$ **to** $|A| - 1$ **do**
    for $j = i + 1$ **to** $|A|$ **do**
        $\eta_{ij} = f(a_i, a_j)$
        $\tau_{ij} = \tau_0$
`// Ant Traversals`
$\lambda = 1$
while $(\lambda + +) < \lambda_{max}$ **do**
    `// Pheromone Evaporation`
    for $i = 1$ **to** $|A| - 1$, $j = i + 1$ **to** $|A|$ **do**
        $\tau_{ij} = \rho \tau_{ij}$
        $\tau_{sum} += \tau_{ij}$
    `// Normalisation`
    for $i = 1$ **to** $|A| - 1$, $j = i + 1$ **to** $|A|$ **do**
        $\tau_{ij} = \frac{\tau_{ij}}{\tau_{sum}}$
    `// Path Construction`
    for $i = 1$ **to** $|P|$ **do**
        $r =$ `Random` $(\{1, 2, \cdots, |A|\})$
        $S^{p^i} = S^{p^i} \cup \{a_r\}$
        Current feature possessed by i^{th} ant, $a_c = a_r$
        while $|S^{p^i}| < |A|$ **do**
            select $a_u \notin S^{p^i} \wedge \text{prob}_u$ is the largest
            if $f(S^{p^i} \cup \{a_u\}) < f(S^{p^i})$ **then**
                break
            else
                $S^{p^i} = S^{p^i} \cup \{a_u\}$
                $a_c = a_u$
                $\tau_{cu} = (\frac{1 - f(S^{p^i})}{2}) + f(S^{p^i})\tau_{cu}$
    for $i = 1$ **to** $|P|$ **do**
        for $i = 1$ **to** $|A| - 1$, $j = i + 1$ **to** $|A|$ **do**
            $\tau_{ij} = \tau_{ij} + f(S^{p^i})$
    Update $\tilde{S}$

Algorithm 2.1.5: Ant Colony Optimisation for FS

implement this function, Hamming Distance [94] could be an option to compute the distance between any two subsets S^{p^i} and S^{p^j}:

$$d(S^{p^i}, S^{p^j}) = |S^{p^i} \oplus S^{p^j}| \tag{2.31}$$

The velocity of particles is not positively infinite; it is limited by the predefined maximum velocity v_{max}. However, the setting of v_{max} is an intractable problem. If v_{max} is set to a small value, it is highly possible that particles fly around their past best subset. If v_{max} is set to a large value, particles may easily bias good subsets. In the literature [219], the value of v_{max} is initialised with the number of conditional features $|A|$ of a problem at hand and then set to a third of $|A|$ during the iteration stage. This configuration of v_{max} has the obvious effect in subset selection for a number of datasets. For more general approaches, configuring v_{max} in a dynamic scheme are worth further investigating.

2.2 Approaches Related to Feature Grouping

In this section, a number of studies [12, 106, 199, 242, 243] concerning feature grouping are introduced. Most of these implement feature grouping via clustering similar (or highly dependent) features together, some of which then use the resulting feature grouping for the task of FS. Since there exist few general approaches to feature grouping in the literature, several recently developed feature grouping methods, which relate closely to this research, are then reviewed in what follows.

2.2.1 FS using Correlation Coefficient Clustering

In an attempt to determine the relevance between the features themselves and the decision attribute, information-based metrics such as mutual information [65], correlation coefficient [93], and fuzzy-rough set dependency [177] may be used. In [100], a correlation coefficient is used to assist in identifying pair-wise redundancy between features and also the relatedness between these features and the decision. The conventional K-means method is adopted for grouping of features. By selecting a representative feature from each group, a feature subset is then formed. The representative feature chosen from each group depends on its correlatedness to the decision attribute. Those features most correlated to the decision attribute are selected to represent the full group of features. However, this approach requires the

$p^i \in P, i = 1$ **to** $|P|$: the particle population
S^{p^i}: feature subset found by particle p^i
c_1, c_2: acceleration constants
$w \in [w_{min}, w_{max}]$: inertia weight
$v_i \in [1, v_{max}]$: current velocity of particle p^i
// Initialisation
Randomly generate feature subsets
// Search Iteration
$\lambda = 1$
while $(\lambda++) < \lambda_{max}$ **do**
    Update $\tilde{S}, \tilde{S}^{p^i}$ (including feature subset evaluations)
    for $i = 1$ **to** $|P|$ **do**
        r_1 =Random ($[0,1]$)
        r_2 =Random ($[0,1]$)
        // Update Velocity of particle p^i
        $v_i = wv_i + c_1 r_1 d(\tilde{S}, S^{p^i}) + c_2 r_2 d(\tilde{S}^{p^i}, S^{p^i})$
        if $v_i > v_{max}$ **then**
            $v_i = v_{max}$
        if $v_i < 1$ **then**
            $v_i = 1$
        // Update Feature Subset
        if $v_i > d(\tilde{S}, S^{p^i})$ **then**
            while $k < v_i$ **do**
                for $j = 1$ **to** $|A|$ **do**
                    if $b_j^{\tilde{S}} \wedge b_j^{S^{p^i}} == 1$ **then**
                        $b = b_j^{S^{p^i}}$, $b_j^{S^{p^i}}$ =Random ($\{0,1\}$)
                        if $b == b_j^{S^{p^i}}$ **then**
                            $k++$
        else
            while $k < v_i - d(\tilde{S}, S^{p^i})$ **do**
                for $j = 1$ **to** $|A|$ **do**
                    if $b_j^{\tilde{S}} \wedge b_j^{S^{p^i}} == 0$ **then**
                        $b = b_j^{\tilde{S}}$, $b_j^{\tilde{S}}$ =Random ($\{0,1\}$)
                        if $b == b_j^{\tilde{S}}$ **then**
                            $k++$
            $S^{p^i} = \tilde{S}$
    $w = w_{min} + (1 - \frac{\lambda}{\lambda_{max}})(w_{max} - w_{min})$

Algorithm 2.1.6: Particle Swarm Optimisation for FS

specification of the number of feature groups and, therefore, the cardinality of the selected feature subset in advance.

2.2.2 FS using Graph-based Clustering

In [199], an efficient clustering-based feature selection algorithm is proposed for high dimensional data. This algorithm adopts the symmetrical uncertainty measure defined by Shannon entropy to gauge inter-feature correlation and correlation between a conditional feature and the decision attribute. A feature is considered to be irrelevant when the degree of symmetrical uncertainty between it and the decision attribute is less than the predefined threshold. Otherwise, features are treated as relevant. The degree of symmetrical uncertainty between features illustrates the level of redundant information. By exploiting these rules, FS is implemented by an algorithm which includes the following steps: 1) removing irrelevant features; 2) clustering features of the most redundant information using graph-theoretic methods; and 3) selecting the most relevant feature from each group to form the final selected feature subset. Using this algorithm, it does not require to a predefined number of feature groups, which can be determined iteratively by the algorithm itself. However, features within the same group may have a large difference in relevance between a single feature and the decision attribute.

2.2.3 Fuzzy Rough-based FS using Feature Grouping

In [106], a hill-climbing approach to FS based on feature grouping is proposed, where an evaluation metric based on fuzzy-rough set dependency is utilised to determine the internal ranking of the features in each group as well as the overall subset quality. A correlation coefficient is used to calculate the degree of redundancy between any pair of features. If the correlation between a given pair of features is greater than a predefined threshold, then one is considered to be redundant and both features are then assigned to the same group. Each individual group is initialised with a single distinct feature prior to recruiting other group members. As a result, an individual feature can be included or assigned to more than one group. Features are then internally ranked within each of the groups according to fuzzy-rough set dependency prior to returning the final subset. This algorithm is generally efficient, but it requires to assume that what degree of correlation coefficient between features is considered redundant. Different configurations of this parameter may have significantly different impacts upon the FS outcome.

2.2.4 Feature Transformation using Feature Grouping

In addition to the aforementioned representations of feature grouping-based FS methods, more recently the concept of feature grouping has also been used to shrink the regression models by removing or merging redundant features. In [242], for instance, a feature grouping method is embedded within the process of sparse modelling. Firstly, the popular OSCAR algorithm [64] is used to generate a so-called coefficiency matrix between features. Those features which have identical coefficients are then grouped together and those with coefficients of zero are immediately discarded. The new features formed by merging features in the same group are subsequently used to train a sparse regression model. Testing against selected real-world datasets (e.g., breast cancer [22]), the regression models generated by this algorithm may be more robust than those obtained by conventional methods, though this may not always be the case. Nevertheless, with the growth of the dimensionality of problems at hand, this particular algorithm tends to be more efficient than others.

2.3 Feature Selection with Harmony Search (HSFS)

The initial purpose of this research is related to the idea of harmony search for the task of FS. This section consists of two parts. The first part introduces the principles of HS and its variants. The second part presents the mechanism for applying HS to FS.

2.3.1 Harmony Search (HS)

HS is a meta-heuristic search algorithm which mimics the improvisation process of musical performers, and is primarily oriented towards discrete-valued variables rather than continuous variables in differential calculus. Each musician represents a decision variable of the objective function playing a note (value) in order that the ensemble may construct a harmony (solution) that optimises this function. Newly generated harmonies are iteratively progressed based on musicians' experience (a pool of existing harmonies) and used to update historical solutions with respect to the harmony quality.

2.3.1.1 Principles of HS

The original purpose of the HS algorithm is to solve optimisation problems. The process consists of five steps, and is described in detail as follows:

1. Initialise parameters: Basic HS uses five pre-defined parameters, including harmony memory size ($|\mathbb{H}|$), the harmony memory considering rate (HMCR), the maximum number of iterations (λ_{max}), the pitch adjustment rate (PAR) and the adjusting bandwidth (BW). $\mathbb{H}$ is a set of harmonies stored in the harmony memory (HM). HMCR and PAR respectively control the global and local search of the HS algorithm. In a typical implementation, both of them take values ranged from 0 to 1. BW is an arbitrary length only for continuous variables and used to adjust the found value. λ_{max} controls the upper limit of the search progress. In addition, another parameter, the group of musicians M is defined by the problem itself, the size of which denoted as $|M|$ is equal to the number of variables in the function being optimised.

2. Initialise HM: A two-dimensional matrix is used to represent HM, where each row indicates a solution vector and each column dedicated to a single musician stores the musician's experience. In HM, the number of rows is predefined by $|\mathbb{H}|$ and the number of columns is equal to $|M|$. Thus, the HM takes the following form:

$$\mathrm{HM} = \left[\begin{array}{cccc|c} x_1^{H^1} & x_2^{H^1} & \cdots & x_{|M|}^{H^1} & f(\mathbf{x}^1) \\ x_1^{H^2} & x_2^{H^2} & \cdots & x_{|M|}^{H^2} & f(\mathbf{x}^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{H^{|\mathbb{H}|}} & x_2^{H^{|\mathbb{H}|}} & \cdots & x_{|M|}^{H^{|\mathbb{H}|}} & f(\mathbf{x}^{|\mathbb{H}|}) \end{array}\right].$$

   $\overline{x}_i^{H^j}$ denotes a value taken by a unit $x_i^{H^j}$ where i indicates i^{th} variable and H^j is the j^{th} harmony. $f(\mathbf{x})$ is an objective function, evaluating the quality of a given solution vector $\mathbf{x}$.

3. Improvise a new harmony: A new solution vector, $\mathbf{x}' = (\overline{x}_1^{H'}, \overline{x}_2^{H'}, \cdots, \overline{x}_{|M|}^{H'})$ is generated considering three impact factors: HMCR, PAR, and a random value r, taking a value between 0 and 1. A new value $\overline{x}_i^{H'}$ of the i^{th} variable is

obtained according to the following rules:

$$\overline{x}_i^{H'} = \begin{cases} \overline{x}_i^{H'} \in X_i & \text{if HMCR} \leq r \\ \overline{x}_i^{H'} \in \text{HM}_i & \text{if HMCR} > r \\ \overline{x}_i^{H'} + \text{random}(-1,1) \times \text{BW} & \text{if PAR} > r \end{cases} \tag{2.32}$$

where X_i is a set of all possible values of the i^{th} variable and $\text{HM}_i = \{\overline{x}_i^{H^1}, \overline{x}_i^{H^2}, \cdots, \overline{x}_i^{H^{|\mathbb{H}|}}\}$ is the i^{th} column of HM, indicating the historical values of the i^{th} musician. During this improvisation process, when either the condition HMCR $\leq r$ or HMCR $> r$ is matched, the value $\overline{x}_i^{H'}$ will be randomly generated from X_i or HM_i respectively. If the condition PAR $> r$ is satisfied, $\overline{x}_i^{H'}$ is obtained by randomly generating a value out of all possible historical values and adjusting this value based on the formula $\overline{x}_i^{H'} + \text{random}(-1,1) \times \text{BW}$. Alternatively, BW can be replaced by using musicians' own experiences: $\overline{x}_i^{H'} + \text{random}(-1,1) \times (v_i^U - v_i^L)$ where v_i^U and v_i^L are the maximum value and the minimum value of the i^{th} musician in the HM respectively. Other variables choose new values for the new solution in the same manner. A single improvisation is completed once all variables have nominated a value. To further ease the understanding of HS, Algorithm 2.3.1 presents an outline of the improvisation procedure in pseudocode.

```
|M|: number of musicians (variables)
x' = (x_1^H', x_2^H', ..., x_|M|^H'): new solution vector
X_i: value domain of i^th musician
HM_i = {x_i^H1, x_i^H2, ..., x_i^H|𝕄|}: the i^th column of HM indicating set of historical
  values of the i^th musician
BW: arbitrary distance bandwidth
for i = 1 to |M| do
    if Random ([0,1])<HMCR then
        x_i^H' =Random (HM_i)
    else
        x_i^H' =Random (X_i)
    if Random ([0,1])<PAR then
        x_i^H' = x_i^H' +Random ([-1,1])×BW
```

Algorithm 2.3.1: Improvisation process of original HS

4. Update HM: The quality of the newly generated solution is evaluated using a pre-specified objective function $f(\mathbf{x})$. If the new solution $\mathbf{x}'$ obtains a higher evaluation than any of the existing solutions, $\mathbf{x}'$ will replace the worst solution in HM. This is symbolically equivalent to:

$$\mathbf{x}' \in \text{HM} \wedge \mathbf{x}^{\text{worst}} \notin \text{HM} \tag{2.33}$$

   Otherwise, do nothing.

5. Check stopping criterion: If the number of improvisation reaches λ_{max}, the algorithm stops. Otherwise, repeat step 3 and step 4.

The PAR and BW play a very important role in the HS algorithm. They not only have the ability to fine-tune the improvised harmony but also influence the convergence rate of the algorithm. The original fixed values of PAR and BW through the whole algorithm impair the flexibility of HS because the different stages of the algorithm may require configuration of a pair of different values for PAR and BW. Therefore, in order to eradicate the drawbacks of fixed values of PAR and BW, a scheme of dynamically adjusting the values of PAR and BW with each further iteration is introduced in [145]. At the outset of the algorithm, PAR and BW are set to a small value and a large value respectively. The value of PAR is linearly increased based on the following formula:

$$\text{PAR} = \text{PAR}_{min} + \frac{\text{PAR}_{max} - \text{PAR}_{min}}{\lambda_{max}} \times \lambda \tag{2.34}$$

where PAR_{max} and PAR_{min} are the maximum and minimum PAR respectively.

And the value of BW is logistically reduced by

$$\text{BW} = \text{BW}_{max} \times \exp\left(\frac{\log(\frac{\text{BW}_{min}}{\text{BW}_{max}}) \times \lambda}{\lambda_{max}}\right) \tag{2.35}$$

where BW_{max} and BW_{min} are the maximum and minimum BW respectively. However, the determination of a suitable set of BW_{max} and BW_{min} (or PAR_{max} and PAR_{min}) becomes another new problem.

2.3.1.2 Applications of HS

Since the structure of HS is simple and easy to implement, HS has been widely applied to various science and engineering optimisation problems [77, 79, 82]. These include: real-world applications (e.g., tour planning, timetabling and sudoku puzzle); bio & medical applications (e.g., RNA structure prediction and hearing aids); computer science problems (e.g., internet routing, web page clustering, and robotics); electrical engineering problems (e.g., energy system dispatch, photo-electronic detection and multi-level inverter optimisation); civil engineering problems (e.g., vehicle routing, flood model calibration and structural design); and mechanical engineering problems (e.g., satellite heat pipe design, heat exchanger design and offshore structure mooring).

2.3.2 HSFS Algorithms

The original HSFS algorithm [54], in terms of approaches to representing feature subsets, has two versions: binary-valued and integer-valued HSFS.

2.3.2.1 Binary-valued HSFS

Bit set representation of a feature subset is very common in metaheuristic-based FS techniques. When applying HS to the binary-valued FS problem, musicians are directly mapped onto all available features of a given dataset, which will take a value between '0' and '1'. Each bit therefore indicates the selection state of its corresponding feature. The value '1' means that this feature is selected as the element of emerging subsets while '0' means that this feature is not selected as that of emerging subsets at all.

The binary-valued HSFS algorithm continues to use the main structure of the original HS. The steps of the binary-valued HSFS algorithm are then listed as follows:

1. Initialise parameters: In this algorithm, four parameters of the original HS remain to be used, including $|\mathbb{H}|$, HMCR, λ_{max} and $|M|$. PAR and BW are replaced by using the flip rate (FR), which works in the same way as the mutation rate of GA. Therefore, every single bit has the FR probability being flipped between the two values '0' and '1'. In particular, $|M|$ is firmly set to $|A|$, where A is all available features of a given dataset. That is, every single musician is denoted as a distinct feature.

2. Initialise HM: HM now is a two-dimensional matrix with the size of $|\mathbb{H}| \times |A|$. Each row indicates a feature subset while each column indicates the historical state of its corresponding feature. The form of HM is as follows:

$$\mathrm{HM} = \left[\begin{array}{cccc|c} b_{a_1}^{H^1} & b_{a_2}^{H^1} & \cdots & b_{a_{|A|}}^{H^1} & f(S^1) \\ b_{a_1}^{H^2} & b_{a_2}^{H^2} & \cdots & b_{a_{|A|}}^{H^2} & f(S^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{a_1}^{H^{|\mathbb{H}|}} & b_{a_2}^{H^{|\mathbb{H}|}} & \cdots & b_{a_{|A|}}^{H^{|\mathbb{H}|}} & f(S^{|\mathbb{H}|}) \end{array}\right].$$

A bit $b_{a_i}^{H^j}$ denotes the selection state of the i^{th} feature a_i in the j^{th} harmony S^{H^j}. The value of the bit $b_{a_i}^{H^j}$, $\overline{b}_{a_i}^{H^j}$ is between '0' and '1'. This means every single musician only has a value domain of two choices. PAR and BW, which are originally used for fine-tuning continuous variables, may be too powerful to be exploited for such binary-valued variables. A harmony S^{H^j} turns out to be a feature subset S^j via the inclusion of features that are valued by '1'. $f(S^j)$ is the evaluation result of S^j, which can be estimated by any of the subset measures introduced in Section 2.1.1.

3. Improvise new feature subset: A new harmony $S^{H'} = \{\overline{b}_{a_1}^{H'}, \overline{b}_{a_2}^{H'}, \cdots, \overline{b}_{a_{|A|}}^{H'}\}$ is generated based on factors involving HMCR, FR and a random number r ($0 < r < 1$). The state of each feature in $S^{H'}$ is changed in accordance with the following rules:

$$\overline{b}_{a_i}^{H'} = \begin{cases} \overline{b}_{a_i}^{H'} \in \{0,1\} & \text{if HMCR} \leq r \\ \overline{b}_{a_i}^{H'} = 0 & \text{if HMCR} > r \wedge \Sigma_{j=1}^{|\mathbb{H}|} \overline{b}_{a_i}^{H^j} < \frac{|\mathbb{H}|}{2} \\ \overline{b}_{a_i}^{H'} = 1 & \text{if HMCR} > r \wedge \Sigma_{j=1}^{|\mathbb{H}|} \overline{b}_{a_i}^{H^j} > \frac{|\mathbb{H}|}{2} \\ \overline{b}_{a_i}^{H'} = \neg \overline{b}_{a_i}^{H'} & \text{if FR} \leq r \end{cases} \tag{2.36}$$

HMCR is still important in this algorithm although the state of every single bit in $S^{H'}$ is randomly determined between two choices when the value of HMCR is less than a random value r. However, if HMCR is larger than a random value, the situation of selecting a value is slightly different from the original HS algorithm. The bit takes the value that most frequently emerges in the historical HM values of its corresponding feature. FR works independently of HMCR. It enables the fine-tuning of the selection state of a feature by flipping the currently selected value to its negation. The improvisation of such binary-valued feature subset is then algorithmically depicted in Algorithm 2.3.2.

1 $|M|$: number of musicians (features)
2 $S^{H'} = \{\overline{b}_{a_1}^{H'}, \overline{b}_{a_2}^{H'}, \cdots, \overline{b}_{a_{|A|}}^{H'}\}$: new binary-valued harmony
3 **for** $i = 1$ **to** $|M|$ **do**
4 **if** Random $([0,1])$<*HMCR* **then**
5 **if** $\Sigma_{j=1}^{|\mathbb{H}|} \overline{b}_{a_i}^{H^j} < \frac{|\mathbb{H}|}{2}$ **then**
6 $\overline{b}_{a_i}^{H'} = 0$
7 **else if** $\Sigma_{j=1}^{|\mathbb{H}|} \overline{b}_{a_i}^{H^j} > \frac{|\mathbb{H}|}{2}$ **then**
8 $\overline{b}_{a_i}^{H'} = 1$
9 **else**
10 $\overline{b}_{a_i}^{H'}$ =Random $(\{0,1\})$
11 **if** Random $([0,1])$<*FR* **then**
12 $\overline{b}_{a_i}^{H'} = \neg \overline{b}_{a_i}^{H'}$

Algorithm 2.3.2: Improvisation process of binary-valued HSFS

4. Update HM: The quality of the newly formed feature subset is evaluated using an evaluation function $f(S)$. If the new harmony $S^{H'}$ is better than the harmony, which has the lowest evaluation result in the current HM, $S^{H'}$ then replaces this so-called worst harmony $S^{H^{worst}}$. This is symbolically equivalent to:

$$S^{H'} \in \text{HM} \wedge S^{H^{worst}} \notin \text{HM} \tag{2.37}$$

Otherwise, do nothing. Note that there may exist more than one harmony matching the lowest evaluation result; in this case, the harmony having most selected features is deemed the worst.

5. Check stopping criterion: If the number of improvisations reaches λ_{max} or the optimal feature subset in HM is not changed within a large number of iterations, the algorithm then stops. Otherwise, repeat steps 3 and 4.

2.3.2.2 Integer-valued HSFS

HS is conventionally used to solve optimisation problems of a fixed number of variables. However for FS, the size of feature subsets varies. In fact, the size of the emerging subsets themselves should be reduced, while optimising their evaluation results. There is no direct means to employ HS to solve the FS problems. Therefore, a mapping scheme, such as those shown in Table 2.1, is devised in the original HSFS

[54] to make possible the use of HS for the FS problems. A musician is best described as a "feature selector", where the available features for the feature selectors are converted to musical notes for musicians. Each musician may vote for one feature to be included in the feature subset when such an emerging subset is being improvised. The harmony is then the combined vote of all musicians, indicating which features are being nominated.

Table 2.1: Concepts mapping from HS to FS

HS	Optimisation	FS
Musician	Variable	Feature Selector
Musician Note	Variable Value	Feature
Harmony	Solution Vector	Subset
Harmony Memory	Solution Storage	Subset Storage
Harmony Evaluation	Fitness Function	Subset Evaluation
Optimal Harmony	Optimal Solution	Optimal Subset

In HSFS, all possible features of a given dataset A form the range of musical notes available to each musician. Multiple musicians are allowed to choose the same feature while they may opt to choose none at all. For further understanding, three example harmonies and their derived feature subsets are described in Table 2.2. The harmony S^{H^1} represents a subset of six distinctive features: $S^1 = \{a_1, a_2, a_3, a_4, a_7, a_{10}\}$. S^{H^2} shows an overlapping selection from musicians m_{1-3}, and an abandoned choice (denoted as $a\_$) from m_6, representing a reduced subset $S^2 = \{a_2, a_3, a_{13}\}$. S^{H^3} signifies the feature subset $S^3 = \{a_2, a_4, a_6, a_{13}\}$, where $a_3 \rightarrow a_6$ indicates that m_4 originally voted for a_3, but was forced to change its choice to a_6 due to HMCR activation.

Table 2.2: Feature subsets encoding scheme

	m_1	m_2	m_3	m_4	m_5	m_6	Derived Subset S
S^{H^1}	a_2	a_1	a_3	a_4	a_7	a_{10}	$\{a_1, a_2, a_3, a_4, a_7, a_{10}\}$
S^{H^2}	a_2	a_2	a_2	a_3	a_{13}	$a\_$	$\{a_2, a_3, a_{13}\}$
S^{H^3}	a_2	$a\_$	$a_3 \rightarrow a_6$	a_2	a_{13}	a_4	$\{a_2, a_4, a_6, a_{13}\}$

Regarding the above mapping and conversion scheme, the steps of the integer-based HSFS algorithm are formally described as follows:

1. Initialise parameters: The parameters involved in the integer-based HSFS algorithm are the same as those in the original HS, including $|\mathbb{H}|$, HMCR, λ_{max} and M. In particular, PAR is no longer used. The underlying motivation of employing PAR is to fine-tune the selected values to their neighbouring values. It may help to improve solutions when the problem domain is real-valued. However, each integer value used here is just a feature index. Therefore, such neighbouring relations are not applicable between features. The number of feature selectors is equal to that of musicians M and set to $|A|$ in the original HSFS, where A is all available features of a given dataset.

2. Initialise HM: HM, a two-dimensional matrix is now a store of features rather than bits. The size of HM is also $|\mathbb{H}| \times |A|$. Each row contains a feature subset while each column contains the historical features selected by a specified feature selector. The HM then takes the following form:

$$\mathrm{HM} = \left[\begin{array}{cccc|c} a_1^{H^1} & a_2^{H^1} & \cdots & a_{|A|}^{H^1} & f(S^1) \\ a_1^{H^2} & a_2^{H^2} & \cdots & a_{|A|}^{H^2} & f(S^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_1^{H^{|\mathbb{H}|}} & a_2^{H^{|\mathbb{H}|}} & \cdots & a_{|A|}^{H^{|\mathbb{H}|}} & f(S^{|\mathbb{H}|}) \end{array}\right]$$

   where any unit $a_j^{H^i}$, $i = 1, 2, \cdots, |\mathbb{H}|$, $j = 1, 2, \cdots, |A|$ of HM takes a feature index a_i, $i = 1, 2, \cdots, |A|$ as its value, representing a selected feature. When the value of any unit $a_j^{H^i}$, $\overline{a}_j^{H^i}$ is null, it means that the j^{th} musician votes no feature for the i^{th} subset. Also, when $\overline{a}_j^{H^i} = \overline{a}_{j+1}^{H^i}$, it means the j^{th} and $(j+1)^{th}$ musicians choose the same feature for the i^{th} subset. A harmony then becomes a feature subset by removing overlapping and null indices. $f(S^i)$ is the feature subset evaluation result of S^i.

3. Improvise a new feature subset: A new feature subset $S^{H'} = \{\overline{a}_1^{H'}, \overline{a}_2^{H'}, \cdots, \overline{a}_{|A|}^{H'}\}$ is improvised according to the following rules:

$$\overline{a}_j^{H'} = \begin{cases} \overline{a}_j^{H'} \in A = \{a_1, a_2, \cdots, a_{|A|}\} & \text{if HMCR} \leq r \\ \overline{a}_j^{H'} \in \mathrm{HM}_j = \{\overline{a}_j^{H^1}, \overline{a}_j^{H^2}, \cdots, \overline{a}_j^{H^{|\mathbb{H}|}}\} & \text{if HMCR} > r \end{cases} \tag{2.38}$$

   where HM_j is a set of historical features selected by the j^{th} musician, stored in a column of HM. When the HMCR $\leq r$ event is activated, musicians will randomly choose a feature from all available features of a given dataset.

Otherwise, they will randomly select a feature from features that are stored in HM. For further understanding the integer-valued HSFS, the improvisation of a new harmony is depicted in Algorithm 2.3.3.

```
|M|: number of musicians (feature selectors)
S^{H'} = {a_1^{H'}, a_2^{H'}, ..., a_{|A|}^{H'}}: new integer-valued harmony
HM_j = {a_j^{H^1}, a_j^{H^2}, ..., a_j^{H^{|𝕀|}}}: historical features selected by j^th musician
A = {a_1, a_2, ..., a_{|A|}}: set of all possible features of a given dataset
for j = 1 to |M| do
    if Random ([0,1]) <HMCR then
        a_j^{H'} =Random (HM_j)
    else
        a_j^{H'} =Random (A)
```

Algorithm 2.3.3: Improvisation process of integer-valued HSFS

4. Update HM: The updating strategy of the integer-based HSFS algorithm is the same as that in the binary-based version. The worst feature subset is replaced with the newly generated subset, which is of higher quality regarding the subset size and the evaluation result:

$$S^{H'} \in \mathrm{HM} \wedge S^{H^{\mathrm{worst}}} \notin \mathrm{HM} \tag{2.39}$$

Otherwise, do nothing.

5. Check stopping criterion: The algorithm terminates when the iteration number reaches λ_{max} or the optimal feature subset in HM is not changed within a large number of iterations. Otherwise, repeat steps 3 and 4.

In addition, this original integer-based HSFS algorithm has been improved in the literature [54], where a refinement mechanism, which works by setting the number of musicians $|M|$ to the size of the current best-found feature subset $|S^{H^{best}}|$ after every certain harmony improvisation, is used to further reduce the size of subsets and a dynamic scheme is applied for iteratively adjusting the predefined parameters including $|\mathbb{H}|$ and HMCR in real-time.

2.4 Summary

In this chapter, the basic theory of FS is introduced regarding the approaches to evaluating feature subsets and strategies for searching features. Evaluation approaches developed in the FS-related literature are categorised into three classes, including: the filter [91], wrapper [120, 121], hybrid [101, 244], and embedded approaches [91]. Three scopes with respect to search strategies—exhaustive search, hill-climbing, and metaheuristic—are discussed. In particular, three popular metaheuristic-based FS techniques are reviewed in detail. They are the FS algorithms based on GA [7], ACO [107], and PSO [219], which have been used for comparative experimental evaluation.

This chapter also discussed two different frameworks for applying feature grouping techniques to FS and reviewed the existing approaches to feature grouping, as the methods proposed in Chapter 4 exploit feature grouping to implement the task of FS.

Moreover, as the HS algorithm has been used or improved for searching feature subsets in this thesis, its main principles and its application to FS (HSFS) were presented. Two versions of HSFS algorithms—so-called binary-valued HSFS and integer-valued HSFS—were then introduced in detail, including a discussion of their advantages and disadvantages.

Chapter 3

Self-Adjusting Harmony Search-based Feature Selection

In this chapter, a technique improved the original idea of feature selection with harmony search (HSFS [54]) is proposed. It includes three new mechanisms. Firstly, a feature grouping is introduced in order to produce a so-called restricted feature domain (RFD) for every single musician (that act as individual feature selectors in the algorithm). The use of RFDs is to limit the locally explorable solution domains (of individual musicians), allowing more informative features to be located more quickly, whilst also reducing the run-time memory requirement of the algorithm. Secondly, a harmony memory consolidation mechanism is developed, which allows musicians to exchange information on tentatively selected features locally, and helps identify and remove non-contributing musicians. As a result, the size of the musician group can be dynamically adjusted during the search. Thirdly, a pitch adjustment strategy is presented which mimics the pitch adjustment behaviour of instrumentalists. It is used by HSFS for fine tuning the emerging feature subsets. In such a scheme, a feature may be substituted by one of its similar features, which is determined by using a certain feature similarity measure. The formal description of these enhancements is given in Section 3.1, which is followed with experimental evaluation as presented in Section 3.2.

3.1 Self-Adjusting HSFS

The original HSFS algorithm, in spite of being easy to be implemented, relies on a limited set of basic procedures to improvise and search for good quality feature subsets. However, the algorithm can potentially be modified to better support FS. This section details three new components developed to enhance the performance of HSFS.

3.1.1 Restricted Feature Domain

In the original HSFS implementation, all musicians $m_i \in M, i = \{1, \ldots, |M|\}$ jointly use a single domain of values, which is the pool of all possible features of a given dataset A. The total number of features $|A|$ inevitably affects the rate at which musicians identify good quality features. The presence of less informative features or multiple duplicates of the same feature also reduces the likelihood of locating better features through harmony memory considering rate (HMCR) activation. As a result, the algorithm may potentially spend unnecessary iterations searching poor quality candidate solutions, and such emerging feature subsets will be discarded since they introduce no improvement to the harmony memory (HM).

A new concept termed "restricted feature domain" is proposed to remedy the aforementioned shortcoming. This mechanism restricts the value domain $\aleph_i$ for any given musician m_i to a selective subset of A. The RFDs are constructed during the initialisation phase and reconstructed when the number of musicians is adjusted during the iteration phase. Hence, the recombination of RFDs dynamically affects the choice of musicians throughout the search process. Of course, the union of these RFDs should be equivalent to the full set of features: $\bigcup_{i \in \{1,2,\ldots,|M|\}} \aleph_i = A$ in order to ensure that no important features are mistakenly left out. The feature distribution amongst all the musicians should also be random but uniform. The cardinality of $\aleph_i$ is thus devised to be controlled by a restricted ratio δ, $0 < \delta \leq 1$, such that $|\aleph_i| = \lceil \delta \cdot |A| \rceil$. The operator $\lceil\ \rceil$ indicates it takes the ceiling integer of a real number.

A set of RFDs can be generated through various methods, so long as the desired properties that are described above are satisfied. Additionally, if problem domain-specific information is available (e.g., provided by human experts), RFDs may also be populated or adjusted in favour of better quality features. At the current stage, for simplicity, an RFD is empirically generated by randomly removing features until

$|\aleph_i| = \lceil \delta \cdot |A| \rceil, \delta = 0.8$, while maintaining a full coverage of features across all musicians. The pseudo code of the suggested mechanism is given in Algorithm 3.1.1 where `Random( )` is an operator randomly taking an element from a set of features.

A: full set of features
M: group of musician $m_i \in M, i = 1, 2, \cdots, |M|$
HMCR: harmony search considering rate
$|\mathbb{H}|$: harmony memory size
$S^{H'}$: new harmony being improvised
$\text{HM}_i = \bigcup_{j=1}^{|\mathbb{H}|} \overline{a}_i^{H^j}$: the i^{th} column of HM, indicating note domain of musician m_i
$\aleph_i = A$: RFD of musician m_i
for $i = 1$ **to** $|M|$ **do**
    while $|\aleph_i| > \lceil \delta \cdot |A| \rceil$ **do**
        $\aleph_i = \aleph_i \setminus \texttt{Random}\ (\aleph_i)$
while $\bigcup_{i \in \{1,2,\cdots,|M|\}} \aleph_i \neq A$ **do**
    for $i \leftarrow 1$ **to** $|M|$ **do**
        $\aleph_i = \aleph_i \setminus \texttt{Random}\ (\aleph_i)$
        $\aleph_i = \aleph_i \cup \texttt{Random}\ (A \setminus \bigcup_{i \in \{1,2,\cdots,|M|\}} \aleph_i)$
$S^{H'} = \emptyset$
for $i = 1$ **to** $|M|$ **do**
    if `Random` $([0, 1]) < HMCR$ **then**
        $S^{H'} = S^{H'} \cup \texttt{Random}\ (\text{HM}_i)$
    else
        $S^{H'} = S^{H'} \cup \texttt{Random}\ (\aleph_i)$
return $S^{H'}$

Algorithm 3.1.1: HSFS with RFD

3.1.2 Self-Configuration of the Musician Size

The main challenge for FS is to effectively reduce the size of candidate feature subsets, while maintaining their original semantics. The iterative refinement procedure employed by the original HSFS aims to address this issue, to a certain extent, by its flexible mapping of musical concepts onto their associated elements in FS. In particular, a musician is not tied to a specific feature, thereby becoming an independent, single-feature-selector. This sharply contrasts with many alternative methods that rely on binary-valued feature subset representation.

The iterative refinement procedure first initialises the number of musicians to be the size of the complete set of original features $|M| = |A|$. This avoids the

configuration of $|M|$ to be manually defined by human, and that the feature subset size may be adjusted according to the actual amount of redundancy present in the data. $|M|$ is then iteratively reduced (in so doing, the size of feature subsets to be selected is restricted) until no smaller solution can be found without sacrificing the evaluation score. As such, although effective, this procedure leads to repetitive executions of the entire search process, and the earlier executions may also over-restrict the search process to a sub-optimal solution region.

```
1 if |M| ≠ max_{j∈{1,2,⋯,|ℍ|}} |S^j| then
2     |M| = max_{j∈{1,2,⋯,|ℍ|}} |S^j|
3     for j = 1 to |ℍ| do
4         if |M| < |S^{H^j}| then
              // Consolidation
5             while |M| < |S^{H^j}| do
6                 S^{H^j} = S^{H^j} \ {a_}
7         else
              // Expansion
8             while |M| > |S^{H^j}| do
9                 S^{H^j} = S^{H^j} ∪ {a_}
```

Algorithm 3.1.2: Process of HMC

The self-adjusting HSFS algorithm proposed herein embeds an alternative procedure to the aforementioned. It attempts to dynamically and naturally adjust $|M|$ throughout a single execution of the search, via a means of identifying and eliminating potentially non-contributing musicians (with their note domains fully filled by duplicated nominations or discarded votes $a\_$). This procedure is referred hereafter as harmony memory consolidation (HMC). The pseudo-code of HMC is given in Algorithm 3.1.2. In particular, to better determine the presence of non-contributing musicians, the following process needs to be performed:

1. The duplicating nominations within each of the harmonies stored in the harmony memory are replaced by $a\_$.

2. The desirable value of $|M|$ may then be derived using the formula given below:

$$|M| = \max_{j\in\{1,2,\cdots,|\mathbb{H}|\}} |\{a_i | a_i \in S^{H^j}, a_i \neq a\_\}| + 1. \tag{3.1}$$

Alternatively, $|M|$ may be determined by first converting harmonies $S^{H^j},\ j = 1, 2, \cdots, |\mathbb{H}|$ into feature subsets S^j, and then computing:

$$|M| = \max_{j \in \{1, \cdots, |\mathbb{H}|\}} |S^j| + 1. \tag{3.2}$$

3. The existing harmonies are trimmed by randomly removing $a_- \in S^{H^j}$, until $|S^{H^j}| = |M|,\ j = 1, 2, \cdots, |\mathbb{H}|$.

4. The normal HSFS improvisation process is then resumed, on the basis of the newly consolidated harmony memory.

Table 3.1: Consolidation of harmony memory

Iteration	Harmony Memory						
k	a_1	a_-	a_2	a_-	a_1	a_2	a_3
	a_2	a_4	a_5	a_-	a_-	a_5	a_5
k (HMC)	a_1	a_-	a_2	a_-	a_-	a_-	a_3
	a_2	a_4	a_5	a_-	a_-	a_-	a_7
$k+1$	a_1	a_2	a_-	a_-	a_3		
	a_2	a_4	a_5	a_-	a_7		

An illustrative example is given in Table 3.1. The initial harmony memory (at iteration k) consists of several duplicate and discarded nominations, which are identified during the HMC process. For instance, feature subset $(a_1, a_-, a_2, a_-, a_1, a_2, a_3)$ may be changed into $(a_1, a_-, a_2, a_-, a_-, a_-, a_3)$ in this given example. The number of musicians is then reduced to $|M| = \max(3, 4) + 1 = 5$, and the respective harmonies are also trimmed. The resultant harmony memory after consolidation is then used for the next iteration $(k+1)$.

Table 3.2: Expansion of harmony memory

Iteration	Harmony Memory				
k	a_1	a_2	a_3	a_-	
	a_2	a_7	a_4	a_5	
$k+1$	a_1	a_-	a_3	a_-	a_2
	a_-	a_7	a_4	a_5	a_2

Note that the HMC procedure may also be utilised to facilitate the expansion of the musician group because the value of $|M|$ is determined with respect to the

size of the largest feature subset in HM as shown in Eq. 3.2. The rationale behind such a mechanism is to allow larger feature subsets to be nominated, if they may lead to a potentially higher (overall) quality solution. Table 3.2 details an example of harmony memory expansion where the harmony memory at iteration k contains a candidate feature subset of size $|S| = |M|$. It is probable that there exists more informative feature subsets of size $|S'| > |M|$ and, therefore, the size of the musician group is enlarged by inserting $a_{\_}$ at random positions for all $S^H \in \text{HM}$.

3.1.3 Feature Subset Adjustment using Feature Similarity Measures

The base HS algorithm involves another parameter termed "pitch adjustment rate" (PAR). It offers a stochastic mechanism that allows a note chosen by a given musician to be shifted to a similar one. The underlying motivation for this mechanism is that minor adjustments into adjacent values may help discover better quality solutions, which is generally true for real-valued optimisation problems. PAR, in conjunction with δ, ensures that fine adjustments can be made to an emerging solution, and the solution region may be sufficiently explored.

The original HSFS algorithm does not exploit the benefits offered by PAR. This is because, now that the values represent feature indices, each feature and its neighbours may not have such general relation, and, therefore, an adjustment will result in a change to a possibly unrelated feature nearby. However, the absence of PAR hinders the strength of the algorithm in terms of its effectiveness for finding good quality feature subsets.

Having recognised this, the pitch adjustment mechanism is re-introduced in this improved HSFS approach together with a method to determine neighbouring features. These neighbours are features similar to one another. In the context of FS, the use of the parameter PAR, ($0 \leq \text{PAR} \leq 1$), will enable a musician to choose a neighbouring feature. Such a feature bears a similarity with the original features within the range calculated on the basis of the formula $(1-\omega)+Rand(2\times\omega)$, where ω is the fret width [80] that constrains the maximal amount of dissimilarity allowed. Note that $(1-\text{PAR})$ denotes the possibility of using the chosen value without further alteration. Also, as suggested in the original HS algorithm, the pitch adjustment procedure and HMCR activation are set to two mutually exclusive events so that

further adjustment is only carried out when a feature is selected from within the harmony memory.

To determine the neighbouring features, a number of existing feature similarity measurements may be employed. For example, non-parameter test-based approaches such as the Walds-Wolfowitch test [112] may be used to detect the closeness of probability distributions of the variables. However, such measures are sensitive to both the location and the dispersion of the distributions [20, 148], and, therefore, they may not be suitable for measuring the similarity of features in arbitrary datasets. Alternatively, the dependency between the features may be utilised to calculate the degree of feature similarity. In order to obtain the degree of similarity between any paired features, two existing dependency measures, both linear (*correlation coefficient-based* [155]) and non-linear (*fuzzy-rough set-based* [111]) are used.

3.1.3.1 Correlation Coefficient-Based Feature Similarity

Correlation coefficient is a common linear method for measuring the degree of similarity between two random variables. Correlation coefficient ρ between two random variables x and y is formulated as:

$$\rho(x,y) = \frac{\sigma_{xy}}{\sigma_x \sigma_y} \tag{3.3}$$

$$\sigma_{xy} = \frac{1}{n-1}(\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}) \tag{3.4}$$

where σ_x and σ_y signify the variance of a variable and σ_{xy} the covariance between two variables. If x and y are entirely correlated, then a purely linear relationship exists and $\rho(x,y)$ is ± 1. Independence of x and y implies $\rho(x,y) = 0$. Hence, the quantity can be used as a measure of similarity between two features [100] with the following properties:

1. $|\rho(x,y)| \leq 1$.
2. $\rho(x,y) = 0$ if and only if x and y are linearly correlated.
3. $\rho(x,y) = \rho(y,x)$.
4. If $x^* = ax + b$ and $y^* = cy + d$ for certain constants a, b, c, d, then $\rho(x^*, y^*) = \rho(x,y)$, implying that the similarity measure is not affected by rescaling and transformation of variables.

The correlation coefficient may be sufficient for the strength of feature similarity, but it makes a strong assumption on linear and highly dependent relationships between features. Non-linear patterns are neglected. For example, assume there exists a quadratic relationship between the value of x and y ($x = y^2$), and y is evenly distributed in the range of $[-1, 1]$. The resulting correlation coefficient is $\rho(x, y) = 0$, indicating that x and y are independent. However, there in fact exists a strong dependency between them. So an alternative approach may be necessary.

3.1.3.2 Fuzzy Rough Set-Based Feature Similarity

Fuzzy-rough set theory [47, 177] is an extension of traditional rough set theory [167] where there are two sets and the lower and upper approximation are defined using fuzzy notions [61]. A rough set is centred upon crisp information granulation. In the crisp case, elements either belong to the lower approximation with absolute certainty or not at all. In the fuzzy-rough case, elements may have a membership in the range $[0, 1]$, allowing greater flexibility in handling uncertainty. Given a vague concept C, the fuzzy lower and upper approximations for S are defined as:

$$\mu_{\underline{R_S}C}(x_i) = \inf_{x_j \in X} I(\mu_{R_S}(x_i, x_j), \mu_C(x_j)) \tag{3.5}$$

$$\mu_{\overline{R_S}C}(x_i) = \sup_{x_j \in X} T(\mu_{R_S}(x_i, x_j), \mu_C(x_j)) \tag{3.6}$$

where I is a fuzzy implicator and T a t-norm. $\mu_C(x_j)$ is the membership of instance x_j belonging to C. R_S is the fuzzy similarity relation induced by the subset of features S:

$$\mu_{R_S}(x_i, x_j) = T_{a \in S}\{\mu_{R_a}(x_i, x_j)\} \tag{3.7}$$

with $\mu_{R_a}(x_i, x_j)$ being the weight of similarity between instance x_i and x_j for feature a. The following three common fuzzy similarity models can be constructed for this purpose.

$$\mu_{R_a}(x_i, x_j) = 1 - \frac{|a(x_i) - a(x_j)|}{|a_{\max} - a_{\min}|} \tag{3.8}$$

$$\mu_{R_a}(x_i, x_j) = \exp\left(-\frac{(a(x_i) - a(x_j))^2}{2\sigma_a^2}\right) \tag{3.9}$$

$$\mu_{R_a}(x_i, x_j) = \max\left(\min\left(\frac{(a(x_j) - a(x_i) + \sigma_a)}{\sigma_a}, \frac{(a(x_i) - a(x_j) + \sigma_a)}{\sigma_a}\right), 0\right) \tag{3.10}$$

where σ_a and σ_a^2 are the standard deviation and the variance of the values of feature a taken by all instances in X respectively, $a_{\max}$ is the maximum value under the feature a, and $a_{\min}$ is the minimum value. The choice of measurement of the fuzzy similarity relation has a significant impact on the resultant fuzzy partitions, and thus the subsequently selected feature subsets.

Regarding the fuzzy lower approximation defined in Eqn. 3.5, the fuzzy positive region, which contains all instances of X that can be classified into classes of X/Q, is calculated as follows:

$$\mu_{POS_{R_P}(Q)}(x) = \sup_{C \in X/Q} \mu_{\underline{R_P C}}(x) \tag{3.11}$$

The resulting degree that a set of features Q depends on another set of features P is denoted as $P \Rightarrow Q$ and is defined as:

$$\gamma_P(Q) = \frac{\sum_{x \in X} \mu_{POS_{R_P}(Q)}(x)}{|X|} \tag{3.12}$$

Since the proposed approach only concerns similarity measurements between two arbitrary single features, say, a_i and $a_j \in A$. Eq. 3.12 may be re-written as:

$$\gamma(a_i, a_j) = \frac{\sum_{x \in X} \mu_{POS_{R_{\{a_i\}}}(\{a_j\})}(x)}{|X|} \tag{3.13}$$

representing the dependency degree of feature a_i upon a_j. This formula is used to locate the suitable closest and most dependent neighbouring feature. For clarification, Algorithm 3.1.3 shows the pseudo-code of the improvisation process using pitch adjustment.

3.1.4 Self-Adjusting HSFS

The procedure for the complete self-adjusting HSFS algorithm is illustrated in Fig. 3.1. It incorporates three new modules: RFD construction, harmony memory consolidation, and PAR-based feature subset improvisation.

The complexity of the HSFS algorithm is analysed below. The initialisation requires $O(|M| \times |\mathbb{H}|)$ operations to randomly populate the subset storage, and the improvisation process is of the order $O(|M| \times k_{\max})$ because every feature selector needs to produce a new feature at every iteration. Here, $|\mathbb{H}|$ is the subset storage size, $|M|$ is the number of feature selectors, and $\lambda_{\max}$ is the maximum number of iterations.

$S^{H'}$: new harmony being improvised
$\text{HM}_i = \bigcup_{j=1}^{|\mathbb{H}|} \overline{a}_i^{H^j}$: the i^{th} column of HM, indicating note domain of musician m_i
$\aleph_i$: RFD of musician m_i
$\gamma(a_p, a_q)$, feature similarity measure described in Eqn. 3.13
for $i = 1$ **to** $|M|$ **do**
    if `Random` *([0,1])* $<$ *HMCR* **then**
        $a_p = \texttt{Random}\ (\text{HM}_i)$
        if `Random` *([0,1])* $<$ *PAR* **then**
            `// Δ is a random degree of the similarity between features`
            $\Delta = 1 - \omega + \texttt{Random}\ (2\omega)$
            $a_q = \underset{a_q \in A, a_q \neq a_p}{\text{argmin}} |(\gamma(a_p, a_q) - \Delta)|$
            $S^{H'} = S^{H'} \cup \{a_q\}$
        else
            $S^{H'} = S^{H'} \cup \{a_p\}$
    else
        $S^{H'} = S^{H'} \cup \{\texttt{Random}\ (\aleph_i)\}$
return $S^{H'}$

Algorithm 3.1.3: Improvisation process using PAR

The construction of restricted feature domains requires $O(|M| \times \lambda_{\max} \times \lceil \delta \times |A| \rceil)$, which occurs in both the initialisation and iteration phases when the number of musicians is changed. The HMC procedure has a computational complexity of $O(|\mathbb{H}| \times |M|)$ since all stored harmonies need to be examined and consolidated. The additional overhead introduced by the PAR process is largely due to the feature similarity matrix construction, which incurs a cost of up to $O(|A|^2 \times |X|^2)$, where $|X|$ is the total number of times instances in dataset are trained.

Thus, the overall algorithm complexity (including the initial cost of computing the feature similarity values) is:

$$O(|A|^2 \times |X|^2) + O(|M| \times (|\mathbb{H}| + \lceil \delta |A| \rceil) \times \lambda_{max}) \tag{3.14}$$

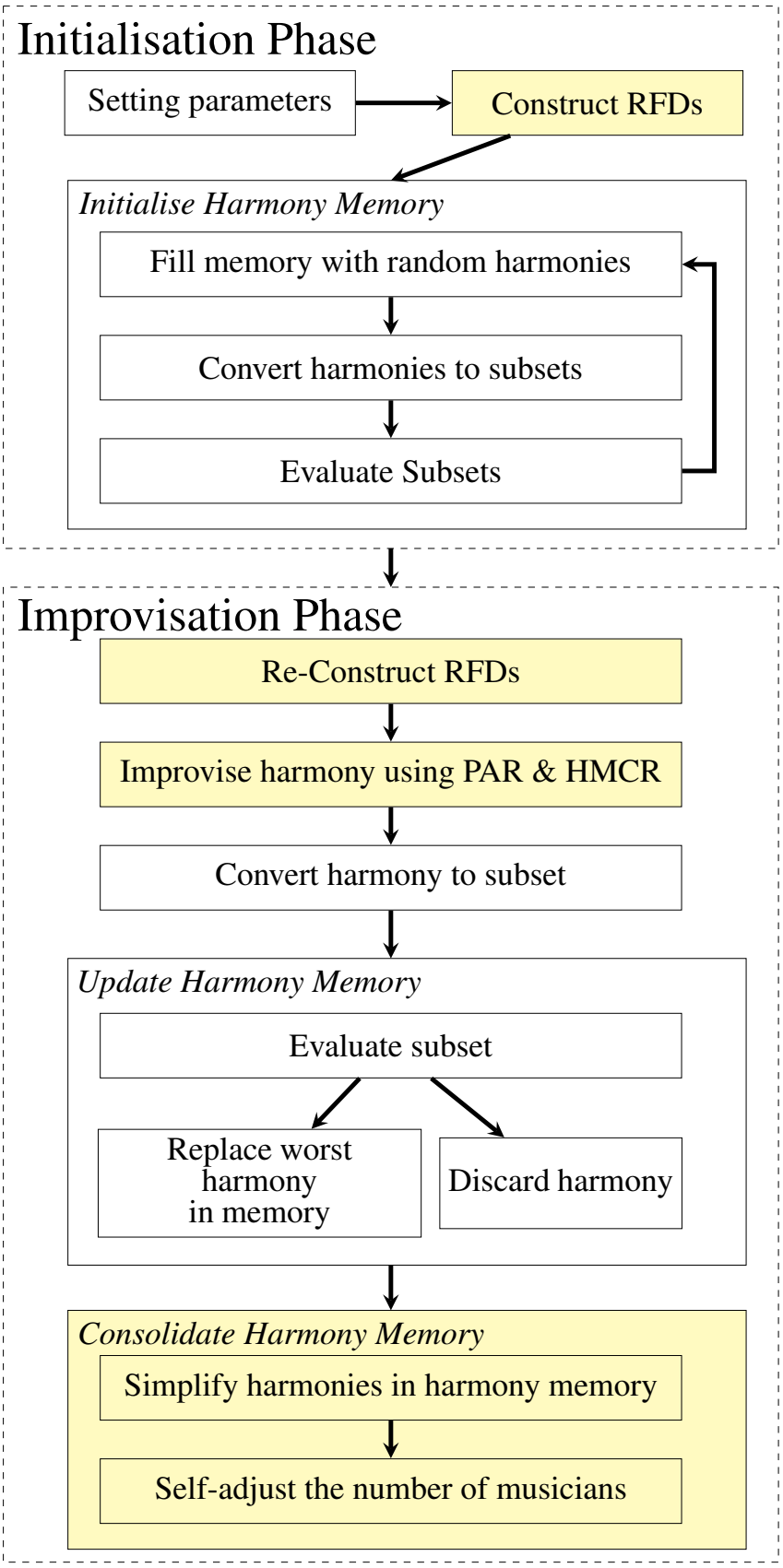

Figure 3.1: Proposed self-adjusting HSFS algorithm

3.2 Experimentation and Discussion

In this section, the results of experimental investigations are reported to demonstrate the capabilities and characteristics of the proposed improvements. Notationally, in the results HSFS_{SA} stands for the whole algorithm that incorporates all of the newly proposed mechanisms. Section 3.2.1 presents a comparison with other search methods, including greedy hill-climbing (GHC), the original HSFS (HSFS_{O}) and two nature-inspired optimisation techniques: genetic algorithms (GAs) [7] and particle swarm optimisations (PSO) [219]. Two filter-based feature subset evaluators with the evaluation results $f(S)$, $0 \leq f(S) \leq 1$, the correlation-based (CFS) evaluator and the probabilistic consistency-based (PCFS) evaluator, are employed. The experimentation uses a total of 10 real-valued UCI benchmark datasets [73], several of which are of high dimensionality and/or contain a large number of objects, thereby presenting reasonably realistic challenges for the proposed techniques. A summary of these datasets is provided in Table 3.3.

The parametric settings used for the investigated algorithms are those that have been constantly employed in the recent related research. In particular, the parameter of the maximum number of generations/iterations λ_{max} of all the stochastic search methods—including HSFS_{SA}, HSFS_{O}, GAs and PSO—is set to 5000 in order to prevent them from being prematurely terminated. Also, these stochastic search methods are population-based, which require configuring the population of solutions P properly. The number of populations $|P|$, strictly speaking, is therefore set to 20 for all of them. The other inherent parameters of the algorithmic mechanisms are treated as follows: both weighting constants c_1 and c_2 of PSO are equal to 2; the crossover rate r_c and the mutation rate r_m of GA are set to 0.6 and 0.033 respectively; the harmony memory considering rate HMCR=0.8 and the number of musicians $|M| = |A|$ regarding HSFS_{O}; and the pitch adjustment rate PAR=0.8 and the HMCR=0.8 with respect to HSFS_{SA}.

Note that the C4.5 algorithm [176] is adopted due to its popularity as a verifier of the quality of the selected feature subsets from an end classifier learner's perspective, where stratified ten-fold cross-validation is exploited for accuracy validation.

3.2.1 Comparison with Alternative Search Strategies

Stratified tenfold cross-validation (10-FCV) is used. For a given dataset, it works by dividing the data into ten sub-tables. Nine of these ten sub-folds are employed for

Table 3.3: Dataset information

Dataset	Features	Objects	Decisions	C4.5
arrhythmia	280	452	13	64.38
handwritten	257	1593	10	75.83
ionosphere	35	230	2	87.83
libras	91	360	15	69.72
multifeat	650	2000	10	94.75
secom	591	1567	2	89.60
segment	20	1500	7	95.73
sonar	61	208	2	71.15
cnae	857	1080	9	88.8
web	2557	149	5	51.6

training, where FS is employed to learn the optimal feature subset. The remaining single fold is used to test the classifier using the selected feature subset. This process is then repeated ten times. Therefore, each fold is used for testing only once and the stratification of the data ensures that each class label has the same representation in all folds, thereby helping to alleviate bias/variance problems [19]. In the experiments, 10-FCV is performed using ten different random folds of the data in order to lessen the impact of random factors within the heuristic algorithms. These 10×10 sets of evaluations are aggregated to produce the final experimental outcomes.

In addition, a paired t-test with two-tailed $p = 0.01$ has been performed in order to compare the statistical differences between results obtained by HSFS_{O} and HSFS_{SA}, as given in Table 3.4 and Table 3.5. The symbol '*b*' indicates that HSFS_{SA} obtains a better result than HSFS_{O}, '=' denotes that there exists no statistical difference between the results, and '*w*' signifies that HSFS_{SA} results in a statistically worse search performance. These comparisons are made in terms of whether the generated feature subsets offer a higher evaluation score, a smaller subset size and/or better classification accuracy.

Using the CFS evaluator to obtain results as reflected in Table 3.4, higher evaluation scores are obtained using the proposed techniques for six of the ten datasets (indicated with *v*) when compared to HSFS_{O}. For the remaining four datasets—*ionosphere*, *secom*, *segment*, and *sonar*—more compact subsets are discovered with equal evaluation scores (or with a tiny difference in evaluation score). The bold figures signify improved performance when compared with GHC, where HSFS_{SA}

Table 3.4: Comparison of FS performance with different subset search methods using CFS, regarding average subset size, evaluation score, and classification accuracy (%). b, $=$, and w indicate statistically better, equal, and worse results respectively, when compared to the original algorithm ($HSFS_O$). Bold figures signify the overall best results obtained for each of the datasets.

	$HSFS_{SA}$			$HSFS_O$			GA			PSO			GHC		
	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)
arrhythmia	**12.0**(b)	**0.449** (b)	77.78 (b)	22.9	0.441	62.89	46.2	0.397	67.04	13.7	0.288	64.39	25	0.438	68.37
handwritten	**29.3**(b)	**0.524** (b)	76.39 (b)	63.8	0.51	75.45	93.9	**0.523**	75.58	127.3	0.472	76.58	75	**0.524**	75.95
ionosphere	**8.7** (b)	**0.536** (=)	89.56 (b)	11.1	0.534	88.43	10.3	**0.539**	86.96	10.3	0.535	90.43	11	0.521	86.96
libras	**15.4**(b)	**0.613** (b)	68.00 (=)	27.3	0.604	68.04	30.5	**0.611**	68.06	26.5	0.592	66.11	26	0.608	66.67
multifeat	**83.1**(b)	0.920 (b)	95.12 (b)	81.6	0.906	93.50	217.9	0.894	95.15	315.2	0.854	94.15	144	**0.925**	95.40
secom	**15.2** (b)	**0.096** (=)	90.44(b)	36.8	**0.096**	90.11	51.2	0.012	90.81	27.5	0.013	91.90	17	**0.096**	92.98
segment	**4.5** (b)	**0.734**(=)	94.80 (=)	5.1	**0.732**	94.80	**4.4**	**0.732**	95.13	**4.4**	0.735	95.00	6	0.725	95.13
sonar	**9.4** (b)	0.352(=)	76.19 (b)	17.9	0.352	73.72	17.5	**0.359**	73.50	11.4	0.327	65.31	19	0.352	73.50
cnae	14.4(b)	**0.426 (b)**	77.80(b)	18.6	0.418	69.80	170.9	0.037	85.74	**10.8**	0.073	52.13	28	0.414	75.56
web	**33.3**(b)	0.543 (b)	57.03(b)	35.4	0.526	55.33	96	0.327	57.00	100.9	0.129	43.76	59	**0.565**	59.71

Table 3.5: Comparison of FS performance with different subset search methods using PCFS, regarding average subset size, evaluation score, and classification accuracy (%). b, $=$, and w indicate statistically better, equal, and worse results respectively, when compared to the original algorithm ($HSFS_O$). Bold figures signify the overall best results obtained for each of the datasets.

	$HSFS_{SA}$			$HSFS_O$			GA			PSO			GHC		
	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)	Size	Eval.	C4.5(%)
arrhythmia	29.1 (b)	**0.989**(=)	67.59 (b)	136.2	**0.989**	64.16	29.2	**0.989**	67.04	112.5	**0.989**	64.62	**21**	0.988	68.61
handwritten	70.2 (b)	**1.000** (=)	79.38 (b)	80	**1.000**	75.07	33	**1.000**	67.10	23.5	**1.000**	63.08	**18**	**1.000**	71.93
ionosphere	**6.8** (b)	**0.997** (=)	87.10(b)	21.5	**0.997**	86.70	10	**0.997**	83.91	8.9	**0.997**	84.78	**7**	0.996	90.43
libras	**16.4**(b)	**0.978** (b)	68.75 (b)	57.2	0.974	69.33	17.3	0.974	66.11	28.2	0.973	69.17	17	0.969	66.94
multifeat	9.1 (b)	**1.000** (=)	95.58 (b)	26.2	**1.000**	94.35	43	**1.000**	85.75	12.2	**1.000**	83.85	**7**	**1.000**	89.95
secom	23.7 (b)	**0.990** (=)	89.42(b)	154.9	**0.989**	88.70	92.2	**0.989**	90.24	284.8	**0.990**	89.66	**1**	0.936	92.98
segment	**6.4** (b)	**0.998** (=)	95.91(b)	8.5	0.997	95.6	7.0	0.997	95.47	7.5	0.997	95.47	9	**0.998**	95.60
sonar	**11.7**(b)	**1.000** (b)	74.72(b)	14.5	0.989	73.2	**12.0**	0.989	73.07	18.5	0.989	74.02	14	0.981	74.52
cnae	**65.2** (b)	**0.983** (=)	85.19 (b)	69	**0.981**	76.85	213.9	**0.983**	88.70	699.1	**0.983**	88.33	67	**0.981**	86.76
web	**18.6** (b)	**1.000** (=)	62.22 (b)	21.1	**1.000**	54.45	1036.8	**1.000**	53.00	585.7	**1.000**	50.95	21	**1.000**	60.38

Table 3.6: Comparison of execution time (millisecond) between different subset methods using CFS.

	$HSFS_{SA}$	$HSFS_{O}$	GA	PSO	GHC
arrhythmia	1792	1138	5418	1371	127
handwritten	1650	954	8044	3416	1807
ionosphere	46	37	180	195	11
libras	817	152	984	435	26
multifeat	9413	5587	59173	27802	10460
secom	6754	4557	28847	16035	923
segment	26	17	80	132	13
sonar	76	80	412	290	16
cnae	155363	139857	65066	23613	1300
web	97512	135671	169760	27068	15879

Table 3.7: Comparison of execution time (millisecond) between different subset methods using PCFS.

	$HSFS_{SA}$	$HSFS_{O}$	GA	PSO	GHC
arrhythmia	4202	4336	34957	4514	912
handwritten	155	140	127	5463	4877
ionosphere	444	394	85	334	30
libras	1198	1234	11559	1099	190
multifeat	424	407	222	10535	5037
secom	29084	30858	291034	30571	2372
segment	3044	2710	2152	597	84
sonar	552	534	5132	579	52
cnae	122361	106798	324843	52925	39711
web	101459	94431	778	26575	1975

enables further reduction of the size of the selected feature subsets. Although the GAs are capable of identifying higher quality subsets for several datasets, they are unable to identify good quality solutions for the higher dimensional datasets such as *arrhythmia*, *multifeat*, *secom*, *cnae*, and *web*. Importantly, the classification accuracy of the classifier learners built using the reduced feature subsets (selected by $HSFS_{SA}$) is improved, particularly for *arrhythmia* (by 13.40%), *sonar* (by 5.04%), and *web* (by 5.43%).

For Table 3.6 and Table 3.7, note that the execution time of $HSFS_{SA}$ is longer than $HSFS_{O}$ for most of the datasets except *sonar* and *web*, which is caused by

the complexities incurred by the introduction of self-adjusting components. This observation confirms the complexity analysis performed in Section 3.1.4.

Employing the PCFS evaluator to obtain results as reflected in Table 3.5, the performance improvement over the original algorithm is more obvious, where better subsets are selected across all datasets. Note that only a single feature is selected by GHC for the *secom* dataset. This is because no additional features offer any increase in the evaluation score, when combined with this selected feature. However, better combinations of features do exist, which are successfully identified by all other employed stochastic approaches. Feature subsets obtained by GAs are mostly comparable to the others in terms of the evaluation scores. GAs are indeed able to identify feature subsets of optimal quality for the dataset *libras*. The classifiers built using the feature subsets selected by HSFS_{SA} also show improved classification performance for six of the ten datasets. Interestingly, higher classification accuracies are obtained by HSFS_{SA} for nine of the whole ten datasets when compared to HSFS_{O}. For the remaining dataset, *libras*, the minor differences in accuracy are acceptable, given the substantial reduction in the averaged feature subset size (16.40 for HSFS_{SA}, and 57.23 for HSFS_{O}).

3.2.2 Effect of Individual Strategy

Two of the datasets with the largest number of features, *multifeat* and *secom*, are employed for the remainder of the experimentation. Each of the proposed modifications to HSFS is tested on its own in order to ascertain what benefits these strategies offer individually. Fig. 3.2 illustrates the effects of the restricted ratio δ which controls the size of the RFDs. The results are collected using different values for δ ranging from 0.1 to 1, with an interval of 0.1. Intuitively, if the value of δ is too small, the musicians may have too few features to work with, which will limit the solution quality. For both datasets, the quality of the selected subsets approach peaks at $\delta = 0.8$, when 80% of features in A are utilised.

Fig. 3.3 aims to demonstrate the effectiveness of the HMC process. The algorithm successfully adjusts the number of musicians to a reasonable level, down from the initial setting of $|M| = |A|$, without subjective intervention. The reduced group size further encourages the remaining feature selectors to identify even smaller candidate solutions. The compactness of the resulting subsets as reported in Table 3.4 and Table 3.5 also provides good evidence for the positive impacts of this process.

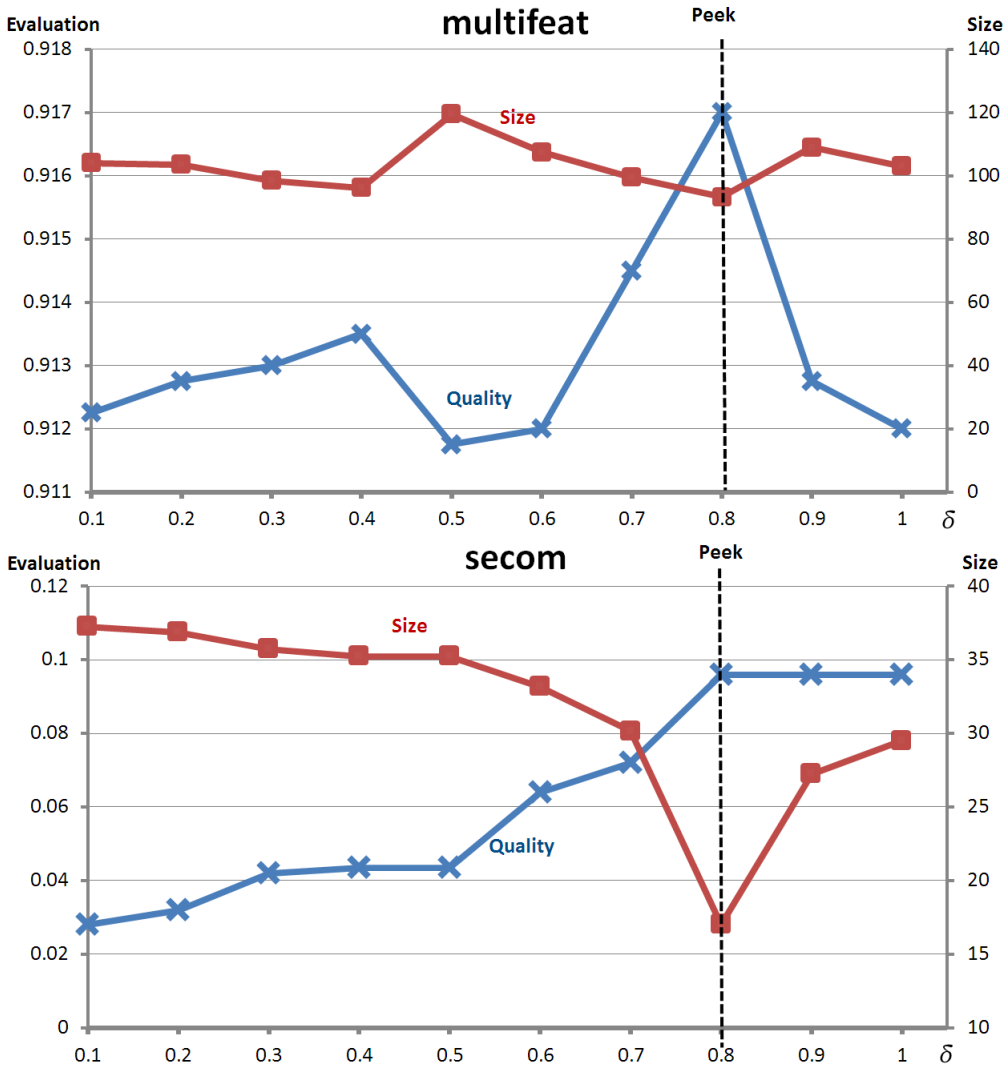

Figure 3.2: Demonstration of the effects of RFD using different $\delta \in [0.1, 1]$

The final set of experiments is carried out in order to study the effects of the pitch adjustment rate, by varying ω from 0 to 1 with an interval of 0.1. This parameter manipulates the shift ratio, where a feature is replaced with a random close neighbouring feature. Fig. 3.4 shows the size and evaluation scores of the best solutions recorded during the search process. The performance of both datasets, *multifeat* and *secom*, peaks at $\omega = 0.2$, when a feature shifts to neighbouring features with a probability of 20%. Obviously, if the value of ω is set too large, good quality features may be neglected.

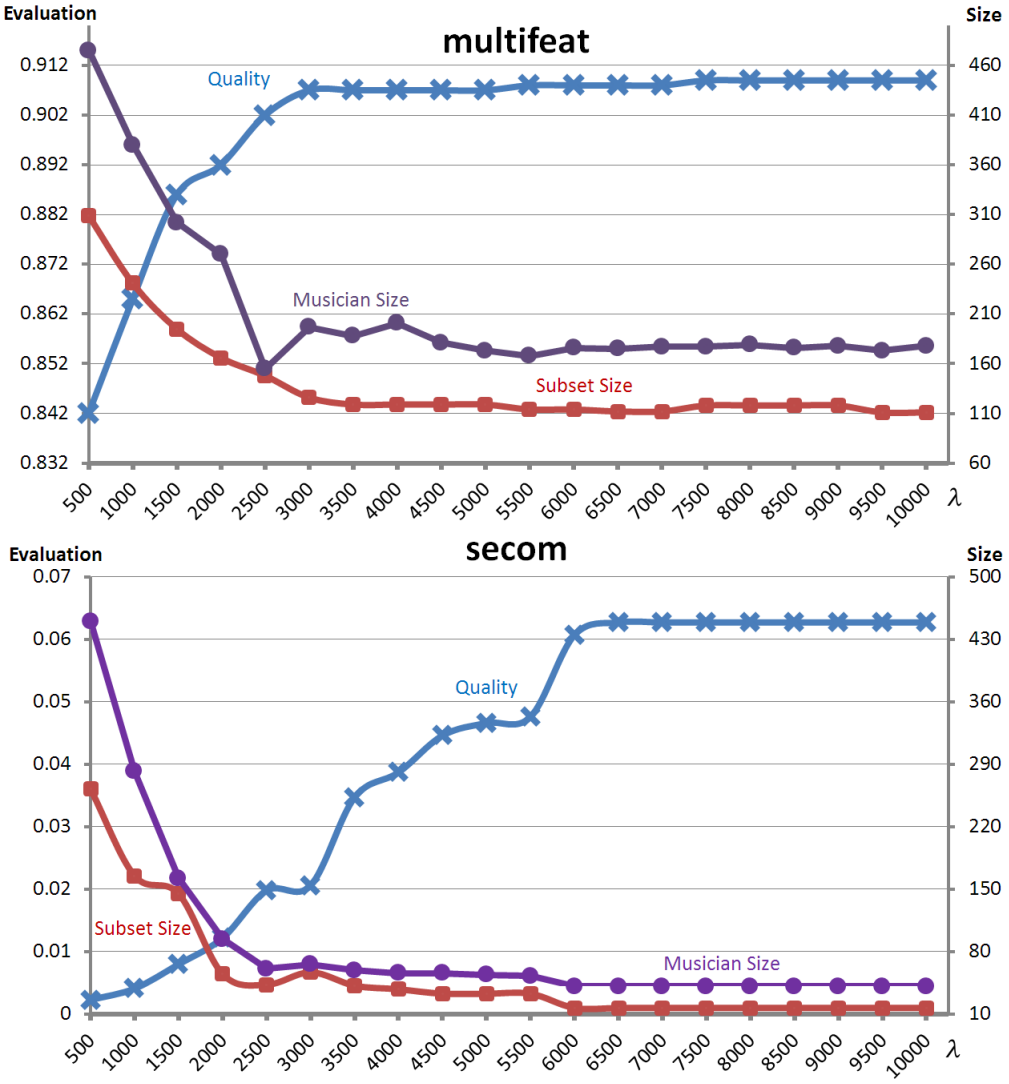

Figure 3.3: Automatic configuration of musician size using HMC

3.3 Summary

This chapter has presented a self-adjusting FS search algorithm which improves the original harmony search-based feature selection [53] with three new techniques. The proposed techniques are conceptually simple and require limited computational overheads in order to achieve positive effects. The musicians in $HSFS_{SA}$ improvise new candidate feature subsets using a selective portion of the full set of original features (RFD), and dynamically adjust their subset size via the HMC process. The pitch adjustment strategy, re-introduced to HSFS via feature similarity measures allows finer yet relevant adjustments to emerging feature subsets. Experimental

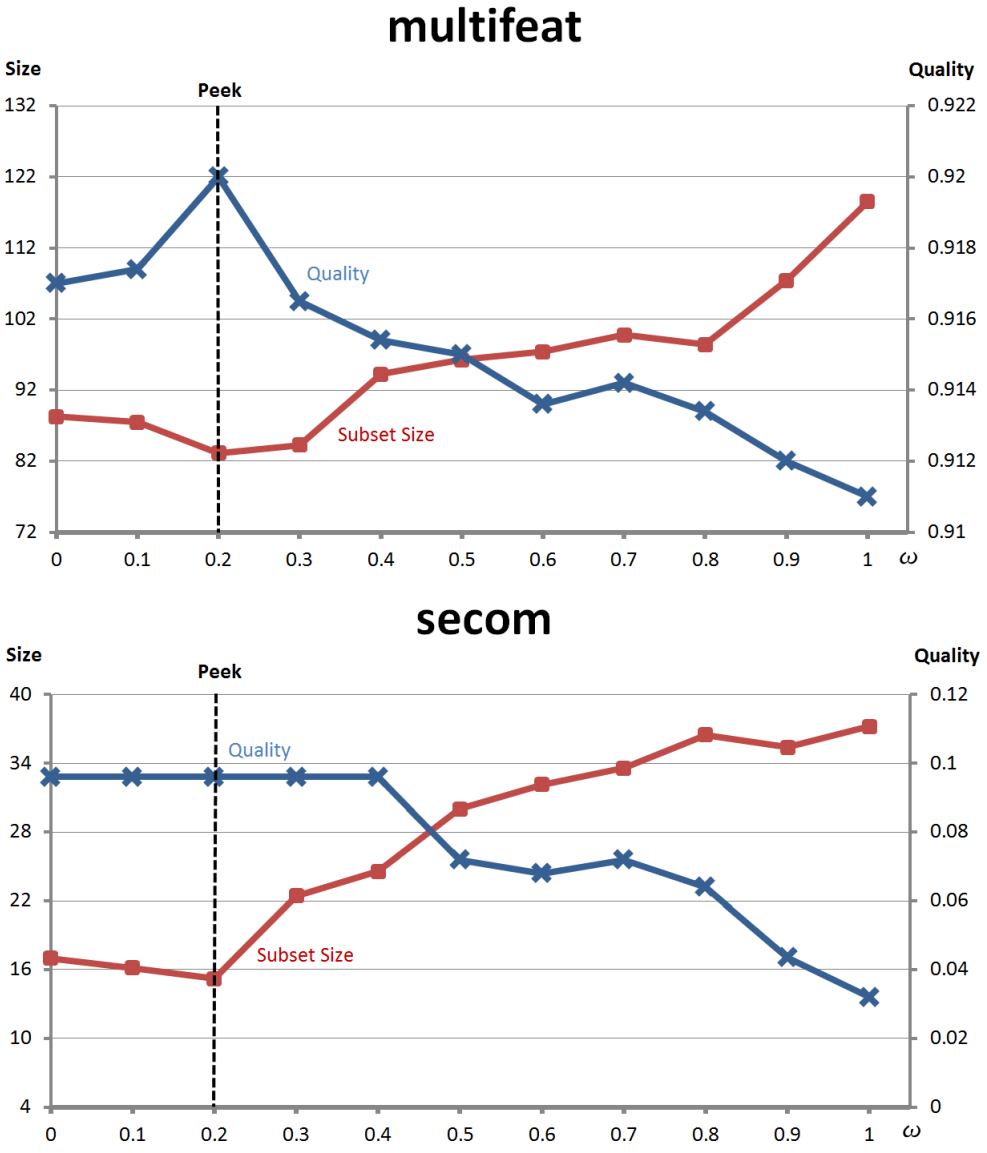

Figure 3.4: Demonstration of the effects of PAR using different $\omega \in [0, 1]$

results show that the added enhancements can indeed further improve the quality of the resultant feature subsets (when compared to the original HSFS approach), and also obviate the need to precisely pre-configure the size of the musician group.

Chapter 4

Feature Grouping-based Feature Selection

The general concept of feature grouping (FG) can be simply considered as a partitioning of a set of features. The purpose is to group so-called similar features and place them into the same group. Having completed this, information regarding inter-feature correlation can be estimated easily. Depending on the measure used to compute the inter-feature similarity, features within a group may be determined to be highly redundant with respect to each other, equally relevant to the decision attribute, or not only highly redundant with respect to each other but also equally relevant to the decision attribute while features between groups may be known to be less redundant (or even independent). In particular, FS may potentially benefit from FG techniques, for example, a-priori knowledge that the properties of features can be exploited by FS. That is, FS can be efficiently and effectively performed on a grouping of features. This has therefore inspired novel frameworks of FG-based FS, which has been carried out in the present research.

There may exist two forms of FG with regards to feature overlap between groups: so-called exclusive FG where any single individual feature only belongs to a single group [106], and non-exclusive FG where different groups may contain the same features [100]. Exclusive FG can be a special case of a more general non-exclusive FG when there are no overlapping features amongst the groups. To implement FG, conventional clustering techniques (e.g., k-means, c-means, hierarchical, and

graph-theoretic clustering [104]) in conjunction with inter-feature similarity metrics may be applied in order to cluster those redundant features together.

Traditionally, clustering techniques are focused on grouping similar data instances together. Distance metrics (e.g., Euclidean distance) are used to calculate the similarity between instances. However, feature is a nominal term such that the similarity between features cannot be simply computed using quantitative metrics. Information-based metrics such as mutual information [65], correlation coefficient [93], fuzzy-rough set dependency [177], and probabilistic consistency [10] can be used for the purposes of computing the similarity between any pair of features. Such measurable high-level similarities lead to classical clustering methods that are available to obtain homogeneous feature groups.

4.1 Proposal for High-level Framework for Feature Grouping-based FS

In the framework of FG-based FS proposed in this thesis, FS is performed on a number of feature groups by selecting representative features from each single group (or selecting none or multiple features for some particular groups) in order to obtain a feature subset rather than identifying a subset in $2^{|A|}$ combinations of features (where A is all of the conditional features of a given dataset). Two schemes can be used for implementing FG-based FS. In the first scheme, FG is considered an independent system and acts as a preprocessing step for FS. In the second, FG is used as an internal component of FS which will iteratively interact with other components of FS such that both a desirable feature grouping and a quality subset can be refined. These two structural schemes are discussed in more detail in the following sections.

4.1.1 FS with FG Being Preprocessed

This structural FG-based FS scheme, formally described in Fig. 4.1, consists of two main parts: 1) generating a desirable grouping by clustering similar features together using the feature clustering/partitioning techniques; and 2) obtaining the feature subset by performing FS on the pre-generated grouping. These two process are independent. The grouping process is completed prior to the outset of executing the FS. A powerful mechanism for obtaining a quality exclusive feature grouping, on which the optimal or suboptimal feature subset can be discovered, is therefore

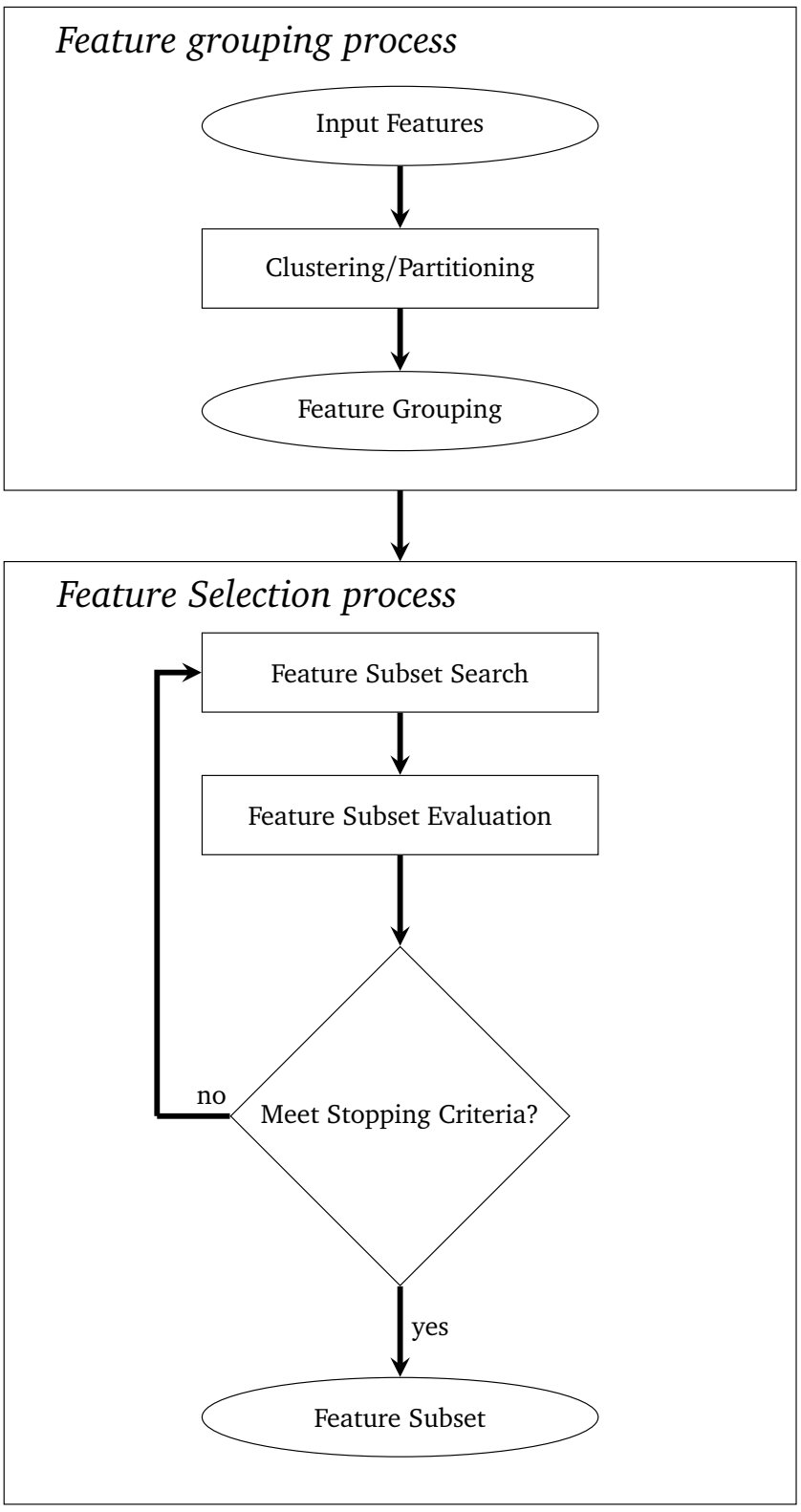

Figure 4.1: FS with FG as a preprocessing step

essential. Having such feature groupings been determined, a certain number of feature combinations that may involve the optimal subset cannot be reached when the inclusion of features is selected partially or not at all from every single feature group. This issue can be alleviated by a non-exclusive FG strategy, which builds a group for each of the input features in a given dataset, and then absorbs a similar feature as the member of existent groups. In doing so, all combinations of feature subsets are fairly accessible disregarding the limitation of the search methods used.

The possible advantages of FG preprocessing features into several homogeneous groups consist of two aspects: 1) the largest size of emerging feature subsets can be determined in advance by an FG when applying the obtained FG to FS that works by selecting one feature from each group to form a feature subset; and 2) it is not necessary to repeat the grouping process to obtain a quality FG.

4.1.2 FS with FG Being An Internal Component

In this structural FG-based FS scheme as formatted in Fig. 4.2, the FG process is used as an internal component in the FS approaches, and is capable of communicating with the subset selection process on the fly. Such communication is illustrated in the work flow of this FG-based FS scheme as listed below:

1. Obtain a feature grouping by performing the FG methods on the entire set of input features of a given dataset.
2. Generate a feature subset by combining features selected from this grouping (e.g., choosing one feature from each group).
3. Evaluate this selected feature subset using any of the evaluation functions described in Chapter 2.
4. If the evaluation result of the feature subset selected from the current feature grouping is better than that of the subset selected from the previous feature grouping, repeat steps 1-3.
5. Otherwise, terminate algorithm.

As seen from the above algorithm steps, obtaining a reasonable feature grouping depends upon the quality of subset produced from it. In doing so, the feature grouping is improved in terms of finding better subsets on its feature groups. However,

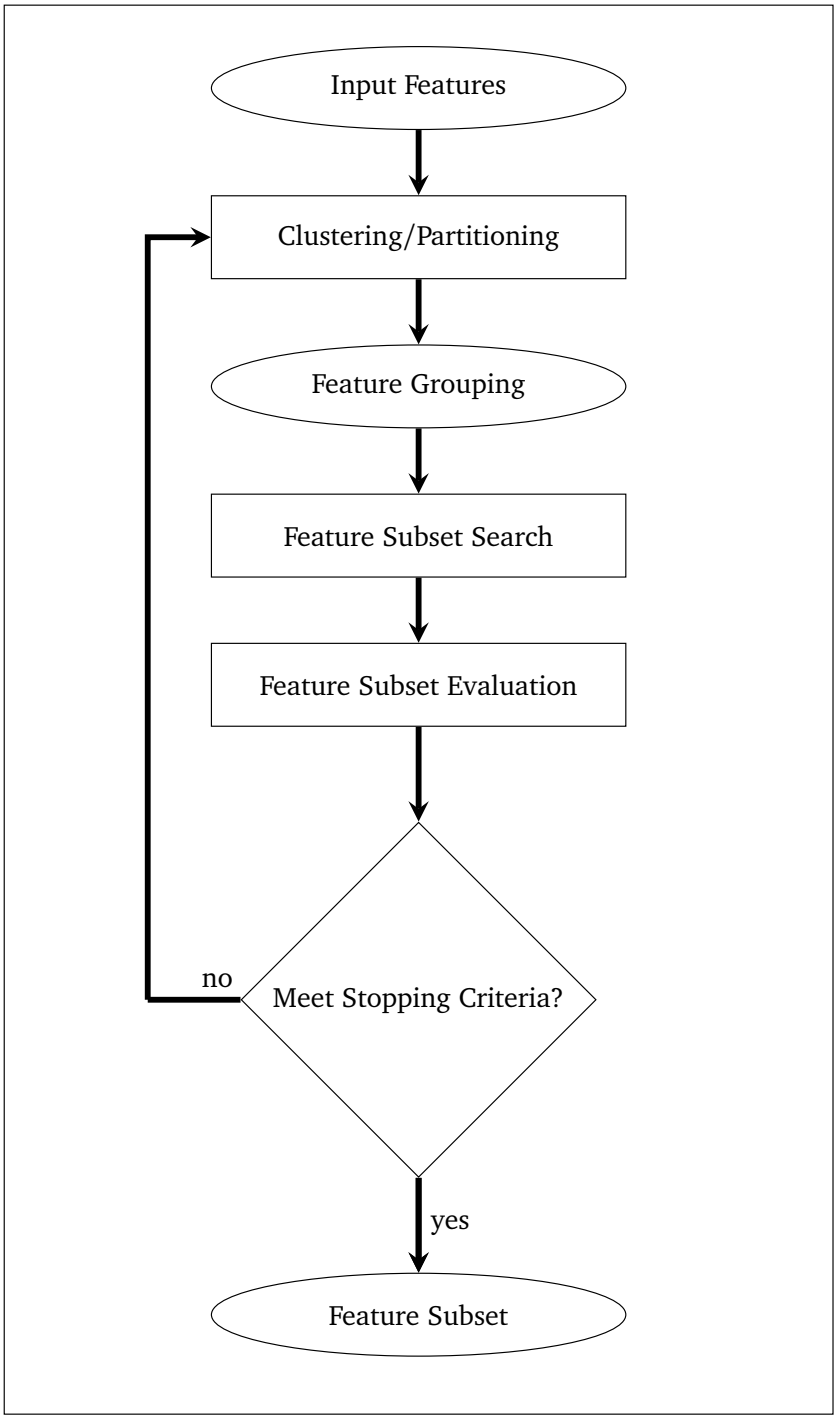

Figure 4.2: FS with FG as an internal component

repeatedly calling the FG process actually augments the computation overhead of this algorithm, although the subset selection process performing on the resulting feature grouping may reduce its overall computational overhead.

4.2 Feature Grouping-based FS using Graph-theoretic Approaches

Information in terms of the redundancy between relevant features (aka., inter-feature relevance) is the main focus of this work. Therefore, identifying homogeneous feature groups may help in removing redundancy from the final returned feature subset. This chapter introduces a series of different strategies to improve existing FS approaches based on minimum spanning tree-based feature grouping (GBFG) [199], where FS is implemented by feature selecting from the groups of an emerging feature grouping. Firstly, three-way mutual information has been adopted in order to compute the relationships between features, rather than the symmetrical uncertainty measure using Shannon entropy [231], such that redundant and collaborative information regarding the decision will be retrieved. Secondly, two new feature selection mechanisms based on GBFG are proposed: one based upon the straightforward ranking of features from the resultant feature groupings, and another which is based on a music-inspired metaheuristic. In addition, a grouping procedure with iterative refinement is also described using the evaluation result of the returned feature subset.

The improved GBFG-based FS framework is illustrated in Fig.4.3, where the ellipse blocks stand for data flow, the rectangle blocks of solid lines represent a process, and the rectangle blocks of dashed lines are the methods or measures that may be used to implement the specified processes. The improved framework is different from the original approaches. In the original GBFG-based FS approaches, there is an initial step to remove features that are considered irrelevant using the symmetrical uncertainty measure. The removal of such features may result in information loss regarding inter-feature collaboration (e.g., the XOR problem as described in Table 4.1 and those inputs of the individual variable 'a' (or 'b') are irrelevant with respect to their outputs while those inputs 'a' and 'b' together are fully relevant with respect to their outputs). This process of removing irrelevant features, therefore, is pruned in the improved framework. The more precise technical details

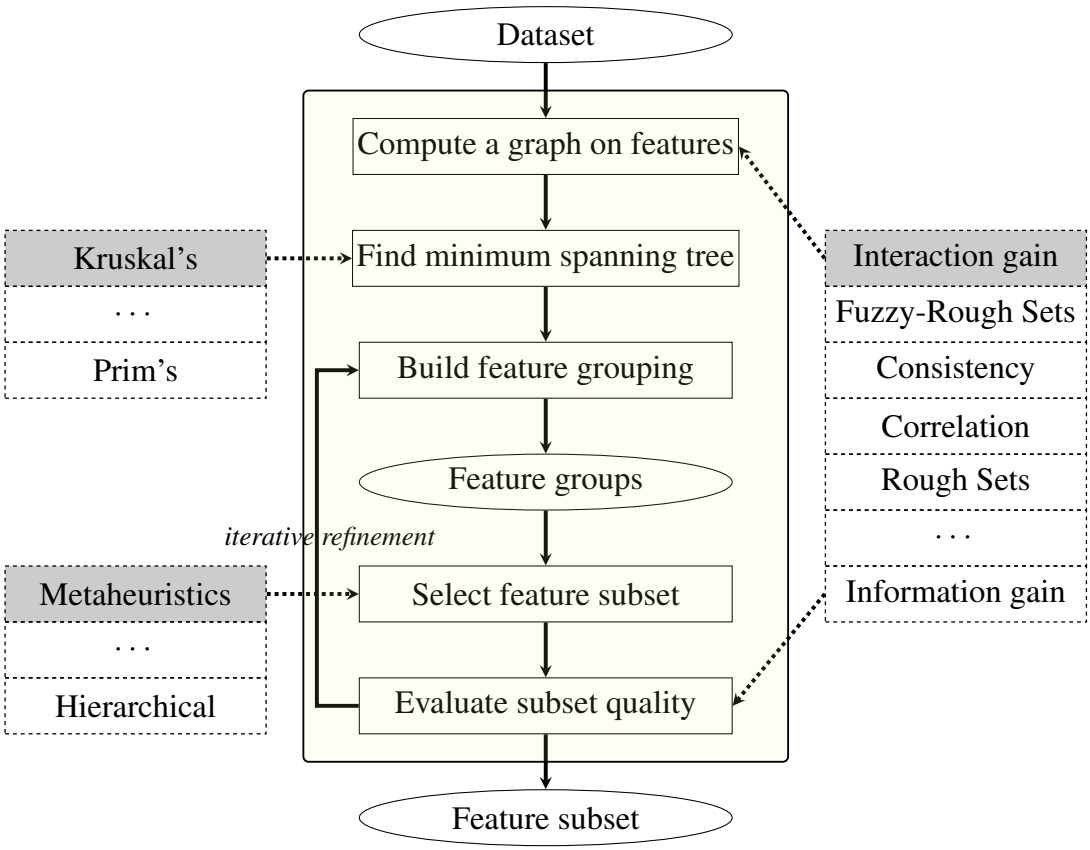

Figure 4.3: Framework for feature selection using GBFG

relating to GBFG and its components are presented in Section 4.3 and FS with GBFG is described in Section 4.4.

Table 4.1: The XOR problem in FS

a	b	a ⊕ b
0	0	0
0	1	1
1	0	1
1	1	0

4.3 Graph-based Feature Grouping

In this section, a three-way mutual information metric is used to measure relationships between features and then generate a feature graph, which is represented as

an adjacency list in order to efficiently build a minimum spanning tree (MST) upon this constructed graph. Thus, the resulting MST is used to obtain feature groupings.

4.3.1 Relationship Metrics and Interaction Gain

The problem of how features may be regarded as either redundant or relevant is significant to the FS. Generally speaking, redundant features are those whose information content is already present in other features; and irrelevant features are those which provide no information with respect to the decision attribute, while naturally, relevant features are highly correlated to the decision. Moreover, relevant features are further divided into two classes according to the level of information which they carry, namely, *strong relevance* and *weak relevance*. In terms of such relations between features, or relations between features and the decision attribute, three possible ways of forming feature groups, if not all, are discussed: 1) simply clustering highly redundant features; 2) clustering features that are equally relevant to the decision attribute; and 3) clustering features that are not only highly redundant (with respect to each other) but also equally relevant to the decision.

Many of the quality metrics developed in the literature (e.g., those reported in [100, 106, 199, 242]) may be adopted in order to distinguish between different types of features by using them as a measure of correlation or dependency. High values of correlation can be used not only to indicate redundancy between features but also to suggest that the relationship between features and the decision are strongly relevant. A moderate value typically implies a weak relevance, while a value level close to zero signifies irrelevance. This regularisation is always defined independently of the decisional features, because metrics most used in the literature can only be applied to assess the pair-wise relationship between two features. Either of paired features could be conditional or decisional.

Taking a different approach in this research, a measure of three-way mutual information is used to build a graph from the original features. This measure is also known as interaction gain [234], which is a metric that attempts to identify relationships between domain features, including collaboration and redundancy with respect to the decision. No assumptions are required when applying three-way mutual information to feature selection, such as those assumed in the maximum-relevance and minimum redundancy based approach [170]. Also, employing this measure has the advantage that it helps to identify the relationships between subsets

of features that are similar, again with respect to the decision attribute [194]. The formality of three-way mutual information is described in conjunction with its properties in Chapter 2.1.1.1.

4.3.2 Feature Graph Construction

In the very initial stage of GBFG, a graph is constructed according to the distribution of the features and their relatedness. Each of the conditional features is represented by a node in the graph and the relationships between features are represented by graph edges. Let $A = \{a_1, a_2, \cdots, a_{|A|}\}$ be the full set of conditional features in a dataset, $a_i, a_j \in A$ and Z be the decisional features, the concept of feature matrix which represents the relatedness can be introduced with the normalised interaction gain, I_{ij}, $i, j \in \{1, 2, \cdots, |A|\}$, $i \neq j$, which is computed from $I(a_i, a_j, Z)$ between the features a_i and a_j as per Eqn. 2.9. Z is a set of decisional features. In so doing, a link between a pair of features is established if a non-zero normalised interaction gain is calculated between them and the link is weighted by such calculated gain values.

Table 4.2: Example: feature relatedness matrix

Feature	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
a_1	0	**0.25**	0	0.4	0	0.5	0.6	0
a_2		0	0.15	0	0	**0.1**	0.35	0
a_3			0	0.45	**0.3**	**-0.6**	0	0
a_4				0	0.2	0.3	**-0.5**	0
a_5					0	0.4	**0.15**	0
a_6						0	0	0
a_7							0	0
a_8								0

Table 4.2 shows an example of a graph matrix. Features $a_1 - a_7$ have certain non-zero weighted links with the others. Feature a_8 is an extreme case which has no interaction with the others at all. In terms of three-way mutual information, this means that a_8 provides information about the decision independently of the rest. The addition of such features will neither improve the performance of inter-feature collaboration of the others nor demonstrate that they are redundant with respect to the rest. Such features naturally have no available links with the others and are disregarded at the feature grouping stage. The graph constructed with this matrix

can be depicted as shown in Fig. 4.4. Of course, in making feature selection, all such features should be included in the returned subset owing to their independence of those features selected from the resulting groups. However, noisy features may be incorrectly identified as such independent features. Therefore, in practical use, an examination of these features should be provided before their inclusion.

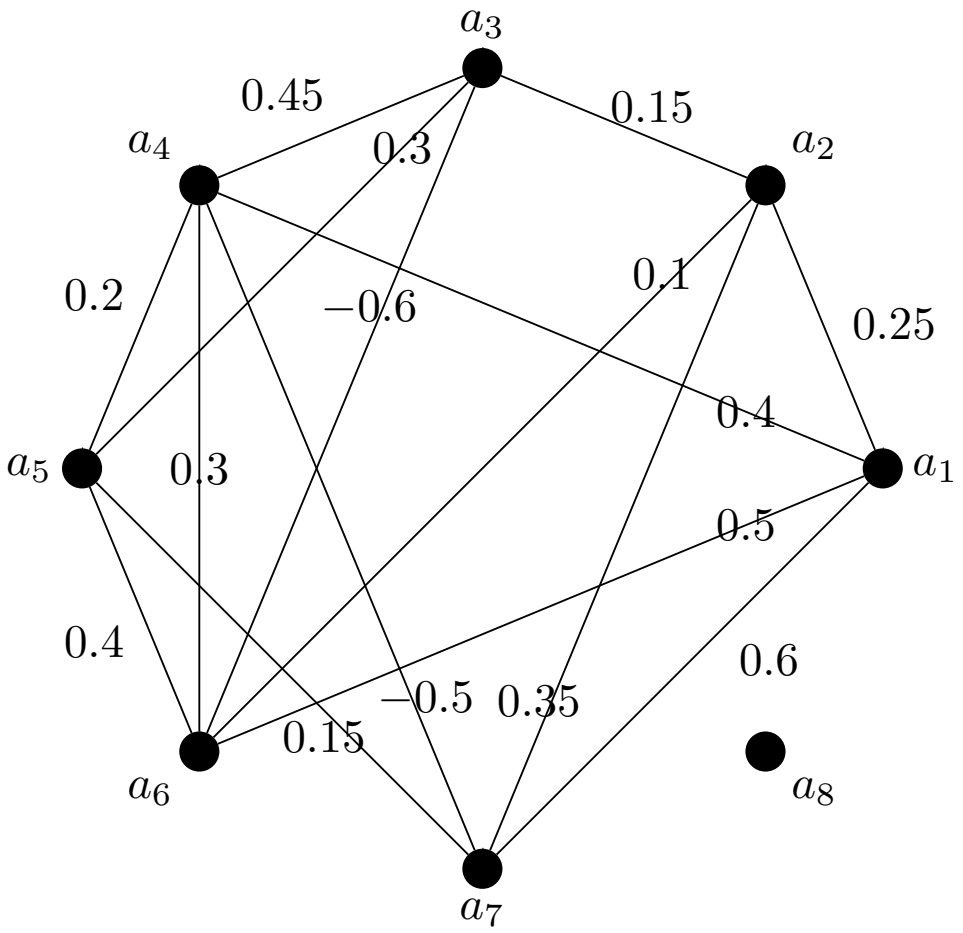

Figure 4.4: Graph constructed with the link weights given in Table 4.2

To represent a graph efficiently which helps information storage and retrieval, the concept of an adjacency list is introduced. An adjacency list is itself a collection of unordered lists of neighbours of the nodes given in the graph and is a common way to represent structural relationships in graph theory. For example, the graph of Fig. 4.4 can be re-represented as the adjacency list in Fig. 4.5, where the head of the list represents the features and from which the arrows map onto their neighbours. Each neighbour has two elements: the first points to an adjacently connected feature; and the second element encodes the weight of the link between the two features.

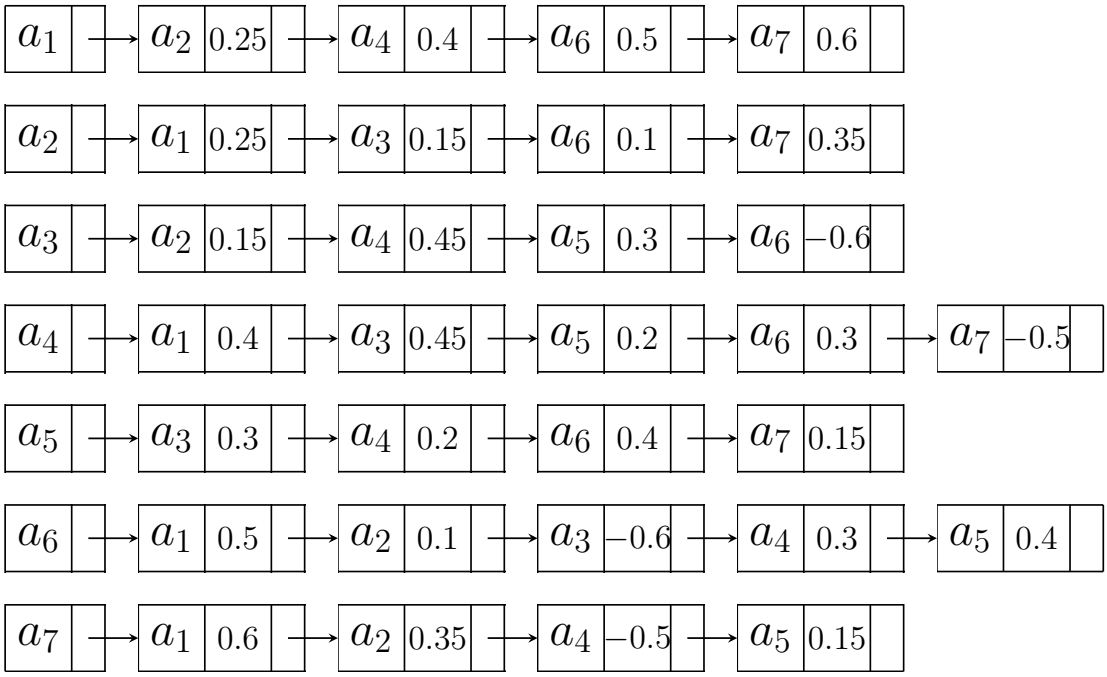

Figure 4.5: Adjacency list for the weighted graph of Fig. 4.4

Such an adjacency list helps to reduce computational overhead when building a so-called MST from the original graph. An MST is basically a sub-graph of a graph, containing all of the nodes from the original super graph such that its nodes are connected together with the minimal total weighting for all edges. Importantly, an MST does not involve cycles for any of its edges. From the adjacency list obtained in Fig. 4.5, the MST illustrated in Fig. 4.6 can be easily derived. Note that if the constructed graph is not connected, then a minimum spanning forest (MSF) that is a set of MSTs (one of which is implied by an independent component in the constructed unconnected graph) is derived instead of an single MST.

4.3.3 Grouping of Features

The use of three-way mutual information reinforces the concepts of collaboration and redundancy. Features which are collaborative can provide more information about the decision than they can individually, and features which are redundant provide common information (possessed by each of them) about the decision attribute. The larger the positive value of three-way mutual information, the stronger the collaborative contribution of the features. Similarly, the smaller the negative value of

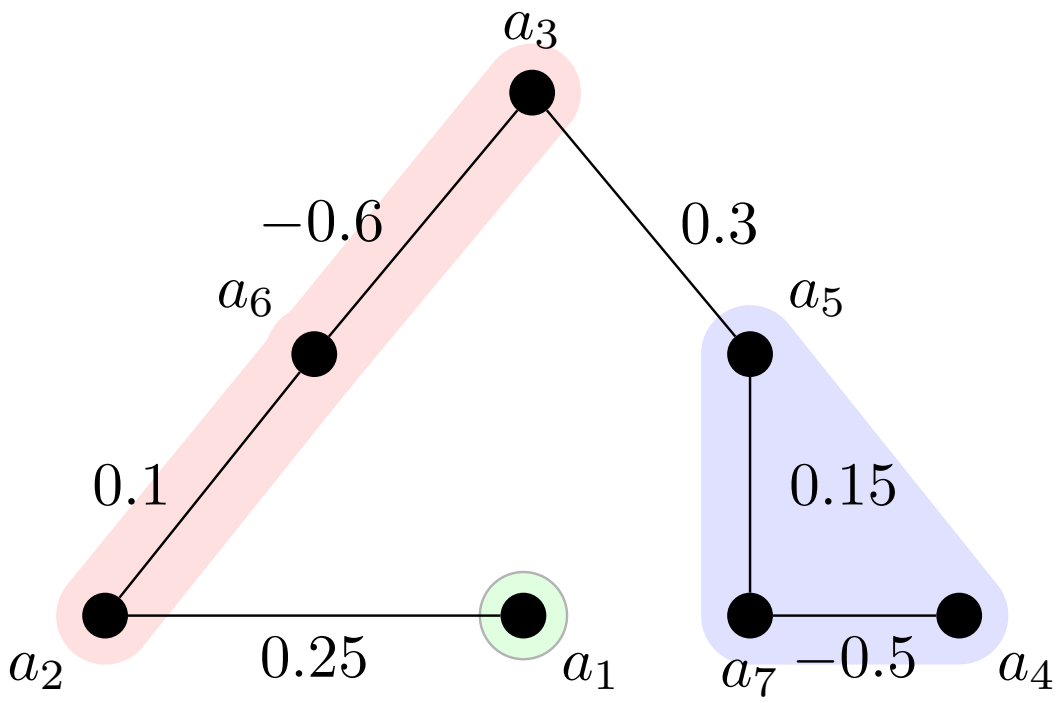

Figure 4.6: A possible MST derived from the original graph (using the example adjacency list of Fig. 4.5)

three-way mutual information, the higher the level of redundancy between features. Reflecting the above observations, features that are of low collaboration or highly redundant with respect to the decision are desirable candidates for membership of the same group such that a feature subset with strong collaboration may be obtained when applying FS on these resultant feature groups. To group features following this observation can be achieved by an MST in conjunction with three-way mutual information.

Excluding those features that are independent of the others, non-zero weighted edges are then added to an adjacency list in order to obtain an MST by using the generic and effective Kruskal's algorithm [41]. Given a connected, undirected, and weighted graph G, this MST algorithm works as follows:

1. Initialise a new graph with non-independent features as nodes without an edge, each forming an individual component (or tree).

2. Select one of the lightest-weighted edges from G randomly.

3. Judge whether the edge connects two different components: if so, these two components are merged together into one; otherwise, this edge is discarded.

4. Repeat steps 2 and 3 until all nodes within the new graph are connected and return the new graph as the resulting MST.

The MST built through the above process contains $|A'|$ nodes and $(|A'|-1)$ edges, where A' is a subset of all possible features A in the given dataset minus those independent features. Denote the collection of all independent features as A'', then $A'' = A \setminus A'$. In particular, each independent feature in A'' will make a group on its own. Note that such resultant T may not be unique, however the sum of edge weights will be identical for the different versions of T (given the fact that all versions are minimum spanning trees in the first place).

In an MST T, the removal of a certain number of edges will result in a forest F, which contains the same number of subtrees as the number of removed edges plus one. The proposed approach works by iteratively removing edge(s) of the overall largest weight from the MST such that features are separated into several different subtrees. Each subtree may then be viewed as an emerging feature group because features within this subtree are highly redundant or of low collaboration. Thus, the resulting forest F can be interpreted as a grouping of features.

For example, recall the MST of Fig. 4.6, which is derived from the graph as per Fig. 4.4 where $a_8 \in A''$ has no links with others, firmly forms an independent group in itself, and is omitted at the current stage. From this, the feature grouping process can be illustrated as follows: Initially, the forest F contains only a single tree $T = \{A', E'\}$ where $A' = \{a_1, a_2, \cdots, a_7\}$ and $E' = \{< a_3, a_5 >, < a_1, a_2 >, < a_5, a_7 >, < a_2, a_6 >, < a_4, a_7 >, < a_3, a_6 >\}$, with its elements respectively weighted by those given in the list $\{0.3, 0.25, 0.15, 0.1, -0.5, -0.6\}$. The proposed approach attempts to remove edges that take the maximum weight in F iteratively. Thus, the edge $< a_3, a_5 >$ (of a weight 0.3) is removed in the first instance and the current tree is then divided into two subtrees $T^1_{new} = \{\{a_4, a_5, a_7\}, \{< a_4, a_5 >, < a_5, a_7 >\}\}$ and $T^2_{new} = \{\{a_1, a_2, a_3, a_6\}, \{< a_2, a_3 >, < a_1, a_2 >, < a_2, a_6 >\}\}$. Note that if there exist other edges with the weight of 0.3, these edges are then removed within the same iteration. The F now contains two members $\{T^1_{new}, T^2_{new}\}$ rather than the original single state $\{T\}$. As such, a grouping of two feature groups $\{\{a_4, a_5, a_7\}, \{a_1, a_2, a_3, a_6\}\}$ is formed in place of the previous fully connected tree.

4.3.4 Complexity of GBFG

GBFG applied to the all input features excluding those independent of the others has three stages: 1) the calculation of feature relationships in the graph; 2) the building of an MST; and 3) the grouping of features. The complexity can therefore be considered in terms of these three steps.

Without losing generality, suppose that the input dataset has $|A|$ features. In the first stage, the complexity of computing the relationship for each pair of features is $O({}^{(|A|^2-|A|)}/_2)$. Note that in practical terms, certain features may not have any correlation with others and these independent features can be pruned from an emerging MST. In stage two, suppose that $|A''|$ features are removed owing to their independence of all the other features, an emerging MST will have $|A'|$ nodes ($|A'| = |A| - |A''| \leq |A|$) and hence, the graph generated from this stage will have ${}^{(|A'|*(|A'|-1))}/_2$ edges. The complexity of building an MST is therefore $O({}^{(|A'|^2-|A'|)}/_2)$. In the third stage, to obtain a feature grouping containing $|F|$ groups (F is a forest of subtrees of an MST, each of which represents a feature group), $(|F|-1)$ edges will be eliminated from the resulting MST in total. The grouping process is then of the order $O(|F|-1)$. Additionally, each of the $|A''|$ independent features will form a group itself. Building such groups requires $|A''|$ operations. Thus, the complexity of the entire approach in conjunction with the group building process for those independent features is $O({}^{(|A'|+|A''|)^2-(|A'|+|A''|)}/_2 + {}^{(|A'|^2-|A'|)}/_2 + |F| - 1 + |A''|)$ in total. For further understanding the grouping process, its pseudo-code is given in Algorithm 4.3.1.

4.4 Feature Selection with GBFG

In this section, two novel feature selection techniques based on GBFG are proposed. The first method carries out feature selection through a straightforward quality-guided selection of features from the generated feature groups. The second performs feature selection from the feature groups by employing harmony search [81] in an effort to discover equivalent or better feature subsets more efficiently than the first method. The key is of course how to efficiently search for effective feature subsets. This forms the main part of the work below.

A: set of conditional features
Z: decisional features
// Step 1: construct a graph using features by calculating three-way mutual information between features
$G = \{V, E\}$: undirected graph, where $V = A$ and $E = \emptyset$
while $a_i, a_j \in V$ **do**
    if $< a_i, a_j > \notin E$ **then**
        $Weight(< a_i, a_j >) = \frac{I(a_i, a_j, Z)}{H(a_i) + H(a_j)}$
        $E = E \cup \{< a_i, a_j >\}$
// Step 2: build an MST, T over G using Kruskal's Algorithm
Remove those independent features A'' in G
$T = Kruskal(G)$ where $T = \{A', E'\}$
// Step 3: generate feature grouping
while *the current grouping cannot generate a better feature subset* **do**
    Remove edge(s) with the maximum weight in T
    Generate feature grouping by clustering those features that are linked as a group
Build an individual group for each feature that is independent of the others

Algorithm 4.3.1: GBFG: Graph-based feature grouping

4.4.1 Evaluation of Feature Subset Quality

To start with, the quality measure that is used to adjudge the emerging feature subset quality is presented first. Feature subsets are formed following the choice of representative features from every single group (resulting from running Algorithm 4.3.1), and then are evaluated using a certain quality measure. Here, the popular probabilistic consistency measure is used (although other evaluation methods such as those based on mutual information [65], correlation coefficient [155], and fuzzy-rough set dependency [111] may be used as an alternative). In addition to its popularity, probabilistic consistency is chosen as the quality measure because it offers a consistent mechanism to calculate the discriminability of a given feature subset $S \subseteq A$ with respect to given class labels. The principle of probabilistic consistency is elaborated in Chapter 2.1.1.1.

Suppose that the evaluated quality of the candidate feature subset selected from the current feature grouping is better than that of the subset selected from the previous grouping, indicating the grouping process may be further improved (to

potentially lead to better quality of feature subsets). Thus, at this stage, the grouping building process is recalled to generate a new feature grouping by further eliminating edges in the previous MSTs. From this, a new feature subset can be obtained by selecting representative features from every single group. This iterative refinement terminates if no better feature subsets are returned. The following presents two implementations of an FS mechanism, both of which utilise the feature subset quality measure as defined above.

4.4.2 Search for GBFG-based Quality Feature Subsets

4.4.2.1 Straightforward Quality-guided Feature Selection

This method can be readily implemented using three-way mutual information. The key steps are described as listed below:

1. Locate a pair of features taking the largest value from three-way mutual information, which indicates that these two features are most collaborative. Note that such a pair of features may not be unique. If there exist more than one pair of features like this, the first located pair will be returned.

2. Include these two most collaborative features into the emerging feature subset S.

3. For the remaining feature groups that do not contain features in S, find a feature from each group that is most collaborative with S about the decision attribute and add it to S until all groups have nominated a representative feature. Here, the degree of collaboration between a feature a_i of a given group and S is recursively defined as follows:

$$Col(a_i, S) = \Sigma_{a_j \in S} I(a_i, a_j, Z). \tag{4.1}$$

Algorithm 4.4.1 implements this straightforward feature selection method. In order to locate a pair of most collaborative features in the constructed graph as per Fig. 4.4 that has up to $(|A'|^2 + |A'|)/2$ non-zero weighted edges where A' is the set of features excluding those independent features A'' out of the all input conditional features A, it will take $(|A'|^2 + |A'|)/2$ operations to find such a pair of features for the worst case. Suppose that a feature grouping of $|F|$ groups is obtained by the GBFG algorithm, where F is a forest of subtrees of an MST (each of which represents a

feature group). F' is a set of feature groups that contain two most collaborative features, which have been selected as the elements of an emerging feature subset. Those groups that have not nominated a representative for the emerging subset yet are denoted as F'', then $F'' = F \setminus F'$. As features in the feature groups of F'' are of potential collaboration with those features that have been selected in an emerging subset, for each group $T^i \in F''$ ($i = 1, 2, \cdots, |F''|$), one of the features that are most collaborative with the elements of the emerging subset is designated to be a new member of it. Thus, $\Sigma_{i=1}^{|F''|}|T^i|$ operations are required to search for such features in F'' groups. The examination and inclusion of those independent features is of the order $O(|A''|)$ as they may provide additional information regarding the decision. Therefore, the complexity of generating a feature subset is $O(^{(|A'|^2+|A'|)}/_2 + \Sigma_{i=1}^{|F''|}|T^i| + |A''|)$ at a time. If such a subset generation process runs ι ($1 \leq \iota \leq |F|$) times until the desirable subset is found, the total complexity of the straightforward quality-guided feature selection will be $O(\iota * (^{(|A'|^2+|A'|)}/_2 + \Sigma_{i=1}^{|F''|}|T^i| + |A''|))$.

```
1  F = {T^1, T^2, ··· , T^n}: set of feature groups from graph-based grouping
2  S = ∅: set of selected features
3  Search edge < a_i, a_j > of the largest weight in the constructed graph as shown
   in Fig. 4.4
   // There may be more such edges in the constructed graph
      and the first met is obtained.
4  Include features a_i and a_j into S
5  foreach i = 1 to n do
6      if T^i ∩ S == ∅ then
7          foreach a ∈ T^i do
8              if Col(a, S) is the largest then
9                  S = S ∪ {a}
10 A'': the collection of all independent features
11 for a ∈ A'' do
12     if f(S ∪ {a}, Z) > f(S, Z) then
13         S = S ∪ {a}
14 return S
```

Algorithm 4.4.1: GBFG-FS: Straightforward feature selection

To continue the example used in Section 4.3.3, a grouping of two feature groups $\{\{a_4, a_5, a_7\}, \{a_1, a_2, a_3, a_6\}\}$ is obtained after the first iteration of the GBFG algorithm without yet considering any independent features. Based on the graph constructed using three-way mutual information, features a_7 and a_1 are identified to be most

collaborative with each other and, therefore, they are included in the emerging feature subset $S = \{a_7, a_1\}$. In this particular case, a_7 and a_1 appear in $\{a_4, a_5, a_7\}$ and $\{a_1, a_2, a_3, a_6\}$ respectively. These two feature groups have therefore nominated a feature each. However, more generally, there may exist situations where two most collaborative features are in the same group. In such cases, both of these two features are included in the emerging subset S, and then another feature is selected from each of the groups that has yet to nominate a feature representative to S. If the evaluation of this obtained subset is better than that of the subset selected from the previous grouping in the same manner, the feature grouping process continues while assigning S to an empty set. Otherwise, the algorithm returns the selected feature subset and terminates. In this example, suppose that the evaluation result of the feature subset obtained from the current grouping is better than that of the subset produced from the previous grouping. Thus, the next grouping results in three groups: $\{a_1\}$, $\{a_2, a_3, a_6\}$, and $\{a_4, a_5, a_7\}$. As with the last iteration, a_1 and a_7 are included in S. For the group $\{a_2, a_3, a_6\}$ that has yet to nominate a feature, compute the collaboration between each feature of $\{a_2, a_3, a_6\}$ and S based on Eqn. 4.1. The value of the collaboration of a_2 and S is $0.25 + 0.35$; that of a_3 and S is $0 + 0$; and that of a_6 and S is $0.5 + 0$. As a_2 has the largest collaboration value, it is then included in S. This leads to feature subset $\{a_1, a_2, a_7\}$ being selected.

In the above illustration, the feature a_8, which is independent of the others, has been excluded intentionally to simplify the description. However, as an independent feature, it may provide different and possibly useful information to the decision. From this viewpoint, it is necessary to evaluate the quality of the feature subset of $\{a_1, a_2, a_7, a_8\}$. If this inclusion improves the evaluation result, features a_1, a_2, a_7 and a_8 will be returned as the outcome of feature selection. Otherwise, return only features a_1, a_2, and a_7 as the feature selection result.

4.4.2.2 GBFG-based Feature Selection with Harmony Search

As mentioned previously, the tree generated in the MST discovery step may not be unique and this means that the overall process is non-deterministic. This is where a metaheuristic approach may offer assistance in strengthening the work. Selecting an 'optimal' feature subset from groups of features is a combinatorially difficult problem. Exhaustive search can guarantee that the best feature subsets will be discovered, but this is often computationally impractical for real-world applications. Harmony

search (HS) [81] may be useful for the task of selecting features from a particular feature grouping. This observation has inspired the following development.

HS for GBFG-based Feature Selection The original harmony search-based feature selection approach (HSFS) as represented in [54] can be extended to take advantage of the GBFG framework. In particular, musicians are mapped onto feature selectors, which are equal to the number of features of a given dataset. Suppose that a dataset has $|A|$ conditional features, each selector then has $|A|$ choices. This means that the space for the harmony search is $|A|^{|A|}$ in theory. Here, harmony search is employed in order to reduce the size of the search space and thus computational effort. In contrast to the original approach [54], where feature selectors are assigned for each feature, here a single selector is assigned to each feature group. Given a feature grouping of $|F|$ groups that are obtained by the GBFG algorithm from $|A'|$ ($|A'| = |A| - |A''|$) features where A'' is a set of independent features and A is a set of all conditional features, the search space is then reduced to $|F|^{|A'|}$ (where typically $|F| \ll |A'|$), while a single selector has $|A'|$ choices. Again, the original approach treats FS as a bi-objective optimisation problem while the new algorithm turns FS into a single objective optimisation problem. That is, the feature subset size is no longer needed to be considered in evaluating feature subsets thanks to the introduction of GBFG. The steps of the GBFG-based HS algorithm are listed as follows:

1. Initialise parameters: In this algorithm, five parameters continue to be used, including $|\mathbb{H}|$, HMCR, PAR, λ_{max}, and $|M|$. BW is not used any more. In FS, each feature is an independent granule. The neighbouring features cannot be computed simply by using the standard arithmetic operators and the random value, which are used by the proposals of BW that are developed in [81, 80]. Instead of using BW, the feature similarity measure via calculating the degree of probabilistic consistency between two features is then used for identifying the neighbouring features after the activation of PAR, which is not used in the original HSFS algorithm. The alternative similarity measures represented in [241] are also considered for use. As a given number of selectors is assigned to each of $|F|$ feature groups (F is a forest of subtrees of an MST, each of which represents a feature group) that are obtained by GBFG (see Section 4.3.3), $|M|$ is set to the number of groups $|F|$ rather than that of features $|A|$ ($|A| \gg |F|$) in the original HSFS algorithm.

2. Initialise HM: HM now is a two-dimensional matrix with the size of $|\mathbb{H}| \times |F|$. Each row stores a feature subset while each column stores HMS historical features, taking the form below:

$$\mathrm{HM} = \left[\begin{array}{cccc|c} a_1^{H^1} & a_2^{H^1} & \cdots & a_{|F|}^{H^1} & f(S^1, Z) \\ a_1^{H^2} & a_2^{H^2} & \cdots & a_{|F|}^{H^2} & f(S^2, Z) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_1^{H^{|\mathbb{H}|}} & a_2^{H^{|\mathbb{H}|}|} & \cdots & a_{|F|}^{H^{|\mathbb{H}|}} & f(S^{|\mathbb{H}|}, Z) \end{array}\right]$$

where the value of a unit $a_i^{H^j}, \overline{a}_i^{H^j}$ is a feature selected by the i^{th} feature selector in the j^{th} harmony and $f(S^j, Z)$ is the evaluation function used to calculate the quality of a given feature subset, S^j to the decisional features Z. Note that there may exist duplicated features in a harmony and thus the size of a feature subset may be less than its corresponding harmony.

3. Improvise new feature subsets: A new harmony $S^{H'} = \{\overline{a}_1^{H'}, \overline{a}_2^{H'}, \cdots, \overline{a}_{|F|}^{H'}\}$ is generated based on as the same three factors as the original HS, involving HMCR, PAR, and a random value r $(0 < r < 1)$. A new feature $\overline{a}_i^{H'}$ is selected by the i^{th} feature selector with respect to the following rules:

$$\overline{a}_i^{H'} = \begin{cases} \overline{a}_i^{H'} \in T^i & \text{if HMCR} \leq r \\ \overline{a}_i^{H'} \in \mathrm{HM}_i = \{\overline{a}_i^{H^1}, \overline{a}_i^{H^2}, \cdots, \overline{a}_i^{H^{|\mathbb{H}|}}\} & \text{if HMCR} > r \\ \text{the closest neighbour of } \overline{a}_i^{H'} & \text{if PAR} > r \end{cases} \tag{4.2}$$

where T^i is a homogeneous group of those features that are only utilised by the i^{th} feature selector and HM_i is a column of historical features selected by the i^{th} musician, being stored in HM. When the condition HMCR $\leq r$ is satisfied, the feature selector will randomly select a feature from its own homogeneous group. Otherwise, the feature selector will randomly select a feature from its historical feature pool. PAR works independently of HMCR. If the condition PAR $> r$ is satisfied, a feature from its historical feature pool is randomly generated and replaced by its closest neighbour feature. Recalling the graph of Fig. 4.4 constructed using three-way mutual information, neighbours of a feature are topologically connected features. The link of the smallest weight determines the closest neighbour of this feature in terms of all possible links regardless of whether the link is negative-weighted or positive-weighted.

4. Update HM: The quality of each newly-produced harmony is computed using the probabilistic consistency measure $f(S, Z)$ after converting the harmony to a feature subset. With respect to the evaluated quality, the update of HM is then denied if no existing feature subsets worse than the new feature subset are found. Otherwise, the HM is updated based on the following rule:

$$S^{H'} \in \text{HM} \wedge S^{H^{\text{worst}}} \notin \text{HM} \tag{4.3}$$

5. Check stopping criterion: If the number of improvisations reaches λ_{max} or the best harmony in HM has not been changed for a large number of iterations, the algorithm stops and returns the selected features as the outcome of feature selection in conjunction with those features that are independent of the others while improving the subset evaluation. Otherwise, repeat steps 3 and 4.

GBFG-HS Algorithm and Complexity The procedure of the GBFG-HS algorithm is illustrated in Algorithm 4.4.2. In line 1, the set of feature groups is obtained using GBFG. Lines 2-4 form the initialisation stage of GBFG-HS where $O(|F| * |\mathbb{H}|)$ operations are reserved to randomly populate HM. At the improvisation stage, which is implemented in lines 5-15, the generation of new harmonies is of the order $O(|F| * \lambda_{max})$ because all feature selectors each select a new feature at every iteration. In lines 16-18, those independent features A'', which may provide different information from the feature subset obtained by the HS process, are examined to decide whether to include them. It takes $|A''|$ operations for the inclusion of those independent features. Therefore, the single process of the GBFG-HS algorithm is $O(|F| * (\lambda_{max} + |\mathbb{H}|))$. Such a single FS process is repeated after a new feature grouping is obtained. If the subset generation process thus performs ι ($\leq \iota \leq |F|$) times until the desirable feature subset is attained, all searching processes using GBFG-HS will be of the complexity $O(\iota * (|F| * (\lambda_{max} + |\mathbb{H}|) + |A''|))$.

4.5 Experimental Evaluation

In this section, a series of experiments are conducted using 20 different UCI-MLR benchmark datasets [22]. Experimental studies include: 1) comparison with popular FS approaches that are based upon an existing feature grouping method (FRFG) [62] or upon metaheuristics or stepwise greedy search, covering genetic algorithms (GA-FS) [221], particle swarm optimisations (PSO-FS) [117], and greedy-hill-climbing

$F = \{T^1, T^2, \cdots, T^{|F|}\}$: set of feature groups from graph-based grouping
Z: decisional features
$S = \emptyset$: returned feature subset
$S^{H'} = \{\overline{a}_1^{H'}, \overline{a}_2^{H'}, \cdots, \overline{a}_{|F|}^{H'}\}$: new harmony being improvised
$\text{HM}_i = \{\overline{a}_i^{H^1}, \overline{a}_i^{H^2}, \cdots, \overline{a}_i^{H^{|\mathbb{H}|}}\}$: the i^{th} column of HM storing historical features selected by the i^{th} musician
$S^{H^{\text{worst}}}$ and $S^{H^{\text{best}}}$: worst harmony and best harmony in HM respectively
$|\mathbb{H}|$: the size of HM, indicating the number of harmonies
$|F|$: the amount of feature groups, indicating the number of musicians
`// Initialisation Phase`
for $j = 1$ **to** $|\mathbb{H}|$ **do**
  for $i = 1$ **to** $|F|$ **do**
    $\overline{a}_i^{H^j} = \texttt{Random}(T^i)$
  end
end
`// Iteration Phase`
while $(\lambda++) \leq \lambda_{max}$ && $S^{H^{best}}$ *has not been updated for a large number of iterations* **do**
  for $i = 1$ **to** $|F|$ **do**
    if `Random([0,1])` $< HMCR$ **then**
      $\overline{a}_i^{H'} = \texttt{Random}(T^i)$
    else
      $\overline{a}_i^{H'} = \texttt{Random}(HM_i)$
    end
    if `Random([0,1])` $< PAR$ **then**
      $\overline{a}_i^{H'} =$ the closest neighbour of $\overline{a}_i^{H'}$
    end
  end
  if $f(S^{H'}, Z) > f(S^{H^{worst}}, Z)$ **then**
    $S^{H'} \in \text{HM} \wedge S^{H^{\text{worst}}} \notin \text{HM}$
  end
end
for $i = 1$ **to** $|F|$ **do**
  if $\overline{a}_i^{H'} \notin S$ **then** $S = S \cup \{\overline{a}_i^{H'}\}$
end
A'': the collection of all independent features
for $a_i \in A''$ **do**
  if $f(S \cup \{a\}, Z) > f(S, Z)$ **then** $S = S \cup \{a_i\}$
end
return S

Algorithm 4.4.2: GBFG-HS: Feature selection with harmony search

(GHC-FS) [92]; and 2) comparison with the original harmony search-based feature selection approach (HSFS) [54].

4.5.1 Experimental Setup

Ten stratified randomisations of 10-fold cross-validation (10FCV) [19] are employed in generating the experimental results. Note that FS is performed as part of the cross-validation and each fold results in a new selection of features. Three different aspects of performance are examined in the evaluation: classification accuracy, final selected subset size, and average runtime per cross-validation fold. For classification, three different learning classifiers are used due to their availability (and also, popularity): JRIP [38], a rule-based classifier; J48 [176], a decision-tree learner; and IBk [42], a nearest-neighbour classifier (with $k = 3$). A paired t-test ($p = 0.05$) is used to examine the statistical significance of the generated results for both classification accuracy and subset size.

The datasets used for the experimental evaluation range in size from 120-5000 instances and 10-2557 features. Most of the data have 2-7 decision classes but a number of them have over 10 up to 19 (e.g., soybean). A summary of the datasets is shown in Table 4.3. As stated previously, all datasets are drawn from [22]; they are selected to facilitate comparative studies since they have been used by the other algorithms that are compared against here.

The parametric settings for the methods, with which the GBFG approaches are compared, are those typically used by the original approaches in the literature. In particular, the GA search-based FS method has an initial population size of 20, a maximum number of generations/iterations of 5000, crossover probability of 0.6 and mutation probability of 0.033. The number of generations/iterations for PSO search is set to 5000, whilst the number of particles is set to 20, with acceleration constants $C_1 = 2$ and $C_2 = 2$. For the harmony search approaches (both HSFS and GBFG-HS), the maximum number of iterations is again set to 5000, harmony-memory consideration rate to 0.7, pitch-adjustment rate to 0.8, and harmony memory size to 20. These parameters may not be ideal for all of the datasets employed here and an optimisation phase may well result in an improvement in performance. However, such a parameter optimisation phase would need to be performed on a dataset-by-dataset basis which would involve a significant investment of computational effort and therefore, is not adopted here.

Note that for consistency and fair comparison within all experiments concerning FS, the probabilistic consistency measure [45] is used in order to evaluate feature subsets. Other subset-based evaluation functions may also be applicable, such as correlation coefficient[155], rough dependency [127], and fuzzy rough dependency[62], but this is beyond the scope of this thesis.

Table 4.3: Summary of datasets

Dataset	Features	Instances	Classes
heart	14	270	2
glass	10	214	6
cleveland	14	297	5
olitos	26	120	4
ozone	73	2534	2
libras	91	360	15
arrhymythia	280	452	16
water2	39	390	2
water3	39	390	3
web	2557	149	5
wine	14	178	3
secom	591	1567	2
soybean	36	683	19
segment	20	1500	7
vote	17	435	2
ionosphere	35	230	2
credit-g	21	1000	2
breastcancer	10	286	2
multifeat	650	2000	10
waveform	41	5000	3
sonar	61	208	2

4.5.2 Comparison with Popular FS Methods

The proposed approaches (GBFG-FS and GBFG-HS) are evaluated in this section by comparing them with several popular existing FS methods that are readily available. FRFG [106] is an existing feature grouping based FS technique and a comparison is made here as it also performs feature grouping for the task of FS. GA-FS and PSO-FS use metaheuristic strategies while GHC-FS employs a greedy search. In FRFG, a grouping on features is formed by clustering redundant features, which are defined if the degree of correlation between features exceeds a predefined threshold β that

Table 4.4: Classification accuracy (%(sd)): unreduced data

Dataset	Classifiers		
	IBk(k=3)	J48	JRIP
heart	79.11(6.74)	78.15(7.38)	78.63(7.37)
glass	69.84(8.57)	68.08(9.24)	68.19(10.23)
cleveland	55.7(6.35)	53.39(7.28)	54.08(3.36)
olitos	81.25(9.08)	65.75(12.07)	68.25(11.29)
ozone	93.71(0.92)	92.48(1.34)	93.13(1.33)
libras	80.67(5.62)	69.36(8.34)	54.61(9.8)
arrhymythia	58.37(3.75)	65.78(5.75)	70.55(5.42)
water2	82.28(4.47)	81.59(6.48)	82.26(6.8)
water3	84.82(4.48)	83.18(5.47)	82.1(4.81)
web	37.97(4.31)	57.63(11.25)	55.57(13.23)
secom	92.72(0.74)	89.49(1.97)	92.52(1.03)
soybean	91.2(3.16)	91.78(3.17)	91.88(3.03)
segment	94.95(1.67)	95.71(1.84)	93.23(2.08)
vote	93.08(3.68)	96.57(2.55)	95.61(2.79)
ionosphere	82.74(5.72)	86.13(6.17)	86.78(7.43)
credit-g	72.21(3.24)	71.25(3.15)	71.92(3.65)
breastcancer	73.13(5.51)	74.28(6.02)	71.37(6.62)
multifeat	97.97(0.94)	94.62(1.68)	92.17(1.84)
waveform	77.67(1.78)	75.25(1.89)	79.14(1.7)
sonar	83.76(8.46)	73.61(9.3)	75.06(8.64)

can take values from 0 to 1. Best results tend to be obtained when β is set to a value between 0.8 and 0.9, therefore two sets of results are presented in each of Tables 4.7 – 4.8. The execution time for the two algorithms proposed in this work and that for the existing FS methods are reported in Table 4.9. Note that in each table, the figures presented in bold typeface indicate a result that is statistically significant.

In terms of classification accuracy, although all three classifiers return slightly different results for the same dataset, there are no statistically significant differences amongst them. When compared with the other approaches with respect to the JRIP classifier, both GBFG methods outperform the others for eleven of the twenty datasets. For the remainder, GHC-FS is statistically significant for the datasets *olitos* while FGFR is statistically significant for the *soybean* dataset. The remainder are all statistically comparable. Thus, the proposed methods are the overall winner. In terms of the classification results for J48, overall, the proposed GBFG-based algorithms also outperform others for half of twenty datasets. In particular, for the

Table 4.5: Comparison with other FS methods: average classification accuracies (%(sd)) for the JRIP classifier

	GBFG-HS	GBFG-FS	FRFG		GA-FS	PSO-FS	GHC-FS
			$\beta = 0.8$	$\beta = 0.9$			
heart	**78.89(7.21)**	76.63(8.61)	**79.41(7.13)**	**79.22(6.94)**	**79.37(6.56)**	**79.41(6.58)**	**78.81(6.89)**
glass	**66.41(7.89)**	**66.43(9.26)**	**67.23(9.10)**	**67.13(9.13)**	**66.85(9.17)**	65.51(9.03)	66.89(9.03)
cleveland	53.18(2.68)	53.78(3.65)	53.92(3.32)	54.56(3.29)	**55.23(3.31)**	**55.53(3.81)**	**55.13(3.21)**
olitos	65.00(12.91)	65.50(12.54)	67.00(13.13)	66.42(13.11)	64.67(12.09)	64.42(14.26)	**69.42(12.20)**
ozone	93.33(1.12)	93.13(1.27)	93.28(1.22)	93.15(1.06)	93.19(1.02)	92.83(1.18)	93.29(1.21)
libras	51.67(12.51)	49.67(9.01)	53.94(7.89)	53.31(8.90)	**54.14(8.69)**	**54.08(7.99)**	52.94(8.05)
arrhymythia	**69.69(6.86)**	65.77(6.77)	**70.34(6.11)**	**70.34(6.11)**	**69.47(5.72)**	**70.38(5.97)**	**71.13(5.41)**
water2	82.56(6.26)	**84.15(5.65)**	82.44(5.17)	83.77(5.15)	82.97(5.95)	82.56(5.80)	83.38(5.41)
water3	**82.31(6.78)**	80.95(6.06)	**82.49(5.49)**	**82.46(6.37)**	**82.74(6.87)**	81.54(6.13)	**82.56(6.69)**
web	**57.64(11.16)**	56.20(11.24)	54.44(12.69)	54.29(12.91)	55.16(11.17)	51.39(12.27)	56.17(12.19)
secom	**93.36(0.32)**	**93.06(0.76)**	**92.47(1.12)**	91.08(1.48)	**92.69(1.00)**	**92.51(1.21)**	**93.20(0.57)**
soybean	76.13(5.10)	74.49(5.17)	**84.65(6.12)**	**83.90(6.08)**	69.57(5.97)	82.05(5.82)	80.00(3.49)
segment	92.47(1.30)	92.16(3.47)	93.84(1.71)	93.84(1.71)	93.15(2.45)	93.59(2.01)	93.94(1.91)
vote	95.64(3.94)	95.42(3.08)	95.52(2.78)	95.70(2.74)	95.47(2.68)	95.40(2.77)	95.61(2.73)
ionosphere	**86.09(6.74)**	84.52(7.75)	**86.65(7.63)**	**86.65(7.63)**	85.04(6.58)	83.74(7.76)	**87.04(7.59)**
credit-g	70.60(5.95)	72.25(3.89)	71.36(3.88)	71.39(4.50)	71.74(3.83)	72.41(3.99)	72.49(4.40)
breastcancer	71.37(6.63)	70.93(5.90)	71.62(6.14)	71.24(7.03)	70.96(6.96)	71.24(6.58)	71.33(6.53)
multifeat	**86.90(2.88)**	72.46(6.18)	**87.10(4.57)**	**87.50(3.45)**	82.32(2.84)	81.53(2.94)	**88.04(2.57)**
waveform	76.64(2.08)	**78.37(2.10)**	**78.35(1.84)**	**78.19(2.12)**	76.88(2.07)	75.66(2.20)	**78.09(1.92)**
sonar	71.64(9.92)	**72.83(9.97)**	**73.81(9.72)**	**74.73(9.88)**	71.51(9.40)	**74.93(9.82)**	**74.93(9.82)**

Table 4.6: Comparison with other FS methods: average classification accuracies(%(sd)) for the J48 classifier

	GBFG-HS	GBFG-FS	FRFG		GA-FS	PSO-FS	GHC-FS
			$\beta = 0.8$	$\beta = 0.9$			
heart	**82.22(5.47)**	78.04(8.52)	78.93(7.31)	78.81(7.30)	79.19(7.35)	79.22(7.33)	78.81(7.54)
glass	**68.14(7.99)**	**69.78(8.70)**	**69.65(9.20)**	**69.84(9.14)**	**69.84(9.26)**	**68.50(9.15)**	**69.65(9.23)**
cleveland	53.23(8.17)	**55.40(6.61)**	53.65(6.93)	54.13(7.64)	**55.13(6.75)**	**55.11(6.93)**	54.90(6.49)
olitos	62.50(13.75)	**65.25(11.85)**	**65.25(12.65)**	63.50(12.80)	62.17(13.37)	63.08(13.41)	**65.17(11.81)**
ozone	93.21(0.67)	93.27(1.06)	93.11(1.11)	93.06(1.11)	93.17(1.17)	92.59(1.24)	93.10(1.08)
libras	65.83(7.75)	62.28(8.71)	**68.44(7.90)**	**68.08(8.61)**	66.33(8.44)	66.67(6.96)	64.00(8.01)
arrhymythia	64.39(5.43)	61.58(7.28)	**67.44(6.54)**	**67.44(6.54)**	66.59(5.67)	66.46(5.83)	66.56(5.83)
water2	**84.62(6.51)**	83.03(4.75)	83.85(5.26)	**83.87(5.38)**	83.79(5.40)	82.82(5.07)	83.67(4.98)
water3	79.49(3.63)	80.23(6.59)	81.92(5.88)	82.41(6.52)	81.20(6.16)	81.44(6.18)	81.13(5.70)
web	**56.38(11.89)**	55.98(12.11)	52.96(12.66)	54.29(12.57)	55.40(12.87)	53.21(14.45)	55.70(11.78)
secom	**93.30(0.33)**	**93.06(0.87)**	91.39(1.28)	92.36(1.11)	91.10(1.72)	90.01(2.03)	**93.16(0.84)**
soybean	80.10(4.87)	78.51(4.96)	**85.55(5.43)**	**85.32(5.05)**	69.46(6.52)	82.45(5.43)	83.18(3.95)
segment	95.20(1.63)	94.06(3.23)	95.97(1.78)	95.97(1.78)	95.49(1.84)	95.38(1.88)	95.84(1.76)
vote	95.87(3.54)	96.27(2.94)	96.46(2.58)	96.50(2.56)	96.55(2.62)	96.60(2.58)	96.43(2.60)
ionosphere	**88.26(5.81)**	85.04(7.78)	87.65(7.74)	87.65(7.74)	86.13(6.47)	84.74(7.19)	**88.96(6.46)**
credit-g	72.80(4.29)	72.20(3.43)	71.55(3.32)	71.80(3.25)	72.03(3.26)	72.47(3.56)	72.43(3.36)
breastcancer	**73.40(7.72)**	72.96(6.01)	**73.22(5.95)**	72.98(6.45)	70.40(6.26)	70.12(6.29)	72.48(6.32)
multifeat	86.75(3.59)	75.30(5.52)	**89.45(4.21)**	**89.50(5.78)**	84.72(2.24)	83.88(2.54)	**89.70(2.17)**
waveform	75.00(1.83)	**76.16(1.87)**	**76.27(2.03)**	**76.37(1.97)**	74.95(2.28)	73.12(2.41)	**76.09(2.03)**
sonar	69.17(11.70)	74.20(10.13)	74.82(8.66)	73.58(10.54)	73.86(9.90)	**75.68(9.82)**	**75.68(9.82)**

Table 4.7: Comparison with other FS methods: average classification accuracies (%(sd)) for the IBk (k=3) classifier

	GBFG-HS	GBFG-FS	FRFG		GA-FS	PSO-FS	GHC-FS
			$\beta = 0.8$	$\beta = 0.9$			
heart	**81.85(6.64)**	77.33(8.16)	79.67(7.34)	79.52(7.26)	79.48(7.76)	79.48(7.76)	78.74(8.03)
glass	**75.69(10.21)**	71.60(9.01)	73.58(9.90)	73.58(9.83)	72.92(9.86)	70.88(9.83)	72.92(9.98)
cleveland	53.21(7.21)	53.92(6.65)	54.87(6.37)	**55.21(6.57)**	54.18(6.20)	54.12(6.96)	53.78(5.88)
olitos	70.00(17.66)	76.42(11.11)	**77.33(10.53)**	76.58(10.71)	74.00(10.68)	76.83(9.52)	75.92(11.90)
ozone	93.25(1.01)	93.45(1.01)	93.70(0.93)	93.70(0.98)	93.50(1.09)	93.63(0.88)	93.56(1.01)
libras	76.39(6.31)	71.58(7.48)	77.22(6.50)	77.33(6.74)	75.86(6.28)	**79.11(5.65)**	76.31(6.26)
arrhymythia	**63.51(5.32)**	62.37(6.00)	61.24(4.74)	61.24(4.74)	60.70(5.18)	59.60(4.34)	61.39(4.49)
water2	83.08(4.05)	84.46(5.14)	86.49(5.42)	**87.15(5.30)**	86.95(5.19)	85.26(4.56)	87.31(5.04)
water3	81.79(3.72)	80.23(5.63)	85.23(4.67)	**84.56(5.00)**	84.87(4.94)	82.77(4.51)	84.77(4.65)
web	**58.12(8.26)**	57.28(8.46)	43.20(9.91)	43.88(10.90)	38.43(7.06)	39.71(8.93)	56.85(13.00)
secom	92.47(1.07)	92.29(1.63)	92.92(0.64)	93.63(1.89)	92.25(1.09)	92.72(0.82)	92.73(0.88)
soybean	73.93(4.68)	76.61(4.86)	**82.68(5.95)**	**82.09(6.12)**	66.97(5.38)	79.87(5.09)	78.42(4.01)
segment	94.67(1.54)	94.49(3.62)	95.58(1.78)	95.58(1.78)	94.80(1.80)	94.95(1.87)	94.94(1.63)
vote	**95.64(3.13)**	94.51(3.67)	93.47(3.40)	93.50(3.59)	94.28(3.22)	94.48(3.70)	94.05(3.57)
ionosphere	**86.52(4.32)**	85.70(7.50)	84.70(7.40)	84.70(7.40)	83.04(6.58)	82.87(7.74)	85.04(6.61)
credit-g	71.90(4.25)	72.17(3.47)	71.11(3.29)	71.45(3.34)	71.74(3.64)	72.38(3.16)	72.00(3.75)
breastcancer	70.62(5.87)	72.25(5.94)	72.98(5.35)	72.66(5.69)	71.27(5.86)	71.20(5.85)	72.25(5.59)
multifeat	91.40(2.94)	82.41(4.95)	93.85(2.44)	**95.50(2.14)**	92.40(1.70)	88.66(2.35)	93.36(1.86)
waveform	75.98(2.72)	**79.49(2.40)**	78.69(1.98)	78.89(2.20)	76.71(2.41)	74.56(2.56)	78.04(2.14)
sonar	81.31(7.82)	81.13(9.52)	81.58(8.29)	82.13(8.71)	80.98(9.09)	82.59(8.42)	82.59(8.42)

Table 4.8: Comparison with other FS methods: average feature subset size (cardinality(sd))

	GBFG-HS	GBFG-FS	FRFG		GA-FS	PSO-FS	GHC-FS
			$\beta = 0.8$	$\beta = 0.9$			
heart	**6.60(1.43)**	7.18(1.63)	10.89(1.46)	10.80(1.31)	9.60(0.49)	9.61(0.49)	9.08(2.11)
glass	**5.50(0.85)**	6.35(0.98)	6.78(0.50)	6.80(0.55)	6.73(0.53)	7.60(0.64)	6.73(0.53)
cleveland	**6.70(1.16)**	**6.84(1.41)**	11.15(1.64)	11.26(1.62)	8.03(0.67)	8.85(0.48)	7.27(2.38)
olitos	**7.80(1.62)**	11.47(2.04)	11.69(2.65)	11.96(3.44)	9.30(0.67)	12.92(1.52)	10.03(1.14)
ozone	**12.40(0.97)**	21.31(8.22)	34.18(9.94)	34.29(9.97)	19.38(1.43)	39.04(5.09)	19.76(2.86)
libras	**8.60(0.52)**	15.19(2.96)	46.20(17.31)	51.10(18.78)	17.50(3.37)	45.00(6.85)	16.87(1.73)
arrhymythia	**8.90(1.60)**	14.49(3.29)	54.24(58.85)	54.24(58.85)	30.12(2.64)	160.66(18.38)	22.14(1.99)
water2	**4.50(1.58)**	**5.09(3.36)**	17.87(4.26)	18.26(4.51)	12.03(1.04)	23.79(2.05)	12.56(1.09)
water3	**3.20(0.42)**	5.51(2.66)	14.83(5.00)	15.41(4.89)	10.23(1.32)	21.98(2.96)	10.57(1.05)
web	**6.98(0.74)**	18.34(1.56)	435.96(562)	386.62(416)	987.96(159)	940.99(162)	17.68(1.41)
secom	**2.10(1.10)**	8.09(2.71)	125.50(147)	91.00(1.78)	94.40(10.06)	401.61(42.25)	3.57(7.03)
soybean	**10.80(1.99)**	13.99(2.24)	19.53(5.16)	18.47(4.30)	10.49(0.63)	18.93(2.81)	12.48(0.72)
segment	**7.00(1.15)**	11.94(2.53)	9.11(2.07)	9.11(2.07)	7.22(0.63)	8.82(1.35)	7.85(0.83)
vote	**5.30(0.82)**	8.38(2.36)	13.04(2.34)	12.93(2.33)	8.53(0.58)	9.86(1.10)	8.66(1.57)
ionosphere	**6.40(1.07)**	8.53(1.85)	9.09(4.42)	9.09(4.42)	9.14(1.63)	11.24(1.82)	7.64(1.26)
credit-g	**10.60(1.51)**	14.32(1.51)	13.37(1.86)	13.56(1.81)	12.18(0.58)	13.65(0.94)	12.62(0.76)
breastcancer	**7.00(0.47)**	8.31(0.93)	8.06(0.40)	8.01(0.54)	7.08(0.27)	7.08(0.27)	7.99(0.48)
multifeat	**6.90(0.88)**	13.00(1.46)	7.70(0.48)	8.00(0.00)	43.00(0.00)	28.00(0.00)	**6.46(0.50)**
waveform	**11.70(1.57)**	13.42(1.16)	13.29(1.62)	13.38(2.16)	11.14(0.90)	17.58(1.87)	**11.68(0.68)**
sonar	**9.20(1.40)**	13.46(1.85)	22.36(8.09)	22.16(9.10)	11.89(1.17)	12.63(1.32)	12.63(1.32)

Table 4.9: Comparison with other FS methods: average execution times (millisecond) per cross-validation fold

	GBFG-HS	GBFG-FS	FRFG		GA-FS	PSO-FS	GHC-FS
			$\beta = 0.8$	$\beta = 0.9$			
heart	2.30E+02	**4.00E+00**	3.120E+02	6.61E+02	1.00E+02	8.40E+01	8.00E+00
glass	1.81E+02	**2.00E+00**	1.180E+02	2.15E+02	1.00E+01	1.26E+02	3.90E+01
cleveland	2.38E+02	**8.00E+00**	3.800E+02	9.68E+02	3.60E+01	9.40E+01	2.90E+01
olitos	1.53E+02	**7.00E+00**	2.410E+02	6.11E+02	4.17E+02	9.30E+01	1.70E+01
ozone	5.21E+03	**6.34E+02**	7.713E+05	2.72E+06	8.36E+04	1.79E+03	2.90E+03
libras	5.10E+02	**4.90E+01**	6.907E+04	1.40E+05	8.10E+03	3.92E+02	5.10E+02
arrhymythia	9.59E+02	**5.30E+01**	5.833E+05	1.35E+06	2.35E+04	2.91E+03	8.56E+02
water2	2.95E+02	**3.00E+00**	5.307E+03	5.56E+04	5.09E+03	2.06E+02	1.37E+02
water3	3.29E+02	**3.00E+00**	5.311E+03	4.33E+04	4.13E+03	1.82E+02	7.00E+01
web	1.41E+06	1.41E+06	3.602E+06	7.32E+06	**6.75E+02**	1.53E+04	4.60E+03
secom	1.33E+03	**8.39E+02**	2.780E+07	6.16E+07	2.05E+05	1.12E+04	1.35E+03
soybean	1.06E+03	**4.80E+01**	7.571E+03	7.55E+04	9.53E+03	2.87E+02	2.15E+02
segment	2.18E+03	**1.13E+02**	1.791E+04	1.21E+05	1.72E+03	3.13E+02	1.23E+02
vote	3.91E+02	**7.00E+00**	7.600E+02	7.77E+03	3.64E+02	1.72E+02	5.90E+01
ionosphere	2.78E+02	**4.00E+00**	1.450E+03	1.67E+04	1.20E+02	1.87E+02	7.70E+01
credit-g	1.86E+03	**1.26E+02**	7.468E+03	7.17E+04	4.36E+03	2.55E+02	1.97E+02
breastcancer	4.26E+02	**6.00E+00**	1.590E+02	1.40E+03	2.60E+01	8.10E+01	3.10E+01
multifeat	8.19E+03	**8.28E+02**	5.895E+07	1.12E+08	1.44E+02	4.01E+03	1.02E+04
waveform	3.40E+04	**1.28E+03**	1.040E+06	3.49E+06	1.66E+05	3.24E+03	3.62E+03
sonar	2.87E+02	**2.20E+01**	3.818E+03	3.70E+04	3.33E+03	7.80E+01	7.80E+01

heart and *web* datasets, GBFG-HS is the only approach that achieves statistically better performance, consistently outperforming the others. The results for the IBk classifier offer similar overall performance and this can be seen across the feature subsets returned by different FS techniques.

Table 4.8 presents the results in terms of average selected feature subset size. The statistically smallest size achieved by GBFG-HS emerges from all twenty datasets while GBFG-FS also offers better performance in reducing features for the majority of datasets when compared with GA-FS, PSO-FS, and FRFG-based FS. These results indicate that the graph-based approach is very effective in achieving compact representations. To the credit of GHS-FS, for the datasets *multifeat* and *waveform*, the average feature subset size achieved is statistically comparable to that obtained by GBFG-HS. Note that when compared with GBFG-FS, GBFG-HS not only achieves better classification accuracy but also offers even further reduction in terms of returned feature subset size for over half of the datasets. GBFG-FS, however, offers a significant reduction in runtime and this is clear from Table 4.9. It also is the most efficient amongst all of the FS methods and across all datasets, with the notable exception of the *web* dataset. For this dataset, GA-FS offers a very good execution time, but its corresponding subset size is huge when compared with the other FS methods. Although the runtime efficiency of GBFG-FS is not as good as that of GBFG-FS, it is more efficient than the other FS methods except for GHC-FS when dealing with large datasets such as *arrhymythia*, *secom*, and *multifeat*.

4.5.3 Comparison with HSFS

In order to further illustrate the potential of GBFG-HS, it is important to demonstrate that it improves upon the original harmony search based feature selection (HSFS) method (which uses the same search strategy) [54]. This section presents such a comparative analysis. Tables 4.10-4.11 show the average classification accuracies (%) for each of the three classifiers, subset sizes returned, and execution time (MS) for each of the 20 benchmark datasets. Again, bold typeface indicates a result that is statistically better than the same respective result, whilst ordinary typeface indicates a comparable result.

In terms of the classification accuracy, it can be seen that GBFG-HS offers comparable results to those of HSFS. It is when the results of the respective approaches in terms of subset size are considered in light of the classification accuracies that the

Table 4.10: Comparison of HSFS and GBFG-HS: average classification accuracy (%(sd)) for JRIP, J48 and IBk (k=3) classifiers

	GBFG-HS			HSFS		
Dataset	IBk(k=3)	J48	JRIP	IBk(k=3)	J48	JRIP
heart	**81.85(6.64)**	**82.22(5.47)**	78.89(7.21)	79.59(7.82)	79.11(7.36)	**79.19(6.86)**
glass	**75.69(10.21)**	68.14(7.99)	**66.41(7.89)**	73.01(10.01)	**70.02(9.37)**	65.98(9.79)
cleveland	53.21(7.21)	53.23(8.17)	53.18(2.68)	**54.35(6.34)**	**55.07(7.23)**	**54.79(3.07)**
olitos	70.00(17.66)	62.50(13.75)	65.00(12.91)	**77.50(10.29)**	**64.17(12.11)**	65.67(11.80)
ozone	93.25(1.01)	93.21(0.67)	93.33(1.12)	93.85(0.98)	93.09(1.14)	93.32(1.13)
libras	76.39(6.31)	65.83(7.75)	51.67(12.51)	**78.86(5.44)**	66.83(7.07)	**54.06(8.67)**
arrhymythia	**63.51(5.32)**	64.39(5.43)	69.69(6.86)	60.24(4.45)	**65.64(6.41)**	69.46(6.25)
water2	83.08(4.05)	**84.62(6.51)**	82.56(6.26)	**86.38(4.73)**	83.31(5.45)	**83.72(6.10)**
water3	81.79(3.72)	79.49(3.63)	82.31(6.78)	**83.41(5.08)**	80.95(6.70)	82.15(5.68)
web	**58.12(8.26)**	**56.38(11.89)**	**57.64(11.16)**	38.99(7.08)	55.46(11.00)	54.28(10.59)
secom	92.47(1.07)	**93.30(0.33)**	**93.36(0.32)**	92.63(1.10)	89.99(1.76)	92.48(1.11)
soybean	73.93(4.68)	80.10(4.87)	76.13(5.10)	**79.36(5.33)**	**83.12(5.05)**	**82.10(6.06)**
segment	94.67(1.54)	95.20(1.63)	92.47(1.30)	94.94(1.90)	95.50(1.89)	**93.15(2.42)**
vote	**95.64(3.13)**	95.87(3.54)	95.64(3.94)	94.14(3.35)	**96.43(2.70)**	95.52(2.87)
ionosphere	**86.52(4.32)**	**88.26(5.81)**	**86.09(6.74)**	81.48(6.70)	85.13(7.10)	84.91(7.48)
credit-g	71.90(4.25)	**72.80(4.29)**	70.60(5.95)	72.00(3.50)	71.92(3.35)	**71.76(4.10)**
breastcancer	70.62(5.87)	**73.40(7.72)**	**71.37(6.63)**	71.51(5.57)	70.26(6.34)	70.43(6.88)
multifeat	91.40(2.94)	86.75(3.59)	86.90(2.88)	**97.71(1.02)**	**93.62(1.74)**	**91.39(2.09)**
waveform	75.98(2.72)	75.00(1.83)	76.64(2.08)	76.83(2.54)	75.07(2.05)	**77.35(2.21)**
sonar	81.31(7.82)	69.17(11.70)	71.64(9.92)	**84.10(7.44)**	**74.33(10.19)**	**74.71(10.48)**

Table 4.11: Comparison of HSFS and GBFG-HS: average subset size (cardinality(sd)) and average execution time (millisecond(sd))

	GBFG-HS		HSFS	
Dataset	Subset Size	Execution Time	Subset Size	Execution Time
heart	**6.60(1.43)**	**230(49)**	9.68(0.53)	333(50)
glass	**5.50(0.85)**	**181(36)**	6.73(0.53)	262(73)
cleveland	**6.70(1.16)**	**238(44)**	8.37(0.69)	330(40)
olitos	**7.80(1.62)**	**153(45)**	13.84(1.14)	195(20)
ozone	**12.40(0.97)**	**5208(476)**	35.84(1.08)	8164(432)
libras	**8.60(0.52)**	**510(91)**	46.61(1.99)	1203(40)
arrhymythia	**8.90(1.60)**	**959(171)**	150.58(2.38)	7702(2225)
water2	**4.50(1.58)**	**295(69)**	20.39(1.11)	837(55)
water3	**3.20(0.42)**	**329(36)**	19.54(1.31)	855(87)
web	**6.98(0.74)**	**1414253(318)**	1459.65(5.51)	2226400(116)
secom	**2.10(1.10)**	**1334(146)**	326.61(3.42)	30692(1818)
soybean	**10.80(1.99)**	**1061(89)**	16.88(1.06)	1511(40)
segment	**7.00(1.15)**	**2176(266)**	8.32(0.66)	3359(442)
vote	**5.30(0.82)**	**391(86)**	9.13(0.68)	864(162)
ionosphere	**6.40(1.07)**	**278(54)**	16.13(1.18)	454(81)
credit-g	**10.60(1.51)**	**1856(277)**	12.29(0.59)	2776(108)
breastcancer	7.00(0.47)	**426(62)**	7.08(0.27)	572(115)
multifeat	**6.90(0.88)**	**8189(664)**	353.72(2.87)	54541(8564)
waveform	**11.70(1.57)**	**33977(3805)**	18.46(0.86)	39292(2371)
sonar	**9.20(1.40)**	**287(51)**	31.53(1.77)	577(55)

advantage of employing the GBFG-HS becomes clear. Of the 20 datasets, GBFG-HS offers reductions that are impressive and statistically better than those of HSFS for almost all datasets: 19 out of 20. Even for the remaining dataset, i.e., the *breast-cancer*, the results are statistically comparable. For the *web* dataset as an example, GBFG-HS offers a reduction of 99.6% over the result returned by HSFS, and 94% for the *arrhythmia* dataset. This illustrates the significantly improved performance of the GBFG-HS approach over the original HSFS, owing to the fact that it is capable of discovering compact and robust feature subsets. When comparing the execution time for each approach, it can be seen from the table that with the exception of the *web* and *waveform* datasets, GBFG-HS offers considerable speed-up in performance. For the two previous exceptions noted, the reasons for the high execution times may be related to the high dimensionality of the datasets meaning that the actual feature grouping phase takes a long time. Therefore, a more efficient algorithm mechanism for feature grouping would be desirable. However, despite this GBFG-HS

does perform better than the original HSFS algorithm [54] in terms of the average size of the selected feature subsets.

In order to gain a more useful insight into the behaviour of both a single particular process of GBFG-HS and HSFS, as the search for feature subsets progresses, a further investigation has been carried out on two of the (relatively) complex datasets: *arrhythmia* and *multifeat*. In Fig. 4.7-4.8, three plots are shown in each of the two figures: the dashed lines represent subset size, whilst the dotted lines illustrate the subset evaluation results that are computed using the probabilistic consistency measure. The solid lines indicate classification accuracy. All of these are plotted with respect to the total number of executed iterations for the evaluation (5,000), at an interval of 500.

As can be seen, the observed trend shows a logarithmic increase in evaluation score coupled with a stable and very low subset size for GBFG-HS from the outset. For *arrhythmia*, the trend is even more pronounced. Statistically, there are no significant differences in the resulting classification accuracies as the outcomes are essentially comparable. However, what is most interesting is that there is a huge difference in the trend in terms of subset size between HSFS and GBFG-HS which becomes obvious from the outset, indicating that whilst GBFG-HS may sometimes not score as well in terms of absolute classification accuracy, it always results in very compact feature subsets.

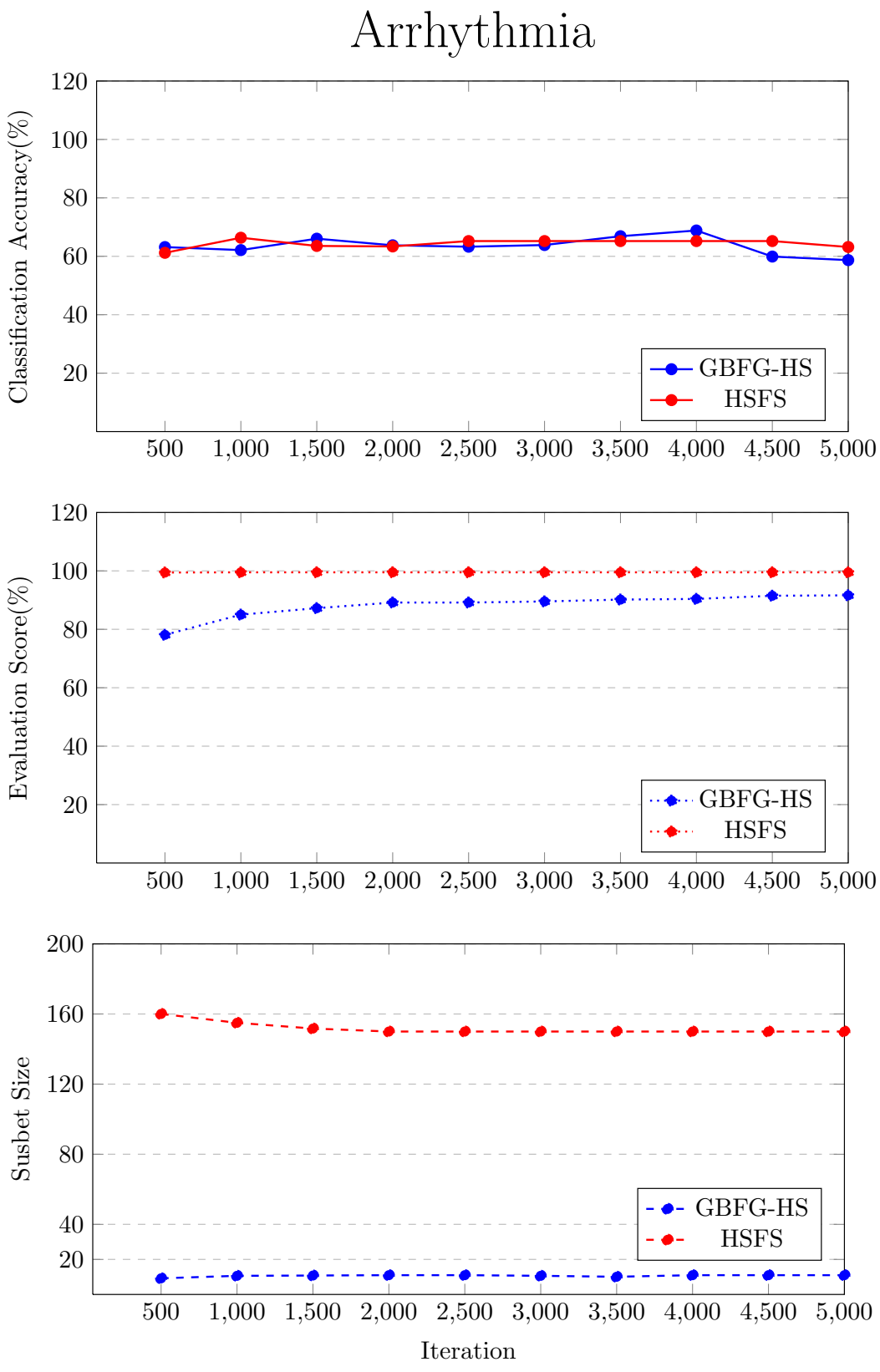

Figure 4.7: Analysis of GBFG-HS and HSFS in terms classification accuracy, subset size, and evaluation score for the arrhythmia datasets

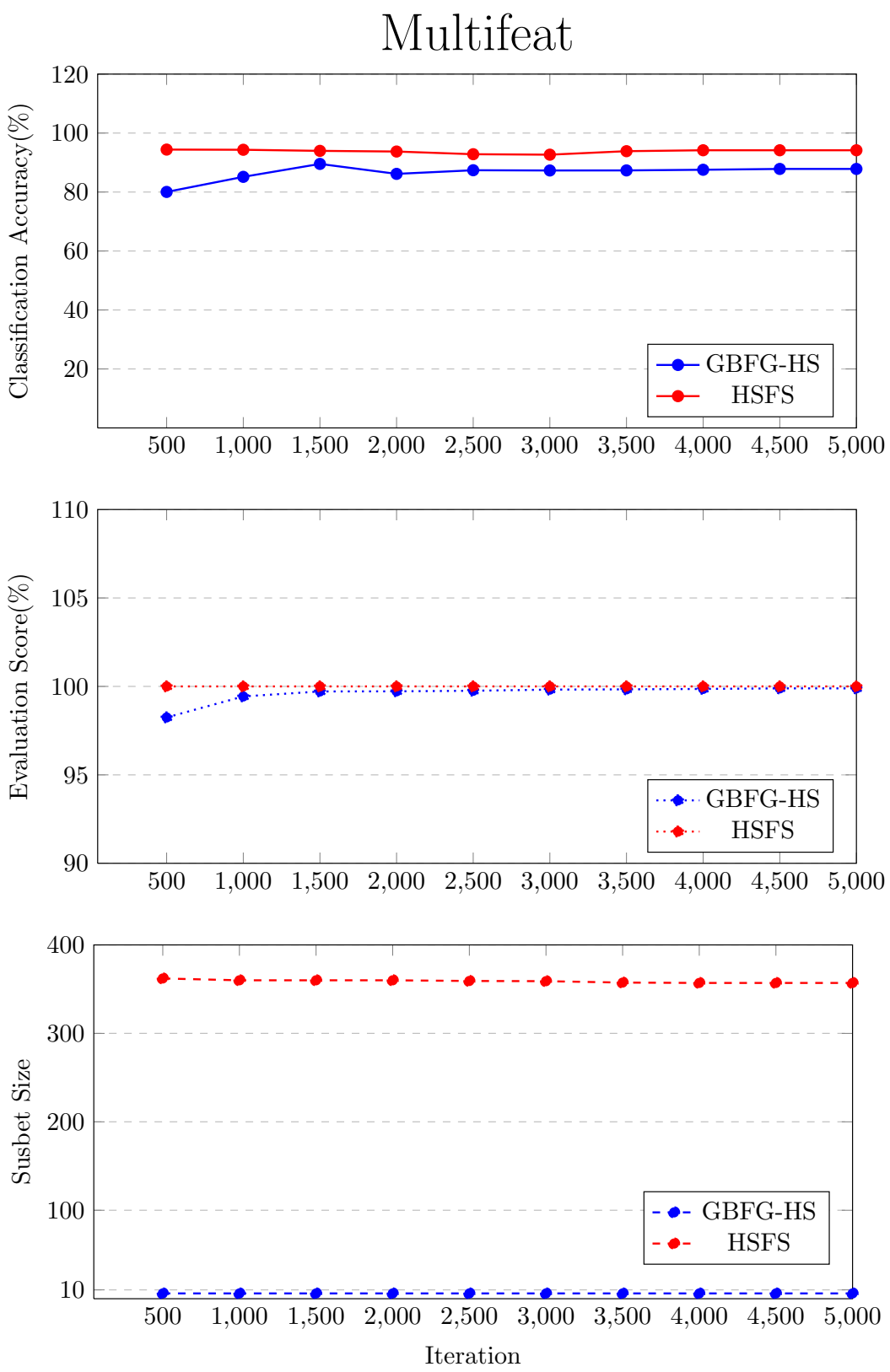

Figure 4.8: Analysis of GBFG-HS and HSFS in terms classification accuracy, subset size, and evaluation score for the multifeat datasets

4.6 Summary

This chapter has presented a novel framework for feature grouping that extends the original idea proposed in [199], upon which two instantiations for the task of feature selection are proposed. The first is a simple group-then-rank approach based on the selection of representative features from the feature groupings generated. The second, however, uses a metaheuristic approach for the search process, as the simple inclusion of feature representatives selected from feature groups may not consider information about inter-feature collaboration. In particular, harmony search has been used for the purpose of selecting the final subset. The multiple alternatives for the finally selected subset obtained using harmony search means that it offers even further flexibility.

Interestingly, other search mechanisms such as particle swarm optimisation, ant colony optimisation, and genetic algorithms are equally applicable to this particular instantiation of the framework for the task of FS. However, when compared with existing FS methods (FRFG, GA-FS, PSO-FS and GHC-FS), the proposed harmony search-based method easily outperforms these, in particular with respect to subset size across all twenty datasets investigated. Particularly, it offers significant gains over FRFG-based and PSO-based FS.

Chapter 5

Feature Selection for Intrusion Detection

A computer network communication has grown increasingly pervasive around the world, so too have various malicious network behaviours (e.g., malware plantation, illegal access and other harmful network attacks). Existing Internet security services such as firewalls and anti-virus software may be able to deny unauthorised access or locate a wide range of malware. However, network-based attacks (e.g., denial of service (DOS) [138] and its upgraded version—distributed denial of service (DDOS) [166]) are not detectable and preventable by this kind of network protection because of their underlying rationales, which are based on the sending of huge numbers of legal requests to the same destination in order to precipitate server clashes. These pose a serious threat to on-line services that require and process sensitive and confidential information (e.g., stock transactions, commercial websites, and mobile banking).

This has inspired the development of many intrusion detection systems (IDSs), including Wireshark, Metasploit, Snort and other popular systems [52]. The main challenge of these IDSs is that they have to analyse intrusions on mountains of collected network data, which may result in expensive computation. Data reduction techniques would help by removing irrelevant, redundant, and noisy information from the data. That is, the dimensionality of data is reduced while the availability of data may be improved.

5.1 Background of Intrusion Detection Systems

An intrusion detection system [157] is a defence device or software system which provides a sense of security for computers and networks. It is capable of monitoring, recognising, and preventing network attack attempts. An IDS is capable of reacting to a diverse range of abnormal network behaviours either from end-system insiders or external penetrators. Particularly, it can deal with intrusions such as DOS attacks which computer-oriented security tools struggle to sense. Of course, it may lead to false alarms as well as fail in its analysis of suspicious attack attempts. The frequency of these negative events reports varies depending on which data analysis modules are in the place and how they are designed. Most recent studies of intrusion detection are focused on the use of machine learning techniques to build these important data analysers [210, 132, 133]. As these data analysers usually suffer from the "curse of dimensionality" problem [18], feature selection has come to play a more important role in developing a lightweight, even effective IDS [5].

5.1.1 General Framework of Intrusion Detection Systems

The basic structure of intrusion detection system—including data collection, processing, and analysis—is described in Fig. 5.1. A general IDS comprises three major components: configuration, sensor, and responder. The setup of thresholds, audit rules, and modes of communication with the responder are accomplished in the configuration component. the sensor is in charge of analysing intrusions and contains decision-making mechanisms to that end. A decision made by a sensor is arrived at by analysing three major information sources: the IDS's own knowledge database, system logs, and network traffics.

An IDS knowledge database normally stores the dynamic history of complex intrusions which are planned by a sequence of network activities. System logs contain various events triggered in a computer operating system such as user login or logout, file access, and software usage. The main data source is from network traffics which are extensively collected from the protected computer network. A responder receives alerts from the sensor and directly reacts based on the predefined modes (e.g., deny requests from the suspicious hosts) or reports singularities to system security operators instead. These operators typically have abundant domain knowledge of the computer network. The core of an IDS is its sensor. A powerful sensor can not only make decisions in response to suspected intrusions but is also

capable of collecting and pre-processing raw data. With these abilities, the sensor can deal with network intrusions in real-time. Therefore, these kinds of sensors are often exclusively designed to develop an on-the-fly IDS.

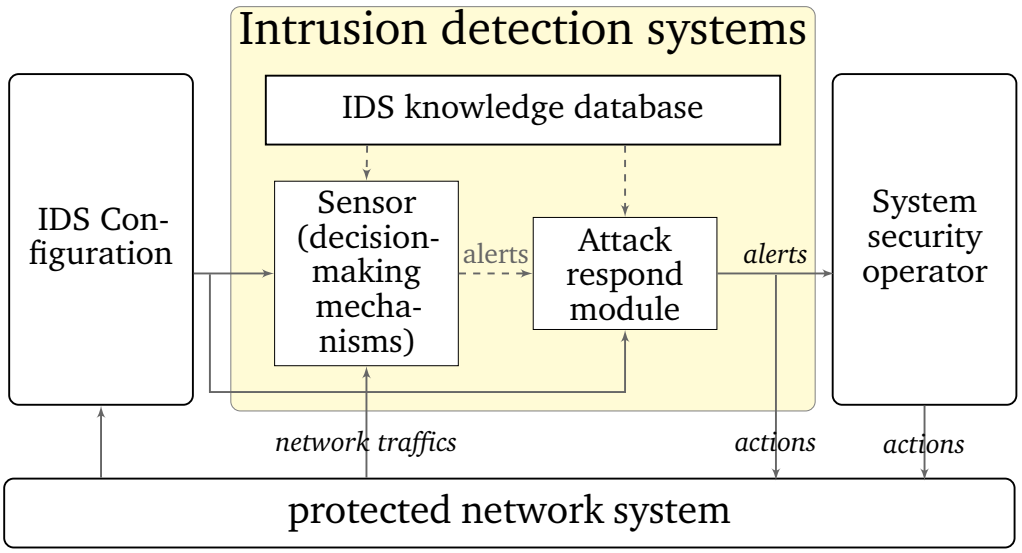

Figure 5.1: General framework of intrusion detection systems with the flowing of information

5.1.2 Taxonomy and Discussion of Intrusion Detection Systems

IDSs [67] can be categorised in a number of ways and in terms of their properties or functionalities. Fig. 5.2 presents possible categorical schemes of IDSs in terms of six aspects: detection strategy, data analysis frequency, protection domain, reaction method, data source, and architecture. For more detail, different types of IDSs, in terms of each categorical scheme, are discussed and compared as follows. Note that a practical IDS may include multiple properties and functionalities and therefore fall into a number of categories across different categorical schemes.

5.1.2.1 Anomaly-based IDS versus Signature-based IDS

Based on different detection strategies, IDSs can be divided into two groups: the first performs anomaly analysis on information extracted from normal network traffic or system activities; and the second detects attacks by matching their signatures against a library of such based on previously discovered intrusions.

Anomaly-based IDS [75] evaluates suspicious events in a system/network based on a model of its normal status. Therefore, this kind of IDS is usually trained with a

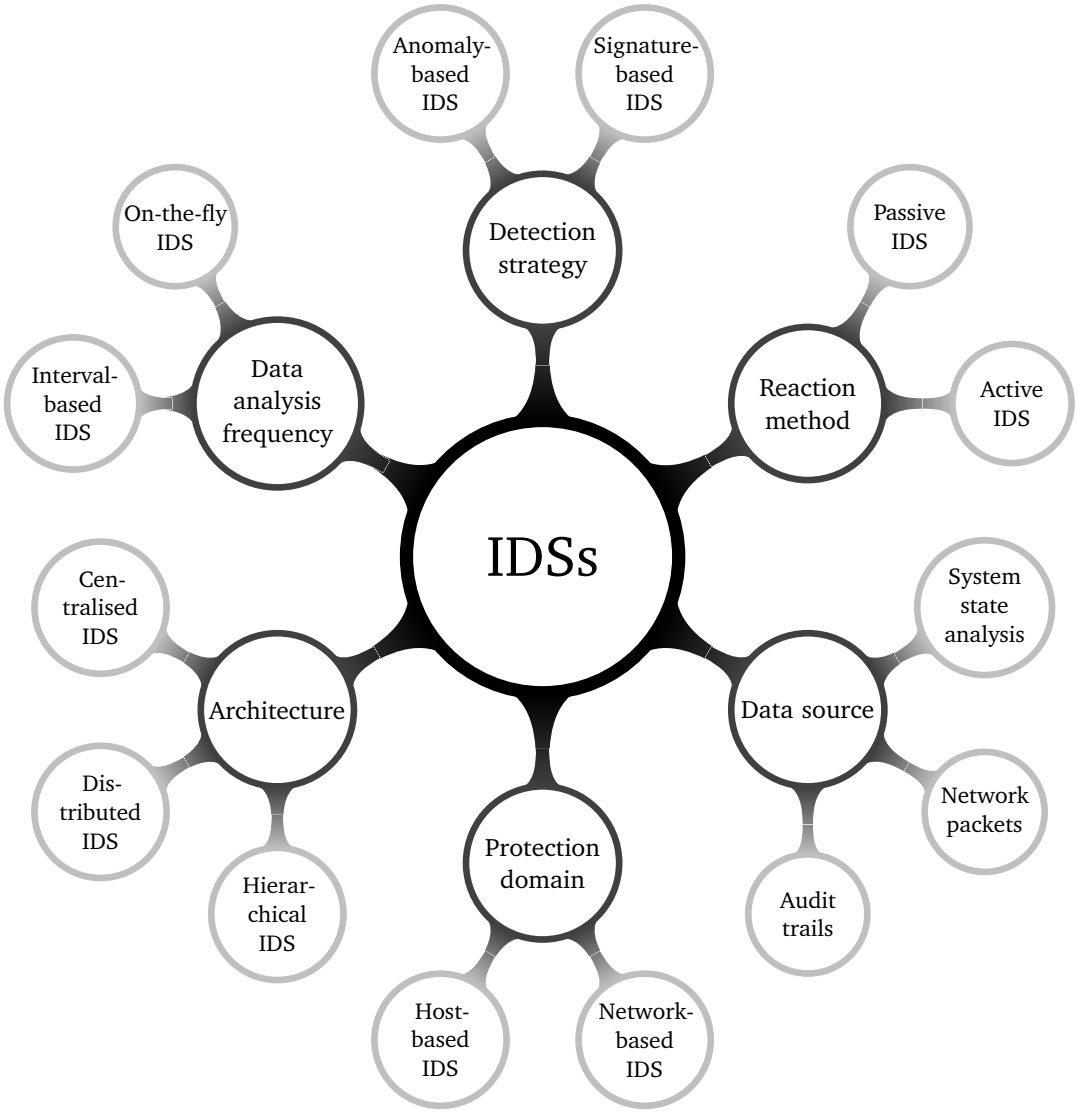

Figure 5.2: Taxonomy of intrusion detection systems

wide range of normal events in order to build a profile of normal behaviours. Any monitored traffic that is inconsistent with this profile will be classified as anomalous. A very strict profile will lead to a high false-positive rate because a good deal of normal traffic not modelled in the profile will be deemed to be system misuses or network intrusions. Another risk for anomaly detection systems is that they may be bypassed by an appropriately designed attack. To address these problems, several methods have been developed in the literature such as [171, 235].

On the other hand, a signature-based IDS [213] uses known attacks (and possible system vulnerabilities) as a posteriori knowledge. It creates a specific identity or signature (which could be a byte sequence in network traffic) for every detected attack. Any traffic event is identified as an intrusion when its signature is matched in a database of attack signatures stored by this kind of IDS. This only performs well if the database is populated by a large and up-to-date collection of attach signatures. However, as the signature dataset grows, the performance of such an IDS is inevitably impaired. More importantly, these signature engines are ineffective when dealing with unknown attacks, which may occur through disguising existing intrusions or changing their behaviour (e.g., using DNS/ICMP instead of SMTP as per the original design). Also, false positives, albeit in smaller number than in the case of anomaly-based IDSs, are occasionally triggered.

5.1.2.2 Host-based IDS versus Network-based IDS

Among IDS implementations, some are focused on protecting individual computing systems (or hosts), while others aim to guard a cloud of computing systems. An instance of the former is termed a host-based intrusion system (HIDS), and an instance of the latter a network-based intrusion detection system (NIDS).

An HIDS [230] is typically deployed on individual computing systems. Its main task is to monitor inbound and outbound traffic between the protected computing system and external connected networks, while dealing with inter-application traffic and traffic of between the operating system (OS) and applications. The amount of usable information from these data sources is, therefore, very limited in the context of a single computing system. This is why an HIDS can perform efficiently. This type of IDS also suffers from a number of limitations. A major disadvantage is that there is a risk that any compromise in the computing system itself may allow attackers to disable or temper with the IDS. A more practical problem is that the pervasive

deployment of HIDS for every computing system will lead to huge maintenance costs. This is compounded by the heterogeneity of host environments (e.g., different operating systems, and different versions of the same operating system).

Unlike an HIDS, an NIDS [216, 217] is often placed at a strategic point (which could be a router, switch, or hub) within an intranet, in an effort to monitor traffic flows passing through this point such that a cloud of hosts are protected with resort to a single IDS. Compared to HIDS, it is a more efficient way of protecting a large number of hosts. Such IDSs are very sensitive to attacks from intruders outside the intranet, but they cannot sense any attack launched within it. As network traffics are sequences of binary signals that, although exploitable for data analysis, are a semantically poor source of information about application-level events, NIDSs have to be equipped with capacities to reassemble, parse, and interpret application-level traffics such that they can access and analyse high-level information. This is more obviously the case when application-level traffic is encrypted, in which case NIDSs may be readily fooled unless they are capable of decrypting this traffic.

5.1.2.3 Passive IDS versus Active IDS

In terms of methods of reacting to suspicious attacks, intrusion detection systems can be divided into two classes: passive IDS and active IDS [13].

A passive IDS is a system that is configured merely to monitor and analyse suspicious intrusions. However, it provides no protective or corrective operations (e.g., removing suspicious files or suspending malware processes) to deal with suspicious intrusions and potential system vulnerabilities. Instead, it will quarantine these threats and issue notifications to alert system security operators. Without the mechanisms to deal with attacks, these IDSs do not threaten suicidal damage to itself and its protected host when false positives occur. However, this could be an issue for an active IDS.

An active IDS (which is also known as an intrusion prevention system) is more advanced than any passive counterpart. It is capable of not only detecting attacks but also thwarting them, and therefore it requires minimal intervention from system security operators. There are two main scenarios for actively defending attacks. The first is to sever network connections from the attacks' points of origin, and the second is to block all malicious requests directed at protected systems. The

second scenario can be readily implemented using existing functionalities included in network transmission devices, while the first scenario is technically more complicated as it needs to send requests to remove malicious activities. One optional method is to leverage third-party cooperation, typically an internet service provider (ISP), such that attack activities can be stopped by physically disabling all network connections with malicious hosts. Alternatively, DOS techniques can be adversely exploited to attack those hosts until no connections or bandwidth are available. Such methods are normally only contemplated in military or law enforcement contexts.

5.1.2.4 Centralised IDS versus Distributed IDS versus Hierarchical IDS

As for architectures of the IDS data analysis component, these fall into three categories: centralised IDS [200], Distributed IDS [197, 200], and Hierarchical IDS [203, 238].

An IDS with a centralised architecture analyses data collected from all hosts being monitored at one point while its data collection components are distributed on each monitored host. Examples of such an IDS include IDES [51], IDIOT [128], NADIR [99], and NSM [158]. These IDSs, in addition to hiring a large number of data collectors, employ a small number of main IDS components: a system configurator, a data analyser, an attack responder, and possibly an attack signature updater. As the number of monitored hosts grows, the data analyser grows in complexity, requiring larger computing and storage resources to keep up with the load. Moreover, each monitored host has its own individual characteristics and, therefore, uniform configurations of the security policies regarding them may be technically difficult to formulate. More importantly, whenever security polices are reconfigured, the entire IDS must be rebooted and this results in a temporary suspension of the monitoring of the involved hosts. During this interim, all hosts remain unprotected. This also is the case when a main component fails, which similarly requires a system reboot. However, since each monitored host, except the one responsible for analysing data, only deploys a data collector, there is very little overhead imposed on them. Additionally, centralised analysis of data collected from these hosts make it trivial to detect attacks exhibiting global behaviours.

For an IDS built in a distributed environment, a diverse range of IDS agents are deployed on different nodes (hosts) spread over a large network, including mechanical agents and smart agents. Those mechanical agents are responsible only

for collecting the underlying data corresponding to a single host within the network monitored. Each smart agent has a set of components, mainly a data analyser and a network-based information transceiver, with which to monitor a segment (e.g., an Ethernet that is a type of local area network) of the entire monitored network. The number of smart agents distributed therefore depends on the number of segments into which the monitored network is divided. A number of hosts that are configured with only a data collector for the centralised IDS may now be assigned a smart agent instead. More overhead is imposed on these hosts as a result. As each active smart agent only performs analysis on part of the data that is collected across the entire monitored network, they cannot deal with traffic in a global manner.

However, they are allowed to cross-check with each other via network-based information transceivers such that, when one or more smart agents for any reason crash, others can provisionally take over their monitoring duties. Of course, there may be a bias of monitoring effectiveness because the analysis components may differ slightly amongst smart agents (e.g., because classifiers in these analysers are trained with different data sources). This is also the case when hosts have different configurations. These smart agents can cast an alert to each other. This means that the spread of infective malwares can be prevented in advance although this may be at the cost of a fraction of hosts already being compromised with these malwares. The distributed IDSs, for example, include DIDS [197], GrIDS [203], EMERALD [173], and AAFID [14].

An hierarchical IDS is an alternative to a distributed IDS, but the set of IDS agents used in the distributed IDS are organised hierarchically. The ground layer contains numerous mechanical agents in charge of collecting data from the monitored network. In the second layer, there are a number of smart agents independently analysing a specific data source. The final layer has a single smart agent that could be functionally more powerful than agents of the second layer. It receives and aggregates reports generated by the agents of the last layer in order to globally analyse network activities. HIDE [238] is an example of an IDS with a hierarchical architecture.

5.1.2.5 Interval-based IDS versus On-the-fly IDS

In terms of the frequency with which data analysis is performed, IDSs that can be divided into two classes: a so-called interval-based IDS [185] performs analysis periodically; and an on-the-fly IDS [141] which does so continuously.

An interval-based IDS feeds information in a periodic manner and performs analysis on this information, looking for vulnerabilities and unwanted changes in the host system environment. This means that data analysis components are not required to continuously run in the background, but can be called whenever needed. This therefore reduces the overhead load of the host system. However, hosts being monitored with such an IDS are exposed in an unprotected environment during the period between two consecutive calls of the data analysis components.

Such security exposure is not an issue when using an on-the-fly IDS, which performs continuous, real-time analysis of every event taking place in host systems, or every outward and inward flow of network traffic within them. This strict monitoring leaves no suspicious activity unmonitored, and enhances the level of system security. However, such intensive analysis requires more host system resources (e.g., overheads of computation and storage space) in order to support the background running of data analysis components.

5.1.2.6 Audit Trail versus System State versus Network Traffic

An audit trail comprises a series of data sources used by IDSs, which typically includes system logs recording file modifications, usage of software/applications and current active users. Audit trails are often used by host-based IDSs because of their ease of access. However, these system logs are usually stored in a single file and likely to be rewritten using unwanted changes by intruders. Distributing a certain number of copies of such a file across and beyond the host system may help detection of unwanted changes to system logs (which will of course incur a storage overhead).

System state [103] is another data source often used by host-based IDSs. This type of data records a sequence of normal system states and the transitions they undergo when infected by a known attack, and this plays the role of a “vaccine”. In the resulting state machine, the initial state indicates the system state prior to an attack; the compromised state corresponds to the system state after a successful attack; and the intermediate states represent transitions corresponding to attack behaviours. Each intermediate state encapsulates a transition involving suspicious behaviours, and its successful detection is evidence that the system is potentially under threat because of intrusion. The earlier behaviours in question, which may pose severe systemic risk, can then be corrected in a precautionary manner and, thereby, an intrusion will fail to complete successfully.

The aforementioned data sources have advantages in handling security problems within the context of an individual host, but network-based attacks (e.g., DOS) may be missed. For this reason, network traffics are also important data source for intrusion detection. To identify a network traffic, a bit stream at a physical level or packets at higher levels can be used. A robust IDS often utilises network traffics together with audit trails for the task of intrusion detection. The main problem of using network traffics is their sheer volume. This poses the challenge of performing intrusion analysis on a wide range of network traffics, and underscores the importance of data reduction techniques to support it.

5.2 Intrusion Detection with FS

As techniques for data mining and machine learning are quite mature nowadays, many classifier learners (e.g., SVM [31], ANN [31], and CART [26]) have been used for the task of intrusion detection. However, these classifier learners must deal with mountains of network traffics or audit trails. Efficiently building classifiers for such large amounts of data therefore remains a challenge. FS, which works by removing irrelevant, redundant and noisy features while preserving the underlying semantics of data, may help in rising to it. Potentially, classifiers upon it may be much faster and detect intrusions with much greater accuracy.

5.2.1 Existing FS Approaches for Intrusion Detection

Given the benefits FS may provide, many classifier-based IDSs have used a diverse range of FS algorithms (which include various wrapper- and filter-based FS methods) to enhance the performance of intrusion detection.

In [32], three categories of feature selection algorithms—filter, wrapper, and hybrid—are thoroughly evaluated in an IDS context, and their advantages and drawbacks made plain. A wrapper-based feature selection approach using an ensemble of classifiers, including Bayesian networks (BN) and Classification and Regression Trees (CART), has been proposed in [30] to develop a lightweight, efficient, and effective IDS for real-world applications. In [206], classifiers based on support vector machines (SVM) and neural networks (NN) are used as rankers, which attempt to rank features by importance and in accordance with author-defined rules.

These wrapper methods are naturally time-consuming due to the continuous repetition of training a classifier corresponding to each generated feature subset although the classification accuracies are immediately returned for features searched. Alternatively, a filter-based feature selection that employs simple evaluation functions to gauge feature subsets instead of classifiers themselves is developed in [5]. Two measures of correlation coefficient (a linear function) and mutual information (a non-linear function) are used to evaluate candidate feature subsets and two new feature selection approaches are accordingly proposed. Furthermore, an improved SVM classifier utilising these filter-based FS methods is then applied to an IDS.

5.2.2 Applying Feature Grouping-based FS to Intrusion Detection

As investigated in Chapter 3, the new proposed self-adjusting FS ($HSFS_{SA}$) is able to locate more compact feature subsets while preserving or improving the availability of data (e.g., classifiers learned on reduced data may be more predictive) when compared with traditional popular FS methods.

However, most of the traditional FS techniques (including the method presented in Chapter 3) developed in the literature work by incrementally including/excluding an individual feature from an emerging subset or randomly selecting feature combinations without considering relationship between features. The information regarding inter-feature correlation may be lost. When applying these FS approaches to the intrusion detection problem, certain irrelevant, redundant, or noisy information may remain in the reduced data such that the derived intrusion detection systems become less efficient or even less predictive.

The grouping-based FS presented in Chapter 4 attempts to select representative features from feature groups, each of which contains features that are highly redundant. That is, it reduces massively redundant information to a smaller size, thereby improving the interpretability of data, and even possibly further reducing data dimensionality.

One of the most significant characters of feature grouping techniques is that they can deal with features missing their values. In practical applications, in particular, related to online learning), a number of features may not observe values in data

streaming and, therefore, it is impossible to use these features for the classification/prediction purposes. However, feature grouping methods can find these features a similar feature and use values of their similar features in order to accomplish the prediction of patterns of coming network traffics.

This thesis only concentrates on the FS problems or the tasks of feature grouping in the offline mode. The FS approaches proposed in this thesis may not take the full advantages of the feature groping technique to its any potential. Therefore, intrusion detection may take advantage of these newly proposed FS methods detailed in previous chapters. Hopefully, a more compact, efficient and effective classifier system may be obtained by using these FS methods. The general framework of the application of the feature grouping-based FS approach to the task of intrusion detection is described in Fig. 5.3.

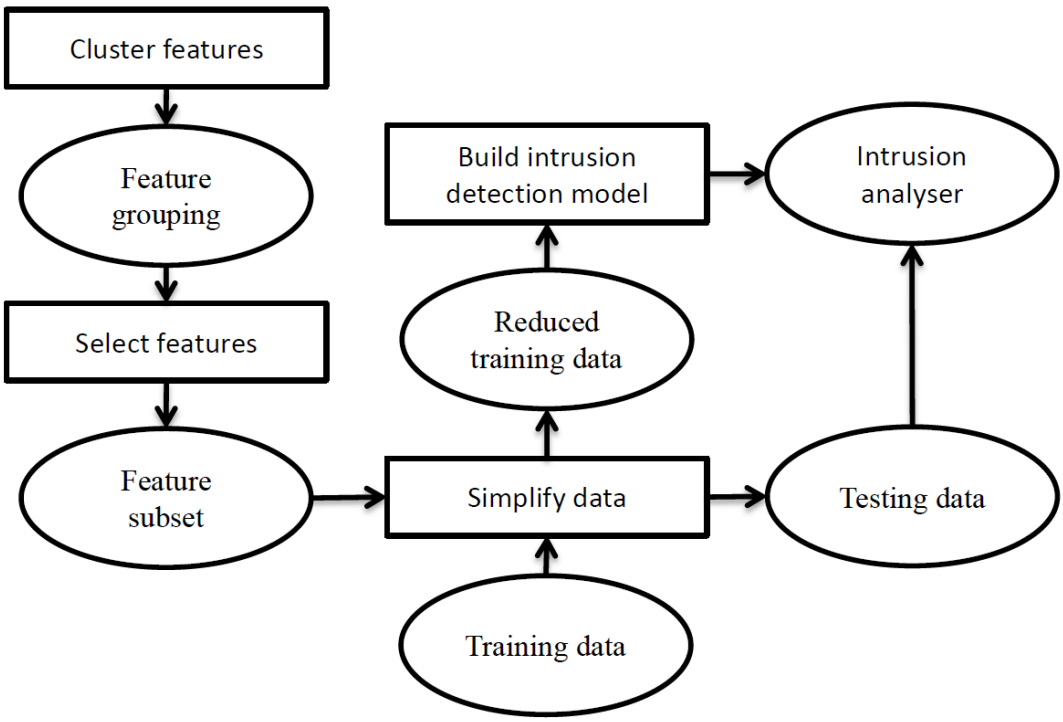

Figure 5.3: The framework of the application of the feature grouping-based FS approach to intrusion detection problem

5.3 Experimentation and Discussion

This section presents experimental evaluation of the application of the FS approaches proposed in Chapters 3-4 to the intrusion detection problem. These FS approaches involve one presented in Chapter 3 where the original HSFS is enhanced by three important improvements, leading to its a new variant (referred to as HSFS_{SA}), and two described in Chapter 4 where FS is performed on an extension of graph-based feature grouping (GBFG), leading to two FS instantiations based on a straightforward selection strategy (GBFG-FS) and a music-inspired harmony search (GBFG-HS). The setup of conducted experiments is described in Section 5.3.1. Comparison with popular FS approaches that are based upon stochastic or stepwise greedy search, covering genetic algorithm (GA-FS) [7], particle swarm optimisation (PSO-FS) [219], and greedy-hill-climbing (GHC-FS) is made in Section 5.3.2.

5.3.1 Experimental Setup

Every individual FS algorithm is set to perform 10 times and takes the average as the experiment's results. These results are quantifications of three different aspects that are the objects of experimental evaluation: classification accuracy, achieved subset size, and time taken for searching subsets. In particular, for classification analysis, a wide range of learning classifiers are used due to their availability and popularity. These include: Naïve Bayes (NB) [183], a probability-based classifier; J48 [176], a decision-tree learner; JRIP [36], a rule-based classifier; and IBk (k=3) [42], a nearest-neighbour classifier (with k=3). A paired t-test ($p = 0.05$) is used to validate the statistical significance of comparative results.

KDD99, which is a popular benchmark dataset in the UCI repository [22], has been widely used in the domain of machine learning and intrusion detection [163]. This dataset contains 41 features as described in Table 5.2–5.5, most of which are numeric while small amounts of which are nominal. The number of training instances of KDD99 is 4,898,431. Due to configuration limitations of the experimental computer, an alternative set of training instances provided with 10% of the original training dataset of the UCI repository is used for classifier learning. The original testing data has 311,029 instances, which is still employed for testing the classification accuracy of trained classifiers. The distribution of class labels of the used training data and testing data are depicted in Table 5.1. Interestingly, the testing data contains certain types of attacks that have yet emerged in the training data.

Table 5.1: Instance distribution of training and testing data used for experiments regarding classification of class labels

2-Class Category	5-Class Category	Training Data		Testing Data	
		Labels	Instances	Labels	Instances
NORMAL (1)	NORMAL (0)	normal	97278	normal	60593
ABNORMAL (0)	PROBE (1)	portsweep	1040	portsweep	354
		ipsweep	1247	ipsweep	306
		satan	1589	satan	1633
		nmap	231	nmap	84
				mscan	1053
				saint	736
	DOS (2)	neptune	107201	neptune	58001
		smurf	280790	smurf	164091
		pod	264	pod	87
		teardrop	979	teardrop	12
		land	21	land	9
		back	2203	back	1098
				apache2	794
				udpstorm	2
				processtable	759
				mailbomb	5000
	U2R (3)	buffer_overflow	30	buffer_overflow	22
		loadmodule	9	loadmodule	2
		perl	3	perl	2
		rootkit	10	rootkit	13
				xterm	13
				ps	16
				httptunnel	158
				sqlattack	2
	R2L (4)	guess_passwd	53	guess_passwd	4367
		ftp_write	8	ftp_write	3
		imap	12	imap	1
		phf	4	phf	2
		multihop	7	multihop	18
		warezclient	1020	snmpgetattack	7741
		spy	2	named	17
		warezmaster	20	warezmaster	1602
				xlock	9
				xsnoop	4
				sendmail	17
				worm	2
				snmpguess	2406
2	5	23	494021	38	311029

The parametric settings for the investigated FS methods based on stochastic search are those most used in the literature. GA-FS has an initial population size of 20, a maximum number of generations of 5000, a crossover probability of 0.6 and a mutation probability of 0.033. PSO-FS has the same settings for initial population size and maximum number of generations as GA-FS, but sets both acceleration constants c_1 and c_2 to 2. HSFS_{O} has 4 four parameters: the number of generation is set to 5000; harmony memory size is set to 20; the number of feature selectors (musicians) is equal to the number of all available features in KDD99; and harmony memory considering rate is configured to 0.8. HSFS_{SA} uses a different set of parameters including the maximum number of generations, harmony memory size, harmony memory considering rate, and pitch adjustment rate. These parameters are set to 5000, 20, 0.8, and 0.8 respectively. The maximum number of generations is set to 500 due to the introduction of iterative refinement of feature subsets while the number of musicians is equal to the number of feature groups obtained from the feature grouping process. The other parameters—harmony memory size, harmony memory considering rate, and pitch adjustment rate—are configured to 20, 0.8, and 0.8 respectively.

Note that all FS methods uniformly employ the probabilistic consistency measure [10] to evaluate the quality of feature subsets. Other subset-based evaluation methods may also be applicable, such as correlation measure [92], rough dependency [207], and fuzzy rough dependency [177]. The choice of the subset quality measure does not influence experimental evaluation for the proposed methods when applied to the intrusion detection domain. However, this would merit future investigation because some of these measures may potentially help search strategies to locate better feature subsets [53].

5.3.2 Comparison with Popular FS Methods

In order to demonstrate the viability of applying the proposed FS methods to intrusion detection, a series of comparisons are conducted against existing popular FS approaches to intrusion detection, involving GA-FS, PSO-FS, HSFS_{O}, and GHC-FS. The first three use metaheuristic strategies while the last one employs a stepwise greedy search. The performance of both the proposed and existing methods regarding classification accuracy, subset size, and execution time are reported in Table 5.6–5.8, where the figures presented in bold typeface indicate a result that is statistically significant. Also, note that: the second row of these tables presents results

Table 5.2: Basic features of individual TCP connections

Feature Name	Description	Type
1) duration	length (number of seconds) of the connection	numeric
2) protocol_type	type of the protocol, e.g., tcp, udp.	nominal
3) service	network service on the destination, e.g., http, telnet, etc.	nominal
4) src_bytes	number of data bytes from source to destination	numeric
5) dst_bytes	number of data bytes from destination to source	numeric
6) flag	normal or error status of the connection	nominal
7) land	1 if connection is from/to the same host/port; 0 otherwise	nominal
8) wrong_fragment	number of "wrong" fragments	numeric
9) urgent	number of urgent packets	numeric

Table 5.3: Content features within a connection suggested by domain knowledge

Feature Name	Description	Type
10) hot	number of "hot" indicators	numeric
11) num_failed_logins	number of failed login attempts	numeric
12) logged_in	1 if successfully logged in; 0 otherwise	nominal
13) num_compromised	number of "compromised" conditions	numeric
14) root_shell	1 if root shell is obtained; 0 otherwise	nominal
15) su_attempted	1 if "su root" command attempted; 0 otherwise	nominal
16) num_root	number of "root" accesses	numeric
17) num_file_creations	number of file creation operations	numeric
18) num_shells	number of shell prompts	numeric
19) num_access_files	number of operations on access control files	numeric
20) num_outbound_cmds	number of outbound commands in an ftp session	numeric
21) is_hot_login	1 if the login belongs to the "hot" list; 0 otherwise	nominal
22) is_guest_login	1 if the login is a "guest"login; 0 otherwise	nominal

Table 5.4: Traffic features computed using a two-second time window

Feature Name	Description	Type
23) count	number of connections to the same host as the current connection in the past two seconds **Note: The following features refer to these same-host connections**	numeric
24) serror_rate	% of connections that have "SYN" errors	numeric
25) rerror_rate	% of connections that have "REJ" errors	numeric
26) same_srv_rate	% of connections to the same service	numeric
27) diff_srv_rate	% of connections to different services	numeric
28) srv_count	number of connections to the same service as the current connection in the past two seconds **Note: The following features refer to these same-service connections**	numeric
29) srv_serror_rate	% of connections that have "SYN" errors	numeric
30) srv_rerror_rate	% of connections that have "REJ" errors	numeric
31) srv_diff_host_rate	% of connections to different hosts	numeric

before data reduction; the results in rows 3-5 are achieved by the proposed methods; and the results in rows 6-9 are obtained using existing methods.

For classification analysis, all four classifiers return different results for the same feature subset, but there are no statistically significant differences amongst them. With respect to the NB classifier, $HSFS_{SA}$ is statistically comparable with GHC-FS while outperforming other methods including $HSFS_{O}$, GA-FS, and PSO-FS. However, the grouping-based GBFG-FS and GBFG-HS barely preserve the classification accuracy that is obtained before data reduction. When compared with existing methods using J48 classifier, GBFG-HS and $HSFS_{SA}$ achieve statistically better or equal classification accuracy. GBFG-FS, albeit statistically better than existing methods except for GHC-FS, achieves the same results across 10 runs of its algorithm as well as GHC-FS. This leads to a t-test that cannot perform between GHC-FS and GBFG-FS. Therefore, the results obtained by them are comparable but not statistically comparable. As for the rest of classifiers, JRIP and IBk (k=3) comfirm alone with J48 that the proposed methods are the overall winner. This can also be seen across the feature subsets returned by different FS techniques.

In Table 5.7, the results in terms of average selected subset size are presented while subsets represented as a set of integers are also given, each of which has the

Table 5.5: Traffic features computed according to IP, service, and port of destination host

Feature Name	Description	Type
32) dst_host_count	number of connections from the same host as current connection	numeric
33) dst_host_srv_count	number of connections from the same host and same service as current connection	numeric
34) dst_host_same_srv_rate	% of connections from the same host and same service as current connection	numeric
35) dst_host_diff_srv_rate	% of connections from the same host but different services as current connection	numeric
36) dst_host_same_src_port_rate	% of connections from the same source port as current connection	numeric
37) dst_host_srv_diff_host_rate	% of connections from the different hosts but same service as current connection	numeric
38) dst_host_serror_rate	% of connections from the same host as current connection that have “S0” error	numeric
39) dst_host_srv_serror_rate	% of connections from the same host and same service as current connection that have “S0” error	numeric
40) dst_host_rerror_rate	% of connections from the same host as current connection that have “SRT” error	numeric
41) dst_host_srv_rerror_rate	% of connections from the same host and same service as current connection that have “SRT” error	numeric

best classification accuracy against the remainder nine subsets obtained by the same individual FS techniques. Interestingly, all presented subsets commonly contain both feature 5 and 6. This is possibly because these two features are of most importance in the KDD99 dataset. When compared with existing methods with respect to the average subset size, GBFG-HS achieves the statistically smallest subset size while GBFG-FS and $HSFS_{SA}$ have better data reduction than $HSFS_{O}$, GA-FS and PSO-FS, but they are (statistically) comparable with GHC-FS.

Although the performance of GHC-HS in terms of classification accuracy and subset size is slightly outdone by the proposed methods, in Table 5.8, the results demonstrate that GHC-HS is the most efficient method in comparison with all others. It is followed by PSO-FS, which is statistically worst for classification accuracy and subset size when compared with other methods. GA-FS is most time-consuming while execution time of the remainder are merely acceptable when dealing with large datasets like KDD99.

Table 5.6: Comparing with existing FS methods using classification accuracy (%(sd)) obtained from various classifiers

FS Methods	NB	J48	JRIP	IBk (k=3)
Unred.	78.08(0)	73.78(0)	91.98(0)	73.92(0)
GBFG-HS	78.17(0.66)	**92.63(0.21)**	**92.09(0.39)**	**90.97(0.71)**
GBFG-FS	78.75(0)	89.33(0)	91.63(0)	91.45(0)
$HSFS_{SA}$	**90.88(0.17)**	**92.57(0.06)**	**90.88(0.56)**	**90.99(0.67)**
$HSFS_{O}$	78.74(0.18)	79.31(0.41)	79.25(3.12)	79.16(0.32)
GA-FS	86.74(0.67)	73.78(0.19)	73.20(0.14)	91.87(0.07)
PSO-FS	86.30(0.37)	73.21(0.09)	73.63(0.17)	91.45(0.23)
GHC-FS	90.52(0)	89.56(0)	91.49(0)	90.88(0)

Table 5.7: Comparing with existing FS methods using the subset size (cardinality(sd))

FS Methods	Subset	Size
Unred.	$\{1, \cdots, 41\}$	41(0)
GBFG-HS	$\{3, 5, 6, 33, 35, 36, 39\}$	**7.6(0.6)**
GBFG-FS	$\{2, 5, 6, 12, 23, 28, 33, 35, 37, 40\}$	10(0)
$HSFS_{SA}$	$\{3, 5, 6, 12, 23, 33, 35, 37, 40\}$	8.9(0.7)
$HSFS_{O}$	$\{3, 4, 5, 6, 7, 11, 12, 15, 16, 18, 21, 38, 40\}$	11.4(0.5)
GA-FS	$\{1, 2, 3, 5, 6, 9, 12, 16, 17, 18, 21, 25, 29, 32, 33, 35, 38, 40\}$	17.6(1.7)
PSO-FS	$\{1, 2, 4, 5, 6, 7, 12, 13, 15, 21, 22, 23, 24, 32, 34, 35, 36, 37, 41\}$	19.1(0.7)
GHC-FS	$\{3, 5, 6, 12, 13, 23, 33, 35, 40\}$	9(0)

Table 5.8: Comparing with existing FS methods using execution time (millisecond(sd))

FS Methods	Execution Time
Unred.	None
GBFG-HS	1049628(1239)
GBFG-FS	1953156(3583)
$HSFS_{SA}$	2953617(18762)
$HSFS_{O}$	2495069(34922)
GA-FS	38113967(45432)
PSO-FS	260936(981)
GHC-FS	**139061(478)**

5.4 Summary

This chapter surveys a wide range of methods in the taxonomy of intrusion detection systems and then discusses the advantage and disadvantage of every variety of classified IDS. For most of IDSs, data dimensionality and its volume have been more transparently challenging. The existing data reduction methods focusing on FS are therefore reviewed. Also, the motivation for applying FS methods proposed in previous chapters is presented. More importantly, a series of experimental evaluations are conducted on dataset KDD99 in order to illustrate the potential of these newly proposed FS methods for improving the performance of IDSs based on classifiers, involving NB, J48, JRIP, and IBk.

Chapter 6

Conclusion

This chapter concludes the thesis. It gives a summary of the research presented in the preceding chapters, focusing on the main contribution. Based on a survey of the existing literature, many popular FS techniques have been developed using metaheuristics without recourse to exhaustive search technique. In particular, a recently developed HSFS algorithm has been identified to be more efficient and effective in searching quality feature subsets. To find a more compact subset remains problematic. This issue is eased by employing an iterative refinement strategy developed in [54]. The refinement process, although effective, leads to repeated executions of the entire search process. Moreover, the earlier refinements may also over-restrict the search process to a sub-optimal solution region. Potential improvements or alternative FS strategies have been proposed in this thesis, involving efficiently identifying diverse informative feature subsets, fine-tuning discovered subsets, and selecting features on feature groupings.

6.1 Self-Adjusting Harmony Search-based Feature Selection

A new so-called "self-adjusting HSFS" method is described in Chapter 3, which extends the original HSFS idea [53] with three important improvements: the concept of a restricted feature domain (RFD), harmony memory consolidation (HMC), and feature subset adjustment. An RFD which allows features to be selected by musicians

in a medium fraction of all input features effectively increases both the probability of locating informative features and the diversity of emerging feature subsets. HMC uses information stored in harmony memory to automatically configure the size of musician group for improvising new harmonies without human assumption, and is capable of dynamically adjusting the cardinality of imminent feature subsets iteratively. The feature subset adjustment, which employs the idea of PAR [81] that makes possible the incorporation of a wide range of feature similarity measures, allows emerging feature subsets to be fine-tuned.

The results of experimental evaluation show that, when compared to the original HSFS approach, the use of these enhancements dramatically improves boththe size and the classification accuracy evaluation of resulting feature subsets.

6.2 Feature Grouping-based Feature Selection using Graph-theoretic Approach

Most traditional FS methods are focused on incrementally adding/removing individual features from emerging feature subsets. This poses the risk of loss of inter-feature correlation, for example redundant and collaborative information. Feature grouping approaches allows for the inclusion of highly redundant features in the same group while reducing the level of redundancy for emerging feature subsets, which are obtained by selecting features from every single feature group.

Two new methods of FS, which are presented in Chapter 4, are implemented in the feature grouping framework where the idea of graph-based feature clustering is employed to generate feature groups. The first of the two methods is based on a straightforward strategy of selecting representative features from every emerged group to form feature subsets. This method is a sort of local-oriented search that may not take full advantage of the resulting feature groupings. The second, based on a popular stochastic search HS, is therefore proposed which may generate multiple alternatives for the final output of feature subsets while avoiding local optima. That is, it provides more flexibility in searching feature subsets.

These new methods have also been experimentally evaluated against other leading FS approaches. The results confirm that the proposed harmony search-based method easily outperforms existing FS methods (FRFG, GA-FS, PSO-FS and GHC-FS), in particular with respect to subset size across all twenty datasets investigated.

Interestingly, when compared with "flat" HSFS (which is not based on feature grouping), the use of feature grouping offers a more efficient scenario for HSFS.

6.3 Feature Selection for Intrusion Detection

The task of intrusion detection is to predict potential malicious behaviours in a computer network by analysing data from network traffic. As more and more data is extracted from network traffic, its efficient and effective analysis becomes increasingly difficult. To ease this difficulty, FS methods—a powerful tool of data reduction—have been leveraged to remove redundant, irrelevant, and noisy information, and this is documented in a large body of recent research [5, 30, 32]. Based on positive experimental evaluations against other leading FS methods, the FS approaches proposed in Chapter 3 and Chapter 4 are applied to the intrusion detection domain. When the proposed FS methods are used to process real-world dataset (KDD99), they extensively reduce the problem dimensionality, while improving the interpretability of data. This has been verified using a series of experimental evaluations. In particular, the feature index of informative features selected by the different approaches are presented in the experimental results, which provide solid and authoritative evidence of the applicability and utility of the proposed methods to intrusion detection systems.

6.4 Future Work

Although promising, much can be done to further strengthen the work currently presented in this thesis. Based on a scale of difficulty involved in addressing potential issues or implementing theoretical extensions, future plans are divided into short term tasks and long term tasks.

6.4.1 Short Term Tasks

In terms of the $HSFS_{SA}$ method presented in Chapter 3, despite its obvious value-addedness, its further refinement is desirable. As the possibility of exchanging information between musicians has been explored in the HMC procedure, there may exist alternative applications of this mechanism in order to promote high quality features, or preserve minority features. Also, these proposed improvements may be used for other nature-inspired FS search algorithms. In particular, the PAR

mechanism of HS is conceptually similar to the mutation operators used by GAs and PSO. Alternative feature similarity measures are also worth investigating, which may prove to be more efficient than fuzzy-rough set-based measures. Additionally, in order to accelerate the storing of informative features in harmony memory when improvising a new harmony, those musicians who discover no features may employ the most informative feature discovered by the other musicians as their search results.

Regarding GBFG, presented in Chapter 4, more efficient strategies for generating groupings are highly desirable. At the moment, a rather simple approach of iteratively removing edges that are weighted equally largest from the MST is employed. However, such equal-weighted edges are very rare when using three-way mutual information. In addition, this can also be time consuming particularly when combined with an iterative refinement step. A particular strategy might be to adopt a fuzzy approach where all edge weights are considered linguistically. It would also be interesting to further investigate the method used for the assessment of the quality of the feature subsets. For the current approach they are assessed by evaluating representatives drawn from every single feature group. Since the size of selected subset is controlled by the number of groups, wrapper or hybrid methods could be considered for the grouping phase. This may help to ease the computational overhead and avoid over-fitting which has traditionally been a challenge for such approaches.

6.4.2 Long Term Tasks

In many real-world applications, an information system is not always static: which may change dynamically (e.g., add/remove new instances or features) over time. Theoretical extensions to these techniques in the area of dynamic FS would be of interest for further developing this work. Alternative areas such as classifier ensemble reduction with FS that attempts to remove bad performance classifiers in a classifier ensemble are also applicable. In particular, grouping-based FS methods merit further investigation in these areas. Such methods are capable of identifying homogeneous groups of features, classifiers, or other potential objects.

In FS, many known feature subset quality evaluators are based on numerical measurements, which often normalise values into the range $[0, 1]$. When applying

such a subset quality evaluator to stochastic search-based FS, these numerical measurements may lead to lengthy algorithm convergence as a tiny quality improvement of feature subsets in the very end stage of algorithm will prolong the feature subset searching process. A fuzzy linguistic terms-based feature subset quality evaluation may therefore be used in order to achieve the earlier convergence of FS algorithms based on stochastic search. Moreover, the aggregation of the size of feature subset and the quality of feature subset into a single fuzzy linguistic term should also be investigated (e.g., using existing fuzzy T-norm operators or a newly defined fuzzy T-norm operator).

The GBFG framework itself is a general approach and is not limited to the task of conventional FS. Within it there is much potential and flexibility for application and for addressing other approaches to FS, including hierarchical feature selection where each grouping represents a hierarchy of features rather than a group of redundant features. Another possible area is semi-supervised or even unsupervised feature selection [98], where unlabelled instances could be represented by assigning them a unique class label before performing the grouping phase.

Rule-based systems are pervasive in the area of artificial intelligence (e.g., for building classifier or inference models). Many existing rule induction approaches may produce a rule base, which contains inefficient or redundant rules and hence impairs the performance of rule bases. The approaches to feature grouping-based feature selection may be used for removing these undesirable rules from induced rule bases. That is, a more compact and robust rule base may be obtained. Such a method may be worth further investigating for fuzzy control systems [8].

In real-world applications, for certain problems, the size of the problem domain will grow continuously over time. This may require the processing of streamed data instances or features in a timely fashion. Therefore, the framework for on-line feature grouping-based FS on streamed data instances or features may be a potential application area for the theoretical extension of current feature grouping-based FS approaches. This framework can be extended to applications, such as on-line multi-class classification and regression problems, or to help deal with other emerging on-line learning tasks, such as on-line transfer learning [165] or on-line AUC maximisation [239].

Appendix A

Data Sets Employed in this Thesis

Table A.1: Information of data sets used in the thesis

Data set	Feature	Instance	Class
arrhymythia	279	452	16
breastcancer	10	286	2
cleveland	14	297	5
credit-g	21	1000	2
cnae	857	1080	9
glass	10	214	6
heart	14	270	2
handwritten	256	1593	10
ionosphere	35	230	2
KDD99 (10%)	41	494021	5
libras	91	360	15
multifeat	650	2000	10
olitos	25	120	4
ozone	73	2534	2
secom	591	1567	2
soybean	35	683	19
segment	20	1500	7
sonar	60	208	2
vote	17	435	2
wine	13	178	3
water2	39	390	2
water3	39	390	3
waveform	40	5000	3
web	2556	149	5

The data sets used in the thesis are mostly public available benchmark data, available through the UCI machine learning repository [22] where datasets are drawn from real-world problem scenarios. Table A.1 offers a summary of the

properties of these data sets. Their underlying problem domains are described in detail below, attaching with the URL of the respective data sets are also given in order to facilitate easy access.

- Arrhythmia (arrhymythia)

  http://archive.ics.uci.edu/ml/datasets/Arrhythmia

  This database contains 279 attributes, 206 of which are linear valued and the rest are nominal [22]. “The aim is to distinguish between the presence and absence of cardiac arrhythmia and to classify it in one of the 16 groups. Class 01 refers to ’normal’ ECG classes 02 to 15 refers to different classes of arrhythmia and class 16 refers to the rest of unclassified ones. For the time being, there exists a computer program that makes such a classification. However there are differences between the cardiolog’s and the programs classification. Taking the cardiolog’s as a gold standard we aim to minimise this difference by means of machine learning tools.” [90]

- Breast Cancer Wisconsin (breastcancer)

  https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original

  “Instances of data are collected periodically as clinical cases are reported. The database therefore reflects this chronological grouping of the data. This grouping information appears immediately below, having been removed from the data itself:

  Group 1: 367 instances (January 1989)

  Group 2: 70 instances (October 1989)

  Group 3: 31 instances (February 1990)

  Group 4: 17 instances (April 1990)

  Group 5: 48 instances (August 1990)

  Group 6: 49 instances (Updated January 1991)

  Group 7: 31 instances (June 1991)

  Group 8: 86 instances (November 1991)

  Note that Group 1 originally contains 369 instances. For the time being, Group 1 has only 367 instances with 2 removed.” [223]

- Cleveland Heart Disease (cleveland)

  http://archive.ics.uci.edu/ml/datasets/Heart+Disease

  "This database contains 76 attributes altogether, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The decision attribute refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0). The names and social security numbers of the patients were recently removed from the database, replaced with dummy values." [83]

- Statlog: German Credit Data (credit-g)

  https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

  "This dataset classifies people described by a set of attributes as good or bad credit risks. Two versions of datasets are provided. The first dataset is described with categorical/symbolic attributes. For more general use, attributes that are ordered categorical have been coded as integer and generate variant dataset that purely contains numeric attributes." [63]

- CNAE-9 (cnae)

  https://archive.ics.uci.edu/ml/datasets/CNAE-9

  "This is a data set containing 1080 documents of free text business descriptions of Brazilian companies categorized into a subset of 9 categories catalogued in a table called National Classification of Economic Activities (CNAE). The original texts were pre-processed to obtain the current data set: initially, it was kept only letters and then it was removed prepositions of the texts. Next, the words were transformed to their canonical form. Finally, each document was represented as a vector, where the weight of each word is its frequency in the document. This data set is highly sparse (99.22% of the matrix is filled with zeros)."

  [35]

- Glass Identification (glass)

  http://archive.ics.uci.edu/ml/datasets/Glass+Identification

10 attributes are included in this dataset, the latter 8 of which describe the chemical content of glass. The second one is about the optical properties of glass. “The study of classification of types of glass (in determining whether the glass was a type of “float” glass or not) was motivated by criminological investigation. At the scene of the crime, the glass left can be used as evidence if it is correctly identified.” [66]

- Statlog: Heart (heart)

  https://archive.ics.uci.edu/ml/datasets/Statlog+(Heart)

  “This data set is a heart disease database, with 6 real-valued attributes: 1, 4, 5, 8, 10, 12; 1 ordered attribute: 11; 3 binary attributes: 2, 6, 9; and 3 nominal features: 7, 3, 13. The class label to be predicted: absence (1) or presence (2) of heart disease.” [195]

- Semeion Handwritten Digit (handwritten)

  https://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit

  “1593 handwritten digits from around 80 persons were scanned, stretched in a rectangular box 16x16 in a grey scale of 256 values.Then each pixel of each image was scaled into a boolean (1/0) value using a fixed threshold. Each person wrote on a paper all the digits from 0 to 9, twice. The commitment was to write the digit the first time in the normal way (trying to write each digit accurately) and the second time in a fast way (with no accuracy).” [27]

- Ionosphere (ionosphere)

  https://archive.ics.uci.edu/ml/datasets/Ionosphere

  “This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. See the paper for more details. The targets were free electrons in the ionosphere. “Good” radar returns are those showing evidence of some type of structure in the ionosphere. “Bad” returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal.” [193]

- (KDD99)

  https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data

  "This is the data set used for The Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 The Fifth International Conference on Knowledge Discovery and Data Mining. The competition task was to build a network intrusion detector, a predictive model capable of distinguishing between "bad" connections, called intrusions or attacks, and "good" normal connections. This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment." [204]

- Libras Movement (libras)

  https://archive.ics.uci.edu/ml/datasets/Libras+Movement

  "The data set contains 15 classes of 24 instances each, where each class references to a hand movement type in LIBRAS (Portuguese name 'LÍngua BRAsileira de Sinais', the official Brazilian signal language). In the video pre-processing, a time normalisation is carried out selecting 45 frames from each video, in according to an uniform distribution. In each frame, the centroid pixels of the segmented objects (the hand) are found, which compose the discrete version of the curve F with 45 points. All curves are normalised in the unitary space." [55]

- Multiple Features (multifeat)

  https://archive.ics.uci.edu/ml/datasets/Multiple+Features

  "This dataset consists of features of handwritten numerals ('0'–'9') extracted from a collection of Dutch utility maps. 200 patterns per class (for a total of 2,000 patterns) have been digitized in binary images. These digits are represented in terms of the following six feature sets:

  1. 76 Fourier coefficients of the character shapes
  2. 216 profile correlations
  3. 64 Karhunen-Love coefficients
  4. 240 pixel averages in 2×3 windows
  5. 47 Zernike moments

6. 6 morphological features.

The first 200 patterns are of class '0', followed by sets of 200 patterns for each of the classes '1–9'." [214]

- (olitos)

  http://michem.disat.unimib.it/chm/download/datasets.htm

  This dataset consists of 120 olive oil samples that are analysed on 25 chemical compositions (e.g., fatty acids, sterols, triterpenic alcohols) of olive oils from Tuscany, Italy (Armanino et al. 1989). There are 4 classes corresponding to 88 different production areas. Class 1, Class 2, Class 3, and Class 4 contain 50, 25, 34, and 11 observations respectively. [11]

- Ozone Level Detection (ozone)

  https://archive.ics.uci.edu/ml/datasets/Ozone+Level+Detection

  "This dataset contains 7 years (from 1998 to 2004) long ground ozone data that is collected at the Houston, Galveston and Brazoria area. 72 attributes are drawn from this data. 10 of these features have been verified to be useful and relevant for air quality control." [236]

- SECOM (secom)

  https://archive.ics.uci.edu/ml/datasets/SECOM

  "A complex modern semi-conductor manufacturing process is normally under consistent surveillance via the monitoring of signals/variables collected from sensors and or process measurement points. The measured signals contain a combination of useful information, irrelevant information as well as noise. When performing system diagnosis, engineers typically have a much larger number of signals than are actually required. The Process Engineers may then use these signals to determine key factors contributing to yield excursions downstream in the process." [149]

- (soybean)

  https://archive.ics.uci.edu/ml/datasets/Soybean+(Large)

  "There are 19 classes, only the first 15 of which have been used in prior work. The folklore seems to be that the last four classes are unjustified by the data since they have so few examples. There are 35 categorical attributes, some

nominal and some ordered. The value “dna” means does not apply. The values for attributes are encoded numerically, with the first value encoded as “0,” the second as “1,” and so forth. An unknown values is encoded as “?”.” [208]

- Image Segmentation (segment)

  https://archive.ics.uci.edu/ml/datasets/Image+Segmentation

  “The instances were drawn randomly from a database of 7 outdoor images. The images were manually segmented to create a classification for every pixel. Each instance is a 3×3 region.” [136]

- Sonar (sonar)

  https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Sonar,+Mines+vs.+Rocks)

  “This dataset contains 111 patterns obtained by bouncing sonar signals off a metal cylinder at various angles and under various conditions, and 97 patterns obtained from rocks under similar conditions. The transmitted sonar signal is a frequency-modulated chirp, rising in frequency. The data set contains signals obtained from a variety of different aspect angles, spanning 90 degrees for the cylinder and 180 degrees for the rock. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The integration aperture for higher frequencies occur later in time, since these frequencies are transmitted later during the chirp.” [89]

- Congressional Voting Records (vote)

  https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

  “This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).” [187]

- Wine (wine)

  https://archive.ics.uci.edu/ml/datasets/Wine

"These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines." The original dataset contains around 30 variables, only 13 of which somehow remains to be used. [71]

- Water Treatment Plant (water3)

  https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant

  "This dataset comes from the daily measures of sensors in a urban waste water treatment plant. The objective is to classify the operational state of the plant in order to predict faults through the state variables of the plant at each of the stages of the treatment process. This domain has been stated as an ill-structured domain." The dataset: water2 has been also used, which is derived from this dataset with 2 different class labels (which is discriminative to the original of 3). [15]

- Waveform Database Generator (waveform)

  https://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+%28Version+2%29

  "This dataset contains 40 attributes, all of which include noise. The latter 19 attributes of them are all noise attributes with mean 0 and variance 1. There are 3 classes of complex waves, each being generated from a combination of 2 of 3 base wave." [161]

- MSNBC.com Anonymous Web Data (web)

  https://archive.ics.uci.edu/ml/datasets/MSNBC.com+Anonymous+Web+Data

  "The data comes from Internet Information Server (IIS) logs for msnbc.com and news-related portions of msn.com for the entire day of 28-09-1999 (Pacific Standard Time). Each sequence in the dataset corresponds to page views of a user during that twenty-four hour period. Each event in the sequence corresponds to a user's request for a page. Requests are not recorded at the finest level of detail—that is, at the level of URL, but rather, they are recorded at the level of page category (as determined by a site administrator). The categories are "frontpage", "news", "tech", "local","opinion", "on-air", "misc", "weather", "health', "living", "business", "sports", "summary", "bbs" (bulletin

board service), “travel”, “msn-news”, and “msn-sports”. Any page requests served via a caching mechanism were not recorded in the server logs and, hence, not present in the data.”[28]

Appendix B

List of Acronyms

10-FCV	10-fold Cross-Validation
ANN	Artificial Neural Network
ABC	Artificial Bee Colony
ACO	Ant Colony Optimisation
BN	Bayesian Networks
BW	Bandwidth
CART	Classification and Regression Trees
CFS	Correlation-based Feature Selection
CSA	Clonal Selection Algorithm
DNS	Domain Name System
DOS	Denial of Service
DDOS	Distributed Denial of Service
FFS	Firefly Search
KNN (IBk)	K-Nearest Neighbour
FG	Feature Grouping
FR	Flip Rate
FRS	Fuzzy Rough Set

FRFG	Fuzzy Rough-based Feature Grouping
FS	Feature Selection
GBFG	Minimum Spanning Tree (or Graph)-based Feature Grouping
GA	Genetic Algorithm
GP	Genetic Programming
GHC	Greedy Hill-Climbing
HIDS	Host-based Intrusion System
HM	Harmony Memory
HS	Harmony Search
HMC	Harmony Memory Consolidation
HMCR	Harmony Memory Considering Rate
HSFS	Feature Selection with Harmony Search
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
ISP	Internet Service Provider
KDD	Knowledge Discovery from Data
MA	Memetic Algorithm
MDL	Minimum Description Length
MST	Minimum Spanning Tree
NB	Naïve Bayes-based Classifier
NN	Neural Networks
NIDS	Network-based Intrusion Detection System
OS	Operating System
OSCAR	Octagonal Shrinkage and Clustering Algorithm for Regression
OWA	Ordered Weighted Averaging
PAR	Pitch Adjustment Rate
PCFS	Probabilistic Consistency-based Feature Selection

PSO	Particle Swarm Optimisation
RNA	Ribonucleic Acid
RFD	Restricted Feature Domain
RS	Random Search
RST	Rough Set Theory
SA	Simulated Annealing
SD	Standard Deviation
SMTP	Simple Mail Transfer Protocol
SS	Scatter Search
SVM	Support Vector Machine

Appendix C

List of Symbols

a and a with any subscript	an individual feature
X	set of instances
X with any subscript	subset of X
A	set of conditional features
Z	set of decisional features
Y	set of features equal to the union of A and Z
S, S with any superscript	subset of A
$\mathbb{S}$	set of feature subsets
$\mathbb{S}'$	subset of $\mathbb{S}$
$\mathbb{R}$	set of real numbers
V_a	set of values taken by conditional feature a
V_Z	set of class labels taken by decisional features Z
$p()$	the probability mass function of a value taken by a feature
{ }	set notation describing a set of features
$\setminus$	set minus operator
$\cup$	set union operator
Σ	accumulation operator
$\bigcup$	the union of multiple sets
$\lvert\ \rvert$	the cardinality of a set
$\lceil\ \rceil$	operator taking the ceiling integer of a real number
[ ]	the interval operator indicating a set of continuous real numbers
C	a vague concept in fuzzy-rough set
$\Leftrightarrow$	conditional expression indicating “if and only if”

∧	logic “and” operator
∨	logic “or” operator
/	logic “not” operator
`Random( )`	randomly take a value from a set

Bibliography

[1] A. Abraham, R. Jain, J. Thomas, and S. Y. Han, "D-scids: Distributed soft computing intrusion detection system," *Journal of Network and Computer Applications*, vol. 30, no. 1, pp. 81–98, 2007.

[2] D. W. Aha, "Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms," *International Journal of Man-Machine Studies*, vol. 36, no. 2, pp. 267–287, 1992.

[3] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[4] M. Al-Betar, A. Khader, and M. Zaman, "University course timetabling using a hybrid harmony search metaheuristic algorithm," *IEEE Trans. Syst., Man, Cybern. C*, vol. 42, no. 5, pp. 664–681, 2012.

[5] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani, "Mutual information-based feature selection for intrusion detection systems," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1184 – 1199, 2011.

[6] P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, Dec 2008.

[7] P. Angelov, "Supplementary crossover operator for genetic algorithms based on the center-of-gravity paradigm," *Control and Cybernetics*, vol. 30, no. 2, pp. 159–176, 2001.

[8] ——, "A fuzzy controller with evolving structure." *Information Sciences*, vol. 161, no. 1-2, pp. 21–35, 4 2004.

[9] P. Angelov, D. P. Filev, and N. Kasabov, *Evolving intelligent sysntems: methodology ad applications*. John Wiley & Sons, 2010, vol. 12.

[10] A. Arauzo-Azofra, J. M. Benitez, and J. L. Castro, "Consistency measures for feature selection," *Journal of Intelligent Information Systems*, vol. 30, no. 3, pp. 273–292, 2008.

[11] C. Armanino, R. Leardi, S. Lanteri, and G. Modi, "Chemometric analysis of tuscan olive oils," *Chemometrics and Intelligent Laboratory Systems*, vol. 5, no. 4, pp. 343–354, 1989.

[12] W.-H. Au, K. C. Chan, A. K. Wong, and Y. Wang, "Attribute clustering for grouping, selection, and classification of gene expression data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 2, no. 2, pp. 83–101, 2005.

[13] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Technical report Chalmers University of Technology, Goteborg, Sweden, Tech. Rep., 2000.

[14] J. S. Balasubramaniyan, J. O. Garcia-Fernandez, D. Isacoff, E. Spafford, and D. Zamboni, "An architecture for intrusion detection using autonomous agents," in *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*. IEEE, 1998, pp. 13–24.

[15] L. Belanche, M. Sànchez, U. Cortés, and P. Serra, "A knowledge-based system for the diagnosis of waste-water treatment plants," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer, 1992, pp. 324–336.

[16] A. J. Bell, "The co-information lattice," in *Proceedings of the Fifth International Workshop on Independent Component Analysis and Blind Signal Separation: ICA*, vol. 2003. Citeseer, 2003.

[17] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Advanced Engineering Informatics*, vol. 18, no. 1, pp. 41–48, 2004.

[18] R. Bellman and R. Corporation, *Dynamic Programming*, ser. Rand Corporation research study. Princeton University Press, 1957. [Online]. Available: https://books.google.it/books?id=wdtoPwAACAAJ

[19] Y. Bengio and Y. Grandvalet, "No unbiased estimator of the variance of k-fold cross-validation," *The Journal of Machine Learning Research*, vol. 5, pp. 1089–1105, 2004.

[20] R. Bhattacharya and V. Patrangenaru, "Nonparametic estimation of location and dispersion on riemannian manifolds," *Journal of Statistical Planning and Inference*, vol. 108, no. 1, pp. 23–35, 2002.

[21] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[22] C. Blake and C. J. Merz, "{UCI} repository of machine learning databases," 1998.

[23] R. J. Brachman, P. G. Selfridge, L. G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D. L. McGuinness, and L. A. Resnick, "Integrated support for data archaeology," *International Journal of Intelligent and Cooperative Information Systems*, vol. 2, no. 02, pp. 159–185, 1993.

[24] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[25] L. Breiman, "Technical note: Some properties of splitting criteria," *Machine Learning*, vol. 24, no. 1, pp. 41–47, 1996.

[26] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[27] M. Buscema, "Metanet*: The theory of independent judges," *Substance use & misuse*, vol. 33, no. 2, pp. 439–461, 1998.

[28] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White, "Visualization of navigation patterns on a web site using model-based clustering," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 280–284.

[29] C. Chatfield, "Model uncertainty," *Encyclopedia of Environmetrics*, 2006.

[30] S. Chebrolu, A. Abraham, and J. P. Thomas, "Feature deduction and ensemble design of intrusion detection systems," *Computers & Security*, vol. 24, no. 4, pp. 295–307, 2005.

[31] W.-H. Chen, S.-H. Hsu, and H.-P. Shen, "Application of svm and ann for intrusion detection," *Computers & Operations Research*, vol. 32, no. 10, pp. 2617–2634, 2005.

[32] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo, "Survey and taxonomy of feature selection algorithms in intrusion detection system," in *Information security and cryptology*. Springer, 2006, pp. 153–167.

[33] Y. Chen, D. Miao, and R. Wang, "A rough set approach to feature selection based on ant colony optimization," *Pattern Recognition Letters*, vol. 31, no. 3, pp. 226–233, 2010.

[34] S. Chikhi and S. Benhammada, "Reliefmss: a variation on a feature ranking relieff algorithm," *International Journal of Business Intelligence and Data Mining*, vol. 4, no. 3-4, pp. 375–390, 2009.

[35] P. M. Ciarelli and E. Oliveira, "Agglomeration and elimination of terms for dimensionality reduction," in *2009 Ninth International Conference on Intelligent Systems Design and Applications*. IEEE, 2009, pp. 547–552.

[36] W. W. Cohen, "Fast effective induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

[37] ——, "Fast effective rule induction," in *Proceedings of the twelfth international conference on machine learning*, 1995, pp. 115–123.

[38] ——, "Pac-learning non-recursive prolog clauses," *Artificial Intelligence*, vol. 79, no. 1, pp. 1–38, 1995.

[39] P. Congdon, "Bayesian statistical modelling," 2002.

[40] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and pattern discovery on the world wide web," in *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*. IEEE, 1997, pp. 558–567.

[41] T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.

[42] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[43] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, and B. Panigrahi, "Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 41, no. 1, pp. 89–106, 2011.

[44] K. Das Sharma, A. Chatterjee, and A. Rakshit, "Design of a hybrid stable adaptive fuzzy controller employing lyapunov theory and harmony search algorithm," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 6, pp. 1440–1447, 2010.

[45] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artificial Intelligence*, vol. 151, no. 1, pp. 155 – 176, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0004370203000791

[46] L. N. De Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE transactions on evolutionary computation*, vol. 6, no. 3, pp. 239–251, 2002.

[47] M. De Cock, C. Cornelis, and E. E. Kerre, "Fuzzy rough sets: the forgotten step," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 1, pp. 121–130, 2007.

[48] J. de Jesus Rubio, P. Angelov, and J. Pacheco, "Uniformly stable backpropagation algorithm to train a feedforward neural network," *IEEE Transactions on Neural Networks*, vol. 22, no. 3, pp. 356–366, March 2011.

[49] 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.

[50] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on software engineering*, no. 2, pp. 222–232, 1987.

[51] D. E. Denning, D. Edwards, R. Jagannathan, T. Lunt, and P. Neumann, "A prototype ides: A real-time intrusiondetection expert system," *Computer Science Laboratory, SRI International*, 1987.

[52] R. Di Pietro and L. V. Mancini, *Intrusion detection systems*. Springer Science & Business Media, 2008, vol. 38.

[53] R. Diao and Q. Shen, "Two new approaches to feature selection with harmony search," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1–7.

[54] ——, "Feature selection with harmony search," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 6, pp. 1509–1523, 2012.

[55] D. Dias, R. Madeo, T. Rocha, H. Biscaro, and S. Peres, "Hand movement recognition for brazilian sign language: A study using distance-based neural networks," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, 2009, pp. 697–704.

[56] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[57] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[58] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical computer science*, vol. 344, no. 2, pp. 243–278, 2005.

[59] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem," *BioSystems*, vol. 43, no. 2, pp. 73–81, 1997.

[60] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," *Techreport, IRIDIA, Universite Libre de Bruxelles*, 2009.

[61] D. Dubois and H. Prade, *Putting rough sets and fuzzy sets together*. Intelligent Decision Support, Kluwer Academic Publishers, Dordrecht,, 1992.

[62] ——, "Rough fuzzy sets and fuzzy rough sets*," *International Journal of General System*, vol. 17, no. 2-3, pp. 191–209, 1990.

[63] J. Eggermont, J. N. Kok, and W. A. Kosters, "Genetic programming for data classification: Partitioning the search space," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1001–1005.

[64] A. Escribano and O. Jorda, *Improved testing and specification of smooth transition regression models*. Springer, 1999.

[65] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[66] I. W. Evett and E. Spiehler, "Rule induction in forensic science," *KBS in Goverment, Online Publications*, pp. 107–118, 1987.

[67] A. Fadia, *Intrusion alert: An ethical hacking guide to intrusion detection*. Vikas Publishing House Pvt Ltd, 2009.

[68] M. E. Farmer, S. Bapna, and A. K. Jain, "Large scale feature selection using modified random mutation hill climbing," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 2. IEEE, 2004, pp. 287–290.

[69] M. Fesanghary, M. Mahdavi, M. Minary-Jolandan, and Y. Alizadeh, "Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, no. 33-40, pp. 3080–3091, 2008.

[70] I. Fister, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[71] M. Forma, R. Leardi, C. Armanino, and S. Lanteri, "Parvus, an extendable package of programs for data exploration, classification and correlation," *Journal of Chemometrics*, vol. 4, no. 2, pp. 191–193, 1990.

[72] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The computer journal*, vol. 41, no. 8, pp. 578–588, 1998.

[73] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.

[74] G. M. Fung and O. L. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine learning*, vol. 59, no. 1-2, pp. 77–97, 2005.

[75] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *computers & security*, vol. 28, no. 1, pp. 18–28, 2009.

[76] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[77] Z. W. Geem, "Harmony search applications in industry," in *Soft Computing Applications in Industry*. Springer, 2008, pp. 117–134.

[78] ——, "Novel derivative of harmony search algorithm for discrete design variables," *Applied mathematics and computation*, vol. 199, no. 1, pp. 223–230, 2008.

[79] ——, "Particle-swarm harmony search for water network design," *Engineering Optimization*, vol. 41, no. 4, pp. 297–311, 2009.

[80] ——, "State-of-the-art in the structure of harmony search algorithm," in *Recent Advances In Harmony Search Algorithm*. Springer, 2010, pp. 1–10.

[81] Z. W. Geem, J. H. Kim, and G. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[82] Z. W. Geem, C.-L. Tseng, and Y. Park, "Harmony search for generalized orienteering problem: best touring in china," in *International Conference on Natural Computation*. Springer, 2005, pp. 741–750.

[83] J. H. Gennari, P. Langley, and D. Fisher, "Models of incremental concept formation," *Artificial intelligence*, vol. 40, no. 1-3, pp. 11–61, 1989.

[84] L. Getoor, *Introduction to statistical relational learning*. MIT press, 2007.

[85] F. Glover, "Tabu search-part i," *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[86] ——, "Tabu search?part ii," *ORSA Journal on computing*, vol. 2, no. 1, pp. 4–32, 1990.

[87] F. Glover, M. Laguna, and R. Martí, "Fundamentals of scatter search and path relinking," *Control and cybernetics*, vol. 29, no. 3, pp. 653–684, 2000.

[88] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[89] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural networks*, vol. 1, no. 1, pp. 75–89, 1988.

[90] H. A. Guvenir, B. Acar, G. Demiroz, and A. Cekin, "A supervised machine learning algorithm for arrhythmia analysis," in *Computers in Cardiology 1997*. IEEE, 1997, pp. 433–436.

[91] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[92] M. A. Hall and G. Holmes, "Benchmarking attribute selection techniques for discrete class data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 6, pp. 1437–1447, Nov 2003.

[93] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[94] R. W. Hamming, "Error detecting and error correcting codes," *Bell System technical journal*, vol. 29, no. 2, pp. 147–160, 1950.

[95] D. J. Hand, H. Mannila, and P. Smyth, *Principles of data mining*. MIT press, 2001.

[96] M. H. Hansen and B. Yu, "Model selection and the principle of minimum description length," *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 746–774, 2001.

[97] W. E. Hart, N. Krasnogor, and J. E. Smith, *Recent advances in memetic algorithms*. Springer Science & Business Media, 2004, vol. 166.

[98] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*. Springer, 2009, pp. 485–585.

[99] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, and J. Ford, "Nadir: An automated system for detecting network intrusion and misuse," *Computers & Security*, vol. 12, no. 3, pp. 235–248, 1993.

[100] H.-H. Hsu and C.-W. Hsieh, "Feature selection via correlation coefficient clustering," *Journal of Software*, vol. 5, no. 12, pp. 1371–1377, 2010.

[101] H.-H. Hsu, C.-W. Hsieh, and M.-D. Lu, "Hybrid feature selection by combining filters and wrappers," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8144–8150, 2011.

[102] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.

[103] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A -based intrusion detection approach," *Software Engineering, IEEE Transactions on*, vol. 21, no. 3, pp. 181–199, 1995.

[104] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[105] A. Jakulin and I. Bratko, "Quantifying and visualizing attribute interactions," *arXiv preprint cs/0308002*, 2003.

[106] R. Jensen, N. Mac Parthalain, and C. Cornells, "Feature grouping-based fuzzy-rough feature selection," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, July 2014, pp. 1488–1495.

[107] R. Jensen and Q. Shen, "Fuzzy-rough data reduction with ant colony optimization," *Fuzzy sets and systems*, vol. 149, no. 1, pp. 5–20, 2005.

[108] ——, "Fuzzy-rough sets assisted attribute selection," *IEEE Transactions on fuzzy systems*, vol. 15, no. 1, pp. 73–89, 2007.

[109] ——, "Are more features better? a response to attributes reduction using fuzzy rough sets," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1456–1458, 2009.

[110] ——, "Feature selection for aiding glass forensic evidence analysis," *Intell. Data Anal.*, vol. 13, no. 5, pp. 703–723, Oct. 2009.

[111] ——, "New approaches to fuzzy-rough feature selection," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 4, pp. 824–838, 2009.

[112] N. Johnson, "Linear statistical inference and its applications," *Technometrics*, vol. 8, no. 3, pp. 551–553, 1966.

[113] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[114] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[115] L. Ke, Z. Feng, and Z. Ren, "An efficient ant colony optimization approach to attribute reduction in rough set theory," *Pattern Recognition Letters*, vol. 29, no. 9, pp. 1351–1357, 2008.

[116] J. Kelly, P. Angelov, M. Walsh, H. Pollock, M. Pitt, P. Martin-Hirsch, and F. Martin, "A self-learning fuzzy classifier with feature selection for intelligent interrogation of mid-ir spectroscopy data from exfoliative cervical cytology using selflearning classifier eclass." *International Journal of Computational Intelligence Research*, vol. 4, no. 4, pp. 392–401, 2008.

[117] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

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

[119] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, vol. 2, 1992, pp. 129–134.

[120] R. Kohavi, "Wrappers for performance enhancement and oblivious decision graphs," Ph.D. dissertation, Citeseer, 1995.

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

[122] D. Koller and M. Sahami, "Toward optimal feature selection," 1996.

[123] I. Kononenko, "Estimating attributes: analysis and extensions of relief," in *Machine Learning: ECML-94*. Springer, 1994, pp. 171–182.

[124] I. Kononenko, E. Šimec, and M. Robnik-Šikonja, "Overcoming the myopia of inductive learning algorithms with relieff," *Applied Intelligence*, vol. 7, no. 1, pp. 39–55, 1997.

[125] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[126] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 5, pp. 474–488, 2005.

[127] M. Kryszkiewicz, "Rough set approach to incomplete information systems," *Information sciences*, vol. 112, no. 1, pp. 39–49, 1998.

[128] S. Kumar and E. H. Spafford, "A pattern matching model for misuse intrusion detection," in *In Proceedings of the 17th National Computer Security Conference*, 1994, pp. 11–21.

[129] M. G. Lagoudakis and R. Parr, "Reinforcement learning as classification: Leveraging modern classifiers," in *ICML*, vol. 3, 2003, pp. 424–431.

[130] C. Lee and G. G. Lee, "Information gain and divergence-based feature selection for machine learning-based text categorization," *Information processing & management*, vol. 42, no. 1, pp. 155–165, 2006.

[131] K. S. Lee and Z. W. Geem, "A new structural optimization method based on the harmony search algorithm," *Computers & Structures*, vol. 82, no. 9, pp. 781–798, 2004.

[132] W. Lee and S. J. Stolfo, "A framework for constructing features and models for intrusion detection systems," *ACM transactions on Information and system security (TiSSEC)*, vol. 3, no. 4, pp. 227–261, 2000.

[133] W. Lee, S. J. Stolfo *et al.*, "Data mining approaches for intrusion detection," in *Usenix security*, 1998.

[134] Z.-J. Lee, S.-F. Su, C.-C. Chuang, and K.-H. Liu, "Genetic algorithm with ant colony optimization (ga-aco) for multiple sequence alignment," *Applied Soft Computing*, vol. 8, no. 1, pp. 55–78, 2008.

[135] S. Li, X. Wu, and M. Tan, "Gene selection using hybrid particle swarm optimization and genetic algorithm," *Soft Computing*, vol. 12, no. 11, pp. 1039–1048, 2008.

[136] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[137] D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 42–50, 2007.

[138] H. Liu, "A new form of dos attack in a cloud and its avoidance mechanism," in *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*. ACM, 2010, pp. 65–76.

[139] H. Liu and H. Motoda, *Computational methods of feature selection*. Chapman & Hall/CRC, 2008.

[140] H. Liu, R. Setiono *et al.*, "A probabilistic approach to feature selection-a filter solution," in *ICML*, vol. 96. Citeseer, 1996, pp. 319–327.

[141] T. F. Lunt *et al.*, "Real-time intrusion detection," *COMPCOM Spring*, vol. 89, pp. 348–353, 1989.

[142] N. Mac Parthaláin, "Guiding rough and fuzzy-rough feature selection using alternative evaluation functions and search strategies," Ph.D. dissertation, University of Wales Aberystwyth, 2006.

[143] N. Mac Parthaláin, R. Jensen, Q. Shen, and R. Zwiggelaar, "Fuzzy-rough approaches for mammographic risk analysis," *Intell. Data Anal.*, vol. 14, no. 2, pp. 225–244, Apr. 2010.

[144] N. Mac Parthaláin and Q. Shen, "Exploring the boundary region of tolerance rough sets for feature selection," *Pattern Recognition*, vol. 42, no. 5, pp. 655–667, 2009.

[145] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[146] M. Mahdavi, M. H. Chehreghani, H. Abolhassani, and R. Forsati, "Novel meta-heuristic algorithms for clustering web documents," *Applied Mathematics and Computation*, vol. 201, no. 1-2, pp. 441–451, 2008.

[147] P. Maji, A. R. Roy, and R. Biswas, "An application of soft sets in a decision making problem," *Computers & Mathematics with Applications*, vol. 44, no. 8, pp. 1077–1083, 2002.

[148] R. A. Maronna, R. D. Martin, and V. J. Yohai, *Robust statistics*. J. Wiley, 2006.

[149] M. McCann, Y. Li, L. P. Maguire, and A. Johnston, "Causality challenge: Benchmarking relevant signal components for effective monitoring and process control," *Journal of Machine Learning Research–Proceedings Track*, vol. 6, pp. 277–288, 2010.

[150] W. J. McGill, "Multivariate information transmission," *Psychometrika*, vol. 19, no. 2, pp. 97–116, 1954.

[151] N. Memon, D. L. Hicks, and H. L. Larsen, "Notice of violation of ieee publication principles harvesting terrorists information from web," in *Information Visualization, 2007. IV'07. 11th International Conference*. IEEE, 2007, pp. 664–671.

[152] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE transactions on evolutionary computation*, vol. 6, no. 4, pp. 333–346, 2002.

[153] R. S. Michalski, "A theory and methodology of inductive learning," in *Machine learning*. Springer, 1983, pp. 83–134.

[154] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, "Machine learning: An artificial intelligence approach," 2013.

[155] P. Mitra, C. Murthy, and S. K. Pal, "Unsupervised feature selection using feature similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 301–312, 2002.

[156] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn. nc: combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 152–168, 2009.

[157] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *Network, IEEE*, vol. 8, no. 3, pp. 26–41, 1994.

[158] ——, "Network intrusion detection," *IEEE network*, vol. 8, no. 3, pp. 26–41, 1994.

[159] K. Nozaki, H. Ishibuchi, and H. Tanaka, "Adaptive fuzzy -based classification systems," *IEEE Transactions on fuzzy Systems*, vol. 4, no. 3, pp. 238–250, 1996.

[160] I.-S. Oh, J.-S. Lee, and B.-R. Moon, "Hybrid genetic algorithms for feature selection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 11, pp. 1424–1437, 2004.

[161] L. Olshen, C. J. Stone *et al.*, "Classification and regression trees," *Wadsworth International Group*, vol. 93, no. 99, p. 101, 1984.

[162] M. G. Omran and M. Mahdavi, "Global-best harmony search," *Applied mathematics and computation*, vol. 198, no. 2, pp. 643–656, 2008.

[163] A. Özgür and H. Erdem, "A review of kdd99 dataset usage in intrusion detection and machine learning between 2010 and 2015," *PeerJ Preprints*, vol. 4, pp. 1–22, 2016.

[164] S. K. Pal and A. Skowron, *Rough-fuzzy hybridization: a new trend in decision making*. Springer-Verlag New York, Inc., 1999.

[165] 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.

[166] K. Park and H. Lee, "On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets," in *ACM SIGCOMM computer communication review*, vol. 31, no. 4. ACM, 2001, pp. 15–26.

[167] Z. Pawlak, "Rough set approach to knowledge-based decision support," *European journal of operational research*, vol. 99, no. 1, pp. 48–57, 1997.

[168] ——, *Rough sets: Theoretical aspects of reasoning about data*. Springer Science & Business Media, 2012, vol. 9.

[169] Z. Pawlak, J. Grzymala-Busse, R. Slowinski, and W. Ziarko, "Rough sets," *Communications of the ACM*, vol. 38, no. 11, pp. 88–95, 1995.

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

[171] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "Mcpad: A multiple classifier system for accurate payload-based anomaly detection," *Computer Networks*, vol. 53, no. 6, pp. 864–881, 2009.

[172] G. Piateski and W. Frawley, *Knowledge discovery in databases*. MIT press, 1991.

[173] P. A. Porras and P. G. Neumann, "Emerald: Event monitoring enabling response to anomalous live disturbances," in *Proceedings of the 20th national information systems security conference*, 1997, pp. 353–365.

[174] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[175] ——, "Learning logical definitions from relations," *Machine learning*, vol. 5, no. 3, pp. 239–266, 1990.

[176] ——, *C4. 5: programs for machine learning*. Elsevier, 2014.

[177] A. M. Radzikowska and E. E. Kerre, "A comparative study of fuzzy rough sets," *Fuzzy sets and systems*, vol. 126, no. 2, pp. 137–155, 2002.

[178] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the gini index and information gain criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004.

[179] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[180] M. Robnik-Šikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICMLą́97)*, 1997, pp. 296–304.

[181] ——, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[182] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *LISA*, vol. 99, no. 1, 1999, pp. 229–238.

[183] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Upper Saddle River, 2003, vol. 2.

[184] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[185] D. Safford, D. L. Schales, D. Hess, and S. Center, "The tamu security package: An ongoing response to internet intruders in an academic environment." in *Usenix Security*, vol. 93, 1993, pp. 91–118.

[186] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.

[187] J. C. Schlimmer, "Concept acquisition through representational adjustment," 1987.

[188] C. Shang and D. Barnes, "Fuzzy-rough feature selection aided support vector machines for mars image classification," *Computer Vision and Image Understanding*, vol. 117, no. 3, pp. 202–213, 2013.

[189] Q. Shen and R. Jensen, "Selecting informative features with fuzzy-rough sets and its application for complex systems monitoring," *Pattern Recognition*, vol. 37, no. 7, pp. 1351–1363, 2004.

[190] X. Shen and H.-C. Huang, "Grouping pursuit through a regularization solution surface," *Journal of the American Statistical Association*, vol. 105, no. 490, pp. 727–739, 2010.

[191] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 69–73.

[192] W. Siedlecki and J. Sklansky, "On automatic feature selection," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 02, pp. 197–220, 1988.

[193] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, no. 3, pp. 262–266, 1989.

[194] N. Slonim, N. Friedman, and N. Tishby, "Multivariate information bottleneck," *Neural computation*, vol. 18, no. 8, pp. 1739–1789, 2006.

[195] E. Smirnov, I. Sprinkhuizen-Kuyper, and G. Nalbantov, "Unanimous voting using support vector machines," in *BNAIC-2004: Proceedings of the Sixteenth Belgium-Netherlands Conference on Artificial Intelligence*, 2004, pp. 43–50.

[196] P. Smyth and R. M. Goodman, "An information theoretic approach to induction from databases," *IEEE transactions on Knowledge and data engineering*, vol. 4, no. 4, pp. 301–316, 1992.

[197] S. R. Snapp, J. Brentano, G. V. Dias, T. L. Goan, L. T. Heberlein, C.-L. Ho, K. N. Levitt, B. Mukherjee, S. E. Smaha, T. Grance *et al.*, "Dids (distributed intrusion detection system)-motivation, architecture, and an early prototype," in *Proceedings of the 14th national computer security conference*, vol. 1. Citeseer, 1991, pp. 167–176.

[198] F. J. Solis and R. J.-B. Wets, "Minimization by random search techniques," *Mathematics of operations research*, vol. 6, no. 1, pp. 19–30, 1981.

[199] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1–14, 2013.

[200] E. H. Spafford and D. Zamboni, "Intrusion detection using autonomous agents," *Computer networks*, vol. 34, no. 4, pp. 547–570, 2000.

[201] P. M. Spira and A. Pan, "On finding and updating spanning trees and shortest paths," *SIAM Journal on Computing*, vol. 4, no. 3, pp. 375–380, 1975.

[202] R. Srinivasa Rao, S. V. L. Narasimham, M. Ramalinga Raju, and A. Srinivasa Rao, "Optimal network reconfiguration of large-scale distribution system using harmony search algorithm," *IEEE Trans. Power Syst.*, vol. 26, no. 3, pp. 1080–1088, 2011.

[203] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids-a graph based intrusion detection system for large networks," in *Proceedings of the 19th national information systems security conference*, vol. 1. Baltimore, 1996, pp. 361–370.

[204] J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection," *Results from the JAM Project by Salvatore*, 2000.

[205] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[206] A. H. Sung and S. Mukkamala, "Identifying important features for intrusion detection using support vector machines and neural networks," in *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*. IEEE, 2003, pp. 209–216.

[207] R. W. Swiniarski and A. Skowron, "Rough set methods in feature selection and recognition," *Pattern recognition letters*, vol. 24, no. 6, pp. 833–849, 2003.

[208] M. Tan and L. Eshelman, "Using weighted networks to represent classification knowledge in noisy domains," in *Proceedings of the Fifth International Conference on Machine Learning*, 1988, pp. 121–134.

[209] H. TE SUN, "Multiple mutual informations and multiple interactions in frequency data," *Information and Control*, vol. 46, pp. 26–45, 1980.

[210] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[211] G. C. Tsang, C. Degang, E. C. Tsang, J. W. Lee, and D. S. Yeung, "On attributes reduction with fuzzy rough sets," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. IEEE, 2005, pp. 2775–2780.

[212] A. Urano, T. Hirata, S. Fujino, S. Morita, M. Yamada, M. Niimura, K. Morikawa, and S. Miyazaki, "Detection method of illegal access to computer system," Mar. 13 2001, uS Patent 6,202,158.

[213] V. Vaidya, "Dynamic signature inspection-based network intrusion detection," Aug. 21 2001, uS Patent 6,279,113.

[214] M. Van Breukelen, R. P. Duin, D. M. Tax, and J. Den Hartog, "Handwritten digit recognition by combined classifiers," *Kybernetika*, vol. 34, no. 4, pp. 381–386, 1998.

[215] A. Vasebi, M. Fesanghary, and S. M. T. Bathaee, "Combined heat and power economic dispatch by harmony search algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 29, no. 10, pp. 713–719, Dec. 2007.

[216] G. Vigna and R. A. Kemmerer, "Netstat: A network-based intrusion detection approach," in *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*. IEEE, 1998, pp. 25–34.

[217] ——, "Netstat: A network-based intrusion detection system," *Journal of computer security*, vol. 7, no. 1, pp. 37–71, 1999.

[218] H. Wang, S. Kwong, Y. Jin, W. Wei, and K.-F. Man, "Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction," *Fuzzy sets and systems*, vol. 149, no. 1, pp. 149–186, 2005.

[219] 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.

[220] G. I. Webb, J. R. Boughton, and Z. Wang, "Not so naive bayes: aggregating one-dependence estimators," *Machine learning*, vol. 58, no. 1, pp. 5–24, 2005.

[221] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[222] M. A. Wiering, H. van Hasselt, A.-D. Pietersma, and L. Schomaker, "Reinforcement learning algorithms for solving classification problems," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*. IEEE, 2011, pp. 91–96.

[223] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology." *Proceedings of the national academy of sciences*, vol. 87, no. 23, pp. 9193–9196, 1990.

[224] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[225] A. D. Wood and J. A. Stankovic, "Denial of service in sensor networks," *computer*, vol. 35, no. 10, pp. 54–62, 2002.

[226] R. R. Yager, "Induced aggregation operators," *Fuzzy Sets and Systems*, vol. 137, no. 1, pp. 59–69, 2003.

[227] R. R. Yager and J. Kacprzyk, *The ordered weighted averaging operators: theory and applications*. Springer Science & Business Media, 2012.

[228] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*. Springer, 1998, pp. 117–136.

[229] X.-S. Yang, "Harmony search as a metaheuristic algorithm," in *Music-inspired harmony search algorithm*. Springer, 2009, pp. 1–14.

[230] D.-Y. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern recognition*, vol. 36, no. 1, pp. 229–243, 2003.

[231] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *ICML*, vol. 3, 2003, pp. 856–863.

[232] ——, "Efficient feature selection via analysis of relevance and redundancy," *The Journal of Machine Learning Research*, vol. 5, pp. 1205–1224, 2004.

[233] A. Zafra, M. Pechenizkiy, and S. Ventura, "Relieff-mi: An extension of relieff to multiple instance learning," *Neurocomputing*, vol. 75, no. 1, pp. 210–218, 2012.

[234] Z. Zeng, H. Zhang, R. Zhang, and C. Yin, "A novel feature selection method considering feature interaction," *Pattern Recognition*, vol. 48, no. 8, pp. 2656 – 2666, 2015.

[235] J. Zhang and M. Zulkernine, "Anomaly based network intrusion detection with unsupervised outlier detection," in *2006 IEEE International Conference on Communications*, vol. 5. IEEE, 2006, pp. 2388–2393.

[236] K. Zhang and W. Fan, "Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond," *Knowledge and Information Systems*, vol. 14, no. 3, pp. 299–326, 2008.

[237] R. Zhang and L. Hanzo, "Iterative multiuser detection and channel decoding for ds-cdma using harmony search," *Signal Processing Letters, IEEE*, vol. 16, no. 10, pp. 917–920, 2009.

[238] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proc. IEEE Workshop on Information Assurance and Security*, 2001, pp. 85–90.

[239] P. Zhao, R. Jin, T. Yang, and S. C. Hoi, "Online auc maximization," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 233–240.

[240] L. Zheng, R. Diao, and Q. Shen, "Efficient feature selection using a self-adjusting harmony search algorithm," in *2013 13th UK Workshop on Computational Intelligence (UKCI)*, Sept 2013, pp. 167–174.

[241] ——, "Self-adjusting harmony search-based feature selection," *Soft Computing*, vol. 19, no. 6, pp. 1567–1579, 2015.

[242] L. Zhong and J. Kwok, "Efficient sparse modeling with automatic feature grouping," *IEEE Trans. Neural Netw.*, vol. 23, no. 9, pp. 1436–1447, Sept 2012.

[243] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 718–729, 2009.

[244] Z. Zhu, Y.-S. Ong, and M. Dash, "Wrapper–filter feature selection algorithm using a memetic framework," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 70–76, 2007.