Discovering Patterns and Anomalies in Association Rules

by

Ping Liang, *LL.B., M.Sc.*

School of Computer Science, Engineering and Mathematics,
Faculty of Science and Engineering

January 18, 2015

A thesis presented to the
Flinders University of South Australia
in total fulfillment of the requirements for the degree of
Doctor of Philosophy

Adelaide, South Australia, 2015

Abstract

Higher order mining (HOM) [183], which mines over patterns/models derived from one or more large and/or complex datasets, has been widely used in a variety of ways and provides benefits such as the ability to combine mining strategies through the modular combination of components and the development of higher order explanations in describing facts about data. Based on the idea of HOM, this thesis addresses two important but unanswered issues.

First, while the discovery of rules that can inform business decision making is the ultimate goal of data mining technology, the search for rules that adhere to a user's definition of interesting remains somewhat elusive, in part because rules are commonly supplied in a low, instance-level format. In order to tackle this problem, this thesis proposes the concept of ruleset patterns to represent complex patterns in sets of rules reflecting a user's definition of interesting and presents a proof-of-concept system, Horace, for efficient ruleset pattern discovery. Since frequent pattern or prefix trees are (generally speaking) isomorphic with the resulting ruleset, Horace employs a novel tree-based approach to searching such intermediate data structures for patterns. Experimental results show the approach is both usable and efficient to search for rules that are sought by users.

Second, the detection of unusual or anomalous data is an important function in automated data analysis or data mining. However, the diversity of anomaly detection algorithms shows that it is often difficult to determine which algorithms might best detect anomalies given any random dataset. This thesis provides a

partial solution to this problem by elevating the search for anomalous data in transaction-oriented datasets to an inspection of the rules that can be produced by *higher order* longitudinal/spatio-temporal association rule mining. The motivation behind the approach is in two aspects. Firstly, the primary or raw data might not be always available; thus in some cases, researchers can operate only on the rules generated from the source data [183]. Furthermore, since HOM facilitates the characterisation of items participating in rulesets in terms of real-world descriptions (such as *competitor*, *catalyst* and so on), such a technique may provide a view of anomalies that is arguably closer to that sought by information analysts. In this thesis, two anomaly detection algorithms are proposed to find anomalies/outliers and a proof-of-concept prototype has been developed and tested. The experimental results demonstrate the soundness and feasibility of the proposed approach.

Certification

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Signed: Dated: January 18, 2015

Acknowledgements

I would like to express my sincere gratitude to a number of people who over the years have contributed in various ways to the completion of this work.

First, I would like to thank my supervisor Professor John Roddick. His encouragement, patience and guidance throughout my research were extremely important for me. Without his persistent help, this dissertation would not have been possible.

I would also like to thank my co-supervisor, Associate Professor Paul Calder, for his useful advice, guidance and support.

Special thanks to my previous colleagues, Dr. Denise de Vries, Dr. Anna Shillabeer, Dr. Aaron Ceglar and Dr. Carl Mooney. Their advice and shared experience were significant in solving the troubles during my study period.

My appreciation also goes to Bruce Hayter and Fay Hayter for their encouragement and help during the past years.

Most importantly, I would like to thank my family for their love and support. My heartfelt thanks go to my wife and my daughter. The thesis is dedicated to them.

Finally, this thesis was proofread by professional editor Kate Leeson. Thanks for her effort to make this thesis more readable.

Ping Liang

January 2015

Adelaide

Table of Contents

Abstract II

Certification IV

Acknowledgements V

List of Tables X

List of Figures XI

List of Algorithms XIII

1 Introduction 1

1.1 Research Context 1

1.2 Research Objectives 4

1.2.1 Discovering Patterns in Association Rules 4

1.2.2 Discovering Anomalies in Longitudinal Association Rules 6

1.3 Contribution 10

1.4 Thesis Organization 11

2 Literature Review and Background 13

2.1 Knowledge Discovery in Databases and Data Mining 13

2.2 Association Rule Mining 16

2.2.1 Formal Definition of Association Rule Mining 16

2.2.2 Apriori: Classic Association Rule Mining 18

2.3 Improvements in Frequent Itemset Generation 21

2.3.1 Candidate Generation Algorithms 22

2.3.2 Pattern Growth Algorithms 24

2.4 Improvements in Rule Generation 31

2.4.1 Constraints-Based Association Rule Mining 32
2.4.2 Removing Redundant Rules..... 33
2.5 Extensions of Association Rule Mining 36
2.5.1 Quantitative Association Rule Mining..... 36
2.5.2 Multi-level Association Rule Mining..... 38
2.5.3 Temporal Association Rule Mining 41
2.6 Mining over Association Rules 44
2.6.1 Overview of Higher Order Mining 44
2.6.2 Clustering Association Rules 47
2.6.3 Classification of Association Rules..... 50
2.6.4 Rule Changing Monitoring 54
2.6.5 Rule Maintenance..... 58
2.7 Summary 60
3 Ruleset Pattern and Horace 62
3.1 Preliminaries.....63
3.2 Defining Patterns in Rules.....65
3.3 The Horace Approach.....69
3.3.1 Overview of Horace 70
3.4 Related Work 72
3.5 Summary 74
4 Searching Ruleset Patterns Using FP-Trees and RP-Trees 76
4.1 FP-Tree 77
4.2 The Ruleset Pattern Tree (RP-Tree) 77
4.3 Algorithm Development.....81
4.3.1 The *SRPFP-a* Algorithm 81
4.3.2 The *SRPFP-b* Algorithm 84

4.4 Experiments and Analysis 89
4.4.1 Results and Performance Study 91
4.5 Summary 96
5 RPL: A Ruleset Pattern Language 98
5.1 Towards a Ruleset Pattern Language - RPL 98
5.1.1 Ruleset Pattern Definition Language (RPDL) 99
5.1.2 Ruleset Pattern Query Language (RPQL) 102
5.2 Implementation of RPQL 104
5.2.1 Indexing 105
5.2.2 Evaluating Queries 106
5.3 Experiments 108
5.4 Related Work 110
5.5 Summary 112
6 Detecting Anomalies in Longitudinal Association Rules 113
6.1 Motivation and Literature Review 114
6.1.1 Anomaly Detection 114
6.1.2 Longitudinal and Spatio-Temporal Knowledge Discovery 117
6.1.3 Motivation 121
6.2 Anomaly Detection in Longitudinal Association Rules 122
6.2.1 Longitudinal Association Rule Generation 123
6.2.2 Generation of the CS-set 123
6.2.3 Detection Process 124
6.3 Detection Algorithms 125
6.3.1 The *TARMA-a* Algorithm 125
6.3.2 The *TARMA-b* Algorithm 126
6.4 Implemented Prototype and Experiments 129

6.4.1 Synthetic Longitudinal Data 131
6.4.2 Real Data 132
6.4.3 Longitudinal Association Rule Generation 133
6.4.4 Experimental Results and Evaluation 133
6.5 Summary 139
7 Conclusion and Future Research 140
7.1 Contributions 141
7.1.1 Discovering Patterns in Association Rules 141
7.1.2 Discovering Anomalies in Longitudinal Association Rules 143
7.2 Future Research 144
Publications Resulting from This Thesis 146
Bibliography 148

List of Tables

Table 1.1: Stroke Patients Details 8

Table 1.2: Sample Association Rules 9

Table 3.1: Sample Ruleset Patterns 68

Table 4.1: Synthetic Data Parameters 90

Table 4.2: Synthetic Data 90

Table 4.3: Real Datasets 90

Table 4.4: Description of Test Pattern 91

Table 4.5: Test Results 92

Table 5.1: Notations of Ruleset Pattern p 99

Table 5.2: Sample Ruleset Patterns 106

Table 5.3: Synthetic Data 109

Table 5.4: Synthetic Data Parameters 109

Table 6.1: Synthetic Data Parameters 132

Table 6.2: Synthetic Data 132

Table 6.3: Real Data 132

Table 6.4: Test Results 134

Table 6.5: Test Results with Different rX 138

List of Figures

Figure 1.1: An Example of Anomalies in Association Rules 9

Figure 2.1: The KDD Process 14

Figure 2.2: Illustration of Apriori 20

Figure 2.3: Sample Data and Frequent 1-itemset 26

Figure 2.4: FP-Tree Construction 26

Figure 2.5: Sample Conditional FP-Tree 28

Figure 2.6: Example Food Hierarchical Structure 39

Figure 2.7: Architecture of the Association Rule Clustering System 48

Figure 3.1: Sample Ruleset 64

Figure 3.2: Overview of Horace 70

Figure 3.3: Overview of Ruleset Pattern and the Pattern Language 71

Figure 4.1: Sample RP-Tree Construction 79

Figure 4.2: Searching Algorithm Illustration (*SRPFP-a*) 82

Figure 4.3: Searching Algorithm Illustration (*SRPFP-b*) 86

Figure 4.4: RP-Tree Substitution Process 88

Figure 4.5: Test Patterns 91

Figure 4.6: Performance Comparison 93

Figure 4.7: Effect of *minH* and *maxL* 95

Figure 5.1: Execution Times of the 4 Representative Queries 110

Figure 6.1: Anomaly Detection Process 123

Figure 6.2: An Example of rXY-neighbourhood 128

Figure 6.3: Screenshots from *TARMAD* System 130

Figure 6.4: Performance – Time vs #Transactions.....134

Figure 6.5: Screenshots for Top N Anomalies in Real Data 135

Figure 6.6: Complex Data.....137

List of Algorithms

Algorithm 2.1: Apriori Itemset Generation 18

Algorithm 2.2: Apriori Rule Generation 20

Algorithm 4.1: RP-Tree Construction 79

Algorithm 4.2: *SRPFP-a* Algorithm 81

Algorithm 4.3: *SRPFP-b* Algorithm 85

Algorithm 5.1: RPQL Query Evaluation 107

Algorithm 6.1: Overarching Detection Process 124

Algorithm 6.2: *TARMA-a* Algorithm 126

Algorithm 6.3: *TARMA-b* Algorithm 129

Algorithm 6.4: *TARMA-c* Algorithm 138

Chapter 1

Introduction

1.1 Research Context

We are living in a world with a wealth of data. With the use of computers and electronic database packages, the amount of data that is collected doubles approximately every twenty months [140]. This explosive growth in data and databases generates the need for new techniques and tools that can intelligently and automatically transform the data into useful information and knowledge.

Knowledge discovery in databases (KDD), which has been defined as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data [64], is an evolving research direction to meet this challenge. The KDD process is composed of different steps that can be summarized in four main phases:

- Data cleaning and data integration, where real world data from multiple sources are cleaned and put in a coherent data store, such as data

warehouses and/or data marts.

- Data pre-processing, where data is transformed or consolidated into forms appropriate for analysis (usually termed data mining).
- Data mining, where various mining techniques are applied over the database in order to discover new patterns/knowledge.
- Post-mining, where evaluation and visualization techniques are utilized to present the mined knowledge to the user.

Association rule mining is one of the most commonly applied techniques of data mining. It aims to find interesting relationships among items in a given dataset [86]. Initial research into association rule mining was largely motivated by the analysis of retail market basket data, the results of which allowed companies to understand purchasing behaviour more fully. One example is that "customers who purchase computers also tend to buy anti-virus software at the same time". The discovery of association rules can help in many retail business decision-making processes, such as cross-selling, shelf layout, and catalogue design. Although initially motivated by the desire to analyse large retail transaction databases, the general utility of association rules makes them applicable to a wide range of different learning tasks. Association rule mining has now been applied in a variety of industry sectors including commerce, defence, health, manufacturing, exploration and engineering.

Data mining techniques extract implicit and interesting patterns from large data collections. Such data are typically assumed to be primary data captured by some application, cleaned and prepared according to the demands of the mining

algorithm [183]. However, in many cases, the primary or raw data are not always available. For example, in some applications, stream data are not stored and are only available for a short time [70]. Also, in some cases, organisations (and governments) are willing to provide (by their nature relatively confidential) association rules but unwilling to provide access to source data. Thus in some cases only association rules generated from the source data are available for the researchers to operate over [183]. Furthermore, even for available primary data, there are limits on the computation speed that can be achieved – such limits are set by hardware and firmware technologies [183].

One approach to tackling those issues is to mine over patterns/models derived from one or more large and/or complex datasets, which can be termed higher order mining (HOM) [183]. For example, Lent et al. [124] have shown how association rules may be clustered. Gupta et al. [80] extended this work by looking at distance-based clustering of association rules, and Perrizo and Denton [170] outlined a framework based on partitions to unify various forms of data mining algorithms.

Compared with traditional data mining techniques which have largely focused on the extraction of knowledge directly from the source data, HOM discovers patterns from non-primary data and has the following benefits [128]:

- the ability to combine mining strategies through the modular combination of components.
- the development of *higher order* explanations in describing facts about data, particularly those describing changes over time, location or some other dimension.

- a comparatively faster execution time due to reduced volumes of data.

1.2 Research Objectives

HOM opens a window for changes in perspective about knowledge discovery, from the analysis of data to the analysis of patterns. Although there have been many advances in this paradigm, the overall potential of HOM is still largely unexploited and worthy of further research [183].

Based on the idea of HOM, this thesis addresses two important but unanswered issues: 1) the discovery of patterns in association rules which represent the *higher order* knowledge sought by users; 2) the discovery of anomalies in association rules that are produced by *higher order* longitudinal/spatio-temporal association rule mining.

1.2.1 Discovering Patterns in Association Rules

Since the early work of Agrawal, Srikant and others, association rule mining has become a mature field. It has provided very powerful mining algorithms, with the capacity to discover rapidly sets of co-occurring items or events in very large databases. A variety of extensions have been proposed that enable, for example,

- temporal [7, 127, 175] and spatial [87, 115] semantics to be accommodated,
- closed sets to be identified [165, 232],
- fuzzy and incomplete data to be handled [49, 119],
- the accommodation of domain-specific concept hierarchies [54, 66, 84, 190], and

- the application of visualisation techniques [154].

However, the search for patterns/knowledge which adhere to a user's definition of interesting, remains somewhat elusive [74], in part because rules are generally supplied in an instance level format such as

$$\{milk\} \wedge \{butter\} => \{bread\}\ \sigma(0.20)\ \gamma(0.65) \quad (1)$$

where the σ (support) and γ (confidence) values are examples of some quality metric for the rule. Such low-level rules, while useful, provide knowledge only about the coincidence of elementary values and can be termed *zero-order* rules. *Higher order* semantics can be derived when sets of rules are inspected to determine patterns of interest between rules.

Example 1.1 Given a set of rules such that:

$$\{a\} => \{c\} \quad \sigma(x) \quad (2)$$

$$\{b\} => \{c\} \quad \sigma(y) \quad (3)$$

$$\{a, b\} => \{c\} \quad \sigma(z) \quad (4)$$

$$where\ \sigma(z) \ll \sigma(x) \times \sigma(y) \quad (5)$$

We might find two competitor items a and b from the above three rules as the observed value for Eq. (5) is considerably lower than one would have expected with independent items.

Studies of patterns reveal that users are often interested in such types of knowledge. For example, a supermarket manager may be interested in finding products that churn with each other. Analysts looking to reduce hospital costs may look for situations where potential alternatives exist, that is, pairs of items which rarely occur together but almost always occur with the same other items.

In the past, specific algorithms have been developed to search for individual cases

of such patterns. For example, Teng [206] outlined a mechanism for learning dissociations (aka *competitors*) from source data. However, direct current work on the discovery of patterns in rules is limited and there are several questions which remain unanswered:

- What are patterns in rules? How can we define them based on a user's definition of interesting?
- How can we efficiently search patterns from the discovered set of rules?
- Users need to specify high-level (i.e. user-oriented) descriptions of the patterns they are interested in. Can we develop a pattern language to enable users to create, update and retrieve such patterns?

This thesis provides some answers to the above questions. The thesis provides a formal definition of patterns in rules, based on which it proposes the Horace framework for pattern searching [130]. Horace consists of a pattern library and its associated pattern language which allows users to define, retrieve and maintain patterns in rules based on their own definition of interesting [129]. At the core of Horace is a tree-based approach to searching for patterns in rules. Since frequent patterns or prefix trees are (generally speaking) isomorphic with the resulting ruleset, Horace expresses the ruleset patterns using a novel ruleset pattern tree (RP-tree) and utilizes a set of algorithms to search such data structures for patterns efficiently and directly [130].

1.2.2 Discovering Anomalies in Longitudinal Association Rules

The popularity of data mining, together with mounting recognition of the value of temporal and spatial data, spatio-temporal data modelling and databases has

resulted in the prospect of mining spatial and temporal rules from both static and longitudinal, temporal and spatial data. Longitudinal and spatio-temporal data mining has the capacity to [128]:

- analyse activity rather than just states and to infer relationships of locational and temporal proximity, some of which may also indicate a cause-effect association, and
- mine the behavioural aspects of objects as opposed to simply mining rules that describe their states at a point in time.

In many domains, the value of knowledge obtained by analysing the changes to phenomena over time and space, as opposed to the situation at an instant or at a single location, has been recognized and a number of temporal and spatial data mining techniques have been developed [182, 61]. For example, spatio-temporal association rules can indicate movement, trends and/or patterns that static rules cannot show.

Anomaly detection is an important problem for many domains, particularly those with a high level of pre-existing domain knowledge. Within medicine, for example, it is commonly the exceptions that provide insight into a problem. It is important to be able to detect statistically significant anomalies from a series of multiple, large and semantically complex snapshots or single location datasets, such as those that could be collected by an organization as part of routine archival operations or statutory reporting. Efficiently solving this problem would enable the more rapid development of knowledge discovery systems capable of uncovering hidden spatio-temporal trends and correlations which might, in some cases, act as a real time

alerting mechanism [128].

In the past, there have been few efforts to address this problem. For example, spatio-temporal outlier detection techniques [30, 53] have been proposed to find spatial outliers over several time periods. Mooney and Roddick [148] tackled this problem by running an association mining algorithm over sets of rules which they themselves generated from association rule algorithms.

Clearly, anomalies in a single data item can be found using standard statistical techniques. This thesis is primarily concerned with the following question: "Can anomalies be detected through an inspection of association rules generated from the source data?"

To illustrate, consider the following example.

Example 1.2 Assume Table 1.1 contains information about the patients who suffered from stroke on a given day, including their name, age, blood pressure and the time stroke occurred.

Table 1.1: Stroke Patients Details

Patient Name	Age	Blood Pressure	Time
patient-a	60	150	2:00am
patient-b	69	170	2:00am
patient-c	78	165	8:00pm
patient-d	80	180	8:00am
patient-e	65	130	12:00pm
(rest of data omitted)	…	…	…

Let A represent the antecedent (age = [60…80]) ^ (blood pressure [130…180]) and B represent the consequent (stroke = yes). Table 1.2 shows some of the association rules which might be generated based on the data stored in Table 1.1. Those rules have the same rule body but a different timestamp, revealing the relationship

between the age and blood pressure of a patient and the possibility of suffering stroke at different times.

Table 1.2: Sample Association Rules

Rule Name	Support	Confidence	Time Stamp
A=>B	0.20	0.55	2:00am
A=>B	0.21	0.64	4:00am
A=>B	0.23	0.53	6:00am
A=>B	0.58	0.54	8:00am
A=>B	0.25	0.50	10:00am
A=>B	0.20	0.52	12:00pm
A=>B	0.21	0.58	14:00pm
A=>B	0.26	0.51	16:00pm
(rest of data omitted)	…	…	…

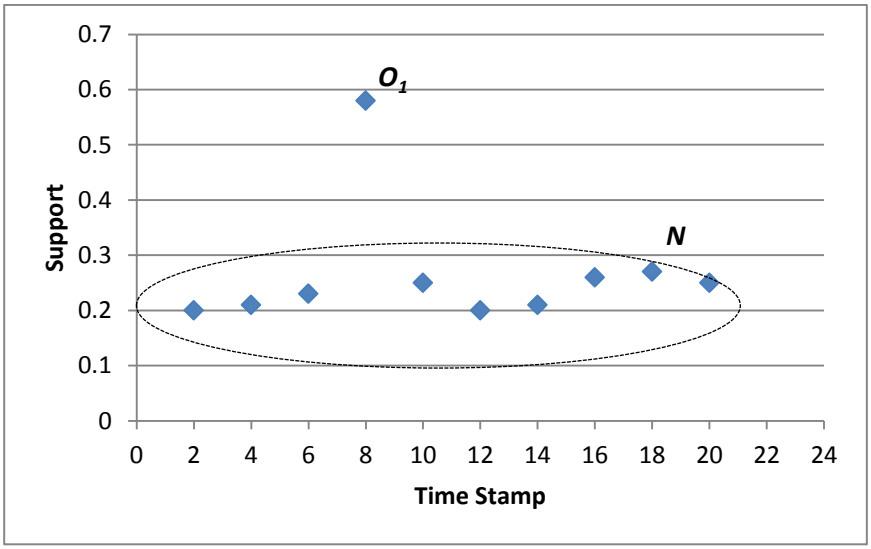

Figure 1.1: An Example of Anomalies in Association Rules

Figure 1.1 illustrates the data in Table 1.2, where the X axis is the timestamp and the Y axis is the support value. As shown in Figure 1.1, there is one normal region *N* since the support values of most of the rules are between 0.2 and 0.26 during the 24-hour period. Point O_1 which represents the rule occurred at 8am has a support value of 0.58 which is sufficiently far away from the region. If we define points in region *N* as normal, then point O_1 can be treated as an anomaly. It reveals that older patients $(60 \leq \text{age} \leq 80)$ with high blood pressure $(130 \leq blood\ pressure \leq$

180) have an unexpectedly high rate of stroke at 8am.

This above example shows the possibility of detecting anomalies through an inspection of association rules generated from the source data. Motivated by this example, this thesis proposes an approach to elevating the search for anomalous data in transaction-oriented datasets to an inspection of the rules that can be produced by *higher order* longitudinal/spatio-temporal association rule mining.

Since HOM facilitates the characterization of items participating in rulesets in terms of real-world descriptions (such as *competitor*, *catalyst* and so on), we argue that such a technique may provide a view of anomalies that is arguably closer to that sought by information analysts. In addition, it provides an alternative approach for anomaly detection if primary sources are not available but only rules generated from the source data the researchers can operate.

In this thesis, two anomaly detection algorithms have been developed and the experimental results have demonstrated the soundness and feasibility of the proposed approach [128].

1.3 Contribution

This thesis makes the following contributions to the domain:

- The concept of the ruleset pattern is developed, which represents patterns in rules. Also, a framework, called Horace, is proposed for ruleset pattern discovery. Since frequent pattern or prefix trees contain the complete set of information held in a database relevant to frequent pattern mining, Horace employs a tree-based approach to searching such data structure

directly for matches of given ruleset patterns.

- The thesis proposes a novel data structure, a ruleset pattern tree (RP-tree), to represent patterns in rules. Two tree searching algorithms are presented to search the frequent pattern tree (FP-tree) efficiently for matches of the RP-tree.
- A ruleset pattern language (RPL) has been developed, which consists of a ruleset pattern definition language (RPDL) and a ruleset pattern query language (RPQL). RPL enables users to create, alter and retrieve patterns from a ruleset pattern library.
- Two anomaly detection methods have been presented which can identify anomalies in a set of longitudinal association rules.
- Prototypes for ruleset pattern discovery and anomaly detection in longitudinal association rules have been built and tested. The experimental results demonstrate the capacity of the proposed approach to find patterns/anomalies that cannot be identified by traditional data mining techniques.

1.4 Thesis Organization

The remainder of the thesis is structured as follows:

Chapter 2 presents a systematic literature review of related work, with a focus on association rule mining and HOM.

Chapter 3 introduces the concept of ruleset patterns together with some specific patterns, including the competitor pattern, twoway-catalyst pattern, and threeway-

catalyst pattern. Also, the Horace framework of ruleset pattern discovery is explored in this chapter.

Chapter 4 presents two novel algorithms for searching FP-trees for ruleset patterns. The details of a prototype and experiment results are also supplied.

Chapter 5 describes RPL, the ruleset pattern language. Detailed description of its two components: the ruleset pattern definition language (RPDL) and the pattern query language (RPQL) have been provided. The evaluations of RPL queries as well as the experimental results are also discussed in this chapter.

Chapter 6 presents an approach for detecting anomalies in a set of longitudinal rules. Two anomaly detection algorithms have been proposed. A prototype for anomaly detection and experimental results are also explored.

Finally, the thesis concludes in Chapter 7, where recommendations are made regarding possible future research and development activities.

Chapter 2

Literature Review and Background

This chapter presents an in-depth review of the topics, areas and research related to the work presented herein. The chapter sections are arranged as follows. Section 2.1 provides an overview of knowledge discovery in databases and data mining. Section 2.2 reviews the formal definition of association rule mining and the Apriori algorithm. Section 2.3 and 2.4 discuss the improvements in frequent itemset generation and rule generation respectively while Section 2.5 provides an overview of some extensions of association rule mining. A survey of HOM is provided in Section 2.6 with a focus on mining over association rules. Section 2.7 concludes the chapter.

2.1 Knowledge Discovery in Databases and Data Mining

Since the 1960s, with the advances in computer science and databases, the volume of information stored in databases has been growing exponentially. For example, the National Aeronautics and Space Administration's (NASA's) Earth Observing

System of orbiting satellites and other space borne instruments sends one terabyte of data to receiving stations every day [218]. It has become a real and universal challenge to find actionable knowledge from such large amount of data. The field of knowledge discovery in databases (KDD) has been designed to meet this challenge.

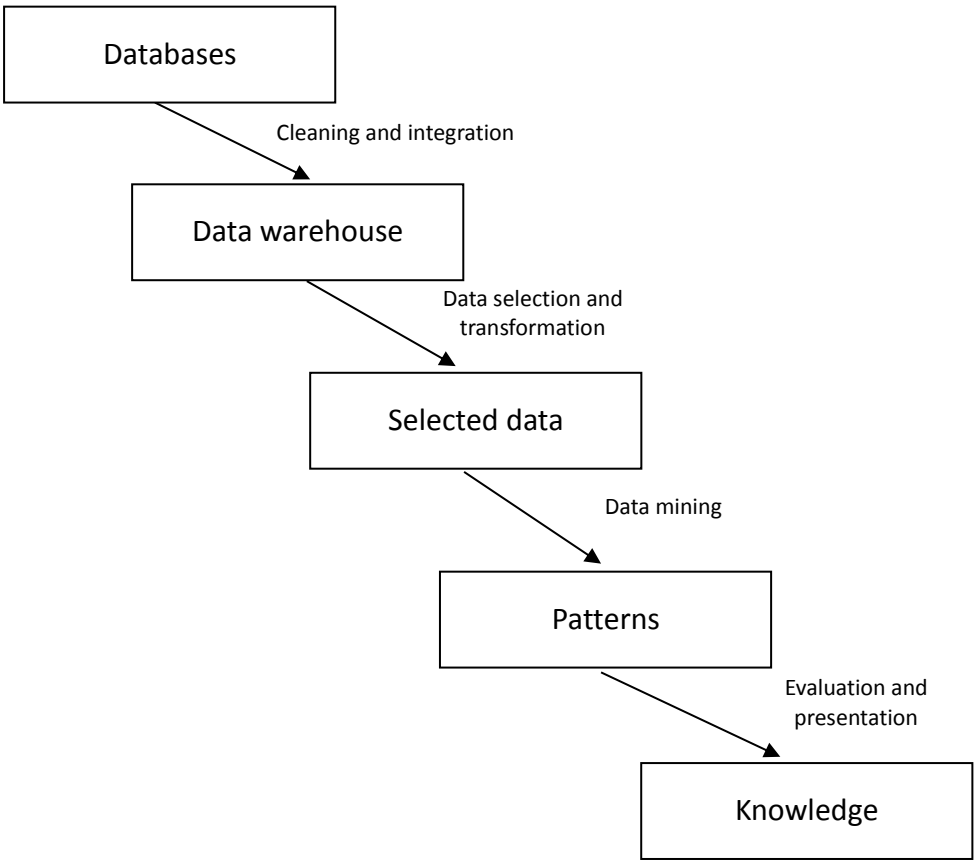

Figure 2.1: The KDD Process

KDD concerns the complex process of identifying valid, novel, potentially useful and ultimately understandable patterns in data [64]. Data mining refers to a particular step in the KDD process and is the automatic extraction of implicit and interesting patterns from large data collections [109]. Figure 2.1 presents the main steps of the KDD process, including data cleaning and data integration, data pre-

processing, data mining, and post-processing.

Real-world data tend to be incomplete, noisy and inconsistent [86]. The first step consists of two pre-mining tasks: data cleaning which is used to fill in missing values, remove noise and correct inconsistent data, and data integration which is utilized to bring data from multiple sources into a coherent data store, such as a data warehouse and/or data mart(s).

During the data pre-processing step, the data warehouse developed during the data cleaning and data integration phase is verified. Data are re-cleaned if needed. Data selection is then performed where data relevant to the analysis task are retrieved from the data store. Finally, data are transformed or consolidated into forms appropriate for mining.

The data mining step is essential in the KDD process. In this step intelligent methods are applied over data to extract interesting patterns. Some important data mining techniques include [86]:

- Classification and prediction, two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Clustering, a process of grouping a set of physical or abstract objects into classes of similar objects.
- Association rule discovery, a technique to find interesting associations among sets of items in transaction databases or other data repositories
- Anomaly detection, a process of identifying data objects that do not comply with the general behaviour or model of the data.

The final step of the KDD process is post-processing or post-mining. In this step, users are able to evaluate the patterns, that is, to determine the importance of the extracted patterns, using several user-driven methods or statistical database oriented methods. Visualization and knowledge representation techniques are also integrated in this step to present the mined knowledge to the user.

2.2 Association Rule Mining

Association rule mining, which was introduced by Agrawal et al. [3], is one of the most well-known techniques of data mining. Association rule mining searches for interesting relationships among items in a given dataset [86]. An example of such an association is that if a customer buys bread and butter then that customer is likely to also buy milk in the same transaction. Although initially motivated by the desire to analyse large retail transaction databases, association rule mining has been applied to a variety of industry sectors including commerce, defence, health, manufacturing, exploration and engineering.

2.2.1 Formal Definition of Association Rule Mining

Let us consider $I = \{i_1, i_2, \dots, i_m\}$ a set of m binary attributes, called *items*. An *itemset* is a non-empty subset of I. An itemset that contains k items is a k-itemset. Let $I = \{t_1, t_2, \dots, t_n\}$ be a set of n transactions, where each transaction t_i represents a binary vector, with $t_i[k] = 1$ if t_i contains the item i_k, and $t_i[k] = 0$ otherwise. A unique identifier is assigned to each transaction, called TID.

Definition 2.1 (Association Rule) An association rule is an implicit expression of the form

$$X => Y,$$

where $X, Y \subset$ I and $X \cap Y = \emptyset$. We call X the antecedent and Y the consequent of the rule.

Definition 2.2 (Support) Support of an association rule is defined as the percentage of transactions that contain $X \cup Y$ compared to the total number of transactions in the database. Support is calculated by the following formula:

$$Support(XY) = \frac{SupportCountOfXY}{TotalNumberOfTransactions}.$$

Definition 2.3 (Confidence) Confidence of an association rule is defined as the percentage of the number of transactions that contain $X \cup Y$ to the total number of records that contain X. If the percentage exceeds the threshold of confidence, an interesting association rule $X => Y$ can be generated.

$$Confidence(X|Y) = \frac{Support(XY)}{Support(X)}$$

Given a set of transactions D, the task of mining association rules is to generate all association rules that have support and confidence greater than the user-specified minimum support (called $minsup$) and minimum confidence (called $minconf$) respectively.

The task of association rule mining can be broken into two steps [3]: 1) find all frequent itemsets that hold transaction support above the minimum support threshold; 2) generate the desired rules from the frequent itemsets if they also satisfy the minimum confidence threshold.

2.2.2 Apriori: Classic Association Rule Mining

The Apriori algorithm was proposed by Agrawal and Srikant [6]. It is regarded as the classical association mining algorithm [47].

The Apriori algorithm provides an approach for frequent itemset generation, where the key idea lies in the *Apriori property* of the support, that is, if an itemset has minimum support, then all its subsets also have minimum support. Thus, any subset of a frequent itemset must also be frequent while any superset of an infrequent itemset must also be infrequent.

Algorithm 2.1: Apriori Itemset Generation [6]

```
1:Input: Database D
2:Output: The set L of itemsets
3: L_1= {1-itemsets}
4: for all (k = 2; L_{k-1} ≠ ∅; k++) do begin
5:      C_k = apriori-gen (L_{k-1})
6:      for all transactions t ∈ D do begin
7:           C_t = subset (C_k, t)
8:           for all candidates c ∈ C_t do
9:                c.count++
10:          end for
11:     end for
12:     L_k = {c ∈ C_k | c.count ≥ minsup}
13: end for
14: apriori-gen (L_{k-1})
15:      for all itemsets c ∈ C_k do begin
16:          for all (k − 1)-subsets s of c do begin
17:               if (s ∉ L_{k-1}) then
18:                    delete c from C_k
19:          end for
20:     end for
```

As shown in Algorithm 2.1, the process of frequent itemset generation works as follows. Let L_k be the frequent k-itemset and C_k be the candidate k-itemset. As shown in line 3, the frequent 1-itemsets are generated in the first pass over the data,

as denoted by L_1. Lines 4 to 13 show the process of *k*-itemsets generation. Starting from L_{k-1} which is generated in the previous step, the function *apriori-gen* (line 14) generates new C_k which are validated during a new pass over data when the support of each candidate is computed. During the process, the *Apriori property* is used as follows: if any (*k*-1)-subset of a candidate *k*-itemset is not in L_{k-1}, then the candidate cannot be frequent and can be removed from C_k. The algorithm ends when no further frequent itemsets are generated.

Example 2.1 Given a transaction database (D) as shown in Figure 2.2(a) and minimum support of 3, Apriori finds the complete set of frequent itemsets as follows:

- Scan D once to find frequent items, namely a, c, d, f, to form a L_1 (as shown in Figure 2.2 (b)). C_2 is generated from L_1 using the Apriori heuristic to prune the candidates: only those candidates that consist of frequent subsets can be potentially frequent.
- Scan D once more to count the support of each itemset in C_2. The itemsets in C_2 passing the support threshold form the L_2, as shown in Figure 2.2 (c).

Similarly, C_3 is generated from L_2 and D is scanned to identify the support count of each itemset in C_3. L_3 is then derived which consists of itemsets passing the support threshold (as shown in Figure 2.2 (d)). The process stops when no candidate can be derived or no candidate is frequent.

TID	Items
100	a, b, c
200	a, c, d, e, f
300	d, e,
400	a, b, c, f
500	a, c, d, f

(a) Transaction Database D

C_1		L_1	
Itemset	Support Count	Itemset	Support Count
{a}	4	{a}	4
{b}	2	{c}	4
{c}	4	{d}	3
{d}	3	{f}	3
{e}	2		
{f}	3		

(b) Result of C_1 & L_1

C_2		L_2	
Itemset	Support Count	Itemset	Support Count
{a, c}	4	{a, c}	4
{a, d}	2	{a, f}	3
{a, f}	3	{c, f}	3
{c, d}	2		
{c, f}	3		
{d, f}	2		

(c) Result of C_2 & L_2

C_3		L_3	
Itemset	Support Count	Itemset	Support Count
{a, c, f}	3	{a, c, f}	3

(d) Result of C_3 & L_3

Figure 2.2: Illustration of Apriori

Algorithm 2.2: Apriori Rule Generation [6]

1:**Input**: Set of itemsets l
2:**Output**: Set of association rules $Rules$
3: **for all** itemsets l_k, $k \geq 2$ **do**
4: **call** genrules(l_k, l_k);
5: **end for**
6: genrules(l_k: k-itemset, a_m: m-itemset)
7: $A = \{(m-1) - itemsets\ a_{m-1} | a_{m-1} \subset a_m\}$
8: **for all** $a_{m-1} \in A$ **do begin**
9: $confidence = \frac{support(l_k)}{support(a_{m-1})}$
10: **if** $(confidence \geq minconf)$ **then**
11: $R = a_{m-1} => (l_k - a_{m-1})$
12: **if** $(m-1 > 1)$ **then**
13: **call** genrules(l_k, a_{m-1})
14: $Rules = Rules \cup R$
15: **end if**
16: **end if**
17: **end for**
18: **return** $Rules$

After the generation of frequent itemsets, the second step of association rule mining is to derive rules from those itemsets which satisfy the minimum confidence

threshold. That is, for each frequent itemset l, generate a rule for every non-empty subset s of l [86]:

$$s => (l - s), if \frac{Support(l)}{Support(s)} \geq minconf.$$

The process is described in Algorithm 2.2.

Example 2.2 Let us consider the transaction database in Figure 2.2(a). Given minimum support count 3 and minimum confidence 0.60, there is a frequent itemset $i = \{a, c, f\}$ as shown in Figure 2.2(d). To get the rules from i, we first have the following possible rules: $\{a\} => \{c, f\}$, $\{c\} => \{a, f\}$, $\{f\} => \{a, c\}$, $\{a, c\} => \{f\}$, $\{a, f\} => \{c\}$, $\{c, f\} => \{a\}$. We then need to calculate the confidence of each possible candidate. For example, to compute the confidence of $\{a\} => \{c, f\}$, we use the support of the complete itemset $\{a, c, f\}$, and the support of the antecedent $\{a\}$.

$$Confidence(\{a\}|\{c, f\}) = \frac{Support(\{a, c, f\})}{Support(\{a\})} = \frac{3}{4} = 0.75$$

Since the confidence 0.75 is greater than the minimum confidence, it is deemed a valid rule.

2.3 Improvements in Frequent Itemset Generation

The entire performance of association rule mining is mainly determined by the step of frequent itemset generation [6]. Therefore, in the last few decades, how to improve the efficiency of frequent itemset generation has attracted a lot of attention from the data mining community.

2.3.1 Candidate Generation Algorithms

Candidate generation algorithms identify candidate itemsets before validating them with respect to incorporated constraints, where the generation of candidates is based upon previously identified valid itemsets [47]. The core algorithm of this genre is Apriori.

The Apriori algorithm has the effect of reducing the number of candidate itemsets and thus reducing computation, Input/Output (I/O) and memory costs [86]. However, it has two major drawbacks. One is that it requires multiple scans of the dataset residing in the disk and the other is that the candidate generation process is complex and resource consuming [236]. To overcome these issues, a number of important Apriori-based algorithms were designed with modifications focusing on two aspects: reducing the number of passes over the whole database and employing various pruning techniques to produce smaller candidate itemsets.

Apriori-TID [6], which was proposed by Agrawal and Srikant, only needs one scan of the database. A set C_k is constructed during the first pass of the database and each member of the set C_k is of the form $< TID, X_k >$, where each X_k is a potentially large k-itemset present in the transaction with identifier TID. The set C_k is used for counting support. Since the size of C_k is smaller than the database, this saves much reading effort [6].

Another approach, AprioriHybrid [6], combines the best features of Apriori and Apriori-TID, where Apriori is used in the initial passes and then Apriori-TID is utilized if it is expected that the set C_k at the end of the pass will fit in memory. Although there is the cost of switching, it has been shown empirically that

AprioriHybrid is significantly faster than both Apriori and Apriori-TID [6].

Further improvements of Apriori-Hybrid were proposed by Hipp, Güntzer and Nakhaeizadeh [95] who employed a hash-tree like structure to contain pointers to TID list sets instead of counters. A further revision developed to find a better approach to determining when to switch from Apriori to Apriori-TID was presented by Bodon [31]. Bodon's work proposed Apriori-Brave which keeps track of memory need and stores the amount of the maximal memory need. After the generation of (k+1)-itemsets, the (k+2)-itemset candidates are generated only when the memory need does not exceed the maximal memory need [31].

Savasere et al. [189] proposed a partition algorithm which requires only two database scans. The algorithm consists of two phases. In the first step, the algorithm divides the database into small non-overlapping partitions that can be processed independently and efficiently in memory to find their frequent itemsets. In the second step, only one scan of the database is required to find the frequent itemsets from the candidates. The partition sizes and the number of partitions are chosen to ensure that each partition can be accommodated in the main memory and the partitions are read only once in each phase.

Brin et al. [39] proposed the DIC (Dynamic Itemset Counting) algorithm. DIC utilizes a dynamic itemset counting technique in which the database is partitioned into blocks marked by start points. DIC reduces the number of passes over the database by introducing an original idea, namely that (k + 1) candidates are computed from the k pass. When a k-itemset is considered frequent, all the (k+1)-itemset candidates that the latter can produce are generated.

In addition to the efforts to reduce the number of database scans, another approach to improving the efficiency of Apriori is the utilization of different pruning techniques to generate a smaller number of candidate itemsets. DHP (Direct Hashing and Pruning) [164] uses a hashing technique to filter out unnecessary itemsets for the generation of the next set of candidate itemsets. Instead of including all the k-itemsets from $L_{k-1} \times L_{k-1}$ into C_k in the Apriori algorithm, a k-itemset is added into C_k only if that k-itemset passes the hash filtering, that is, k-itemset is hashed into a hash entry if its value is larger than or equal to the minimum support [164]. Such hash filtering can drastically reduce the size of C_k. Further improvements of DHP were proposed in PHP (Perfect Hashing and Pruning) [158]. In this approach, a hash table with size equal to the distinct items in the database is created during the first pass where each distinct item in the database is mapped to different location. The prune method of the hash table prunes all the entries whose support is less than the minimum support.

2.3.2 Pattern Growth Algorithms

Pattern growth techniques eliminate the need for candidate generation by constructing complex hyper-structures that contain representations of the itemsets within the dataset [47]. Generally, a hyper-structure is composed of two principal structures [47]:

- Pattern frame - represents a tree-based or array-based structure containing items with their support which is constructed in a database pass by using each transaction.
- Item list - contains the list of frequent items. Each item is linked to the

first element in the pattern frame that contains it.

2.3.2.1 FP-Growth Algorithm

The fundamental pattern growth algorithm, FP-growth, was proposed by Han et al. [88]. In their work, a frequent pattern tree (FP-tree) is used for storing compressed, crucial information about frequent patterns. An FP-tree consists of one root (labelled "null"), a set of prefix subtrees as the children of the root and a header table. Each node in the sub-tree has three fields: item-name, support count and node-link. Each entry in the header table has two fields: item-name and the head of the node-link

There are two steps to construct an FP-tree. At the first step, an initial scan of the database is conducted to identify the frequent 1-itemsets and an ordered list of frequent items is generated. The ordered list is sorted by their frequency and is stored in the header table. At the second step, an FP-tree is constructed as follows.

Firstly, a second complete scan of the dataset is performed. For each transaction read, only the set of frequent items present in the header table is collected and items are sorted in descending order according to their frequency. These sorted transaction items are inserted into the FP-tree as follows: for the first item on the sorted transactional dataset, check if it exists as one of the children of the root. If it exists then increase the support count for this node by 1. Otherwise, add a new node for this item as a child of the root node with 1 as support count. Then, consider the current item node as the new temporary root and repeat the same procedure with the next item on the sorted transaction. To facilitate tree traversal, during the process of adding any new item-node to the FP-tree, a link is maintained

between this item-node in the tree and its entry in the header table.

TID	Items
T1	a, b, c, d
T2	a, b, d, e
T3	a, c, d
T4	a, b, c
T5	a, b, c
T6	a, c, d
T7	a, b, e
T8	a, b, d

(a) Transaction Database

Item	Support Count
a	8
b	6
c	5
d	5
e	2

(b) All Items

Item	Support Count
a	8
b	6
c	5
d	5

(c) Frequent 1-itemset

Figure 2.3: Sample Data and Frequent 1-itemset

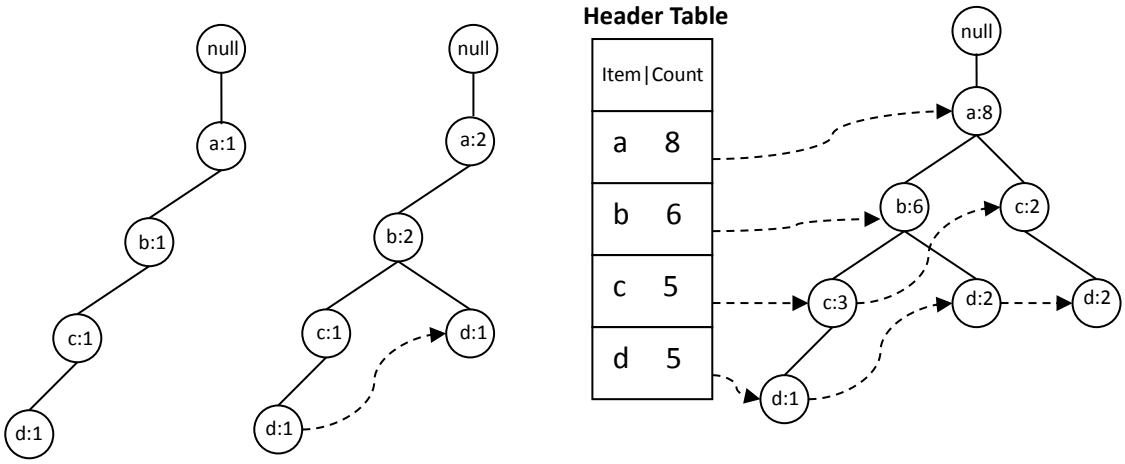

Figure 2.4: FP-Tree Construction

Example 2.3 For illustration, let us take an example with transactions shown in Figure 2.3(a). Figure 2.3(b) shows all items with their support count. Given the minimum support threshold is 4, the non-frequent item is removed, which is e. Finally, all frequent items are sorted according to their support count to generate the sorted frequent 1-itemset, as shown in Figure 2.3(c).

During the tree construction process, frequent items in the first transaction (a, b, c, d) are sorted according to their support count and then inserted into the root, as

shown in Figure 2.4(a). When inserting the second transaction, the sorted frequent item list (a, b, d) shares the same prefix (a, b) with an existing path on the tree. The support counts of item-nodes (a and b) are increased by 1 and a new sub-path is created with the remaining items on the list (d) all with support equal to 1 (as shown in Figure 2.4(b)). During the process, a link is established between the two nodes with item-name d. The same procedure occurs until all transactions shown in Figure 2.3(a) have been inserted. Figure 2.4(c) shows the resultant FP-tree.

FP-growth employs a *divide-and-conquer* technique for frequent itemset generation which is based on one important concept: conditional pattern base [88, 89]. Given a frequent itemset, a conditional pattern base consists of a set of prefix paths in the FP-tree co-occurring with that itemset. A conditional FP-tree is constructed based on the conditional base. Starting from each frequent *length*-1 pattern (as an initial suffix pattern), the pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-tree [88, 89].

Example 2.4 To illustrate, let us take the FP-tree in Figure 2.4(c) as an example. Let the minimum support threshold be 2. The process to generate frequent itemsets is as follows.

FP-growth starts with item d which is the last item in the header table. A set of branches is obtained through the node link from the FP-tree. The paths in those branches are (a, b, c, d: 1), (a, b, d: 2) and (a, c, d: 2) (the number after ":" represents the support count of the nodes in a FP-tree branch). Considering d as a suffix, the corresponding three prefix paths are (a, b, c: 1), (a, b: 2) and (a, c: 2), which form the conditional pattern base. The conditional FP-tree is constructed as

shown in Figure 2.5(a), which generates a set of itemsets: {a, d}(5), {b, d}(3), {c, d}(3), {a, b, d}(3), {b, c, d:1},{a, c, d}(3) and {a, b, c, d}(1) (the number in "()" represents the support count of an itemset). Since {a, b, c, d} and {b, c, d} has a support count less than the minimum support, they are pruned out.

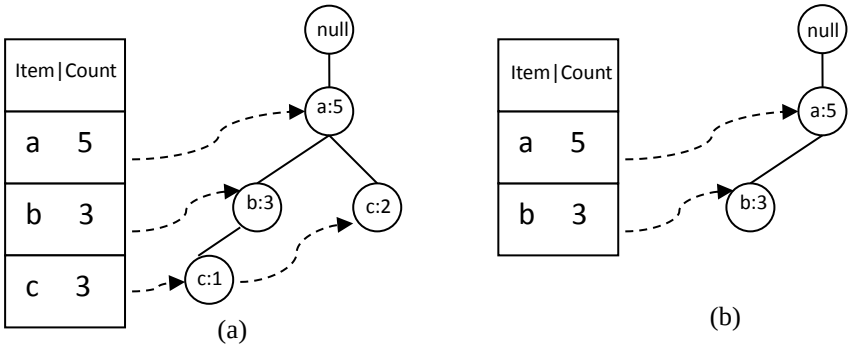

Figure 2.5: Sample Conditional FP-Tree

FP-growth then moves to the next item, c, in the header table. The item has two prefix paths for the conditional pattern base, namely, (a, b, c:3) and (a, c:2), which generates a single branch conditional FP-tree (a, b), as shown in Figure 2.5(b). The sets of frequent itemsets are: {a, c}(5),{b, c}(3) and {a, b, c}(3), which are all frequent as their support counts are greater than the minimum support. Similarly, FP-growth processes the rest of the items in the header table to generate all frequent itemsets.

The FP-tree is usually smaller than the original database and, thus, saves costly database scans in the mining process. Furthermore, FP-growth uses a *divide-and-conquer* technique that considerably reduces the size of the subsequent conditional FP-tree. However, the FP-growth algorithm still suffers some drawbacks. First, it is difficult to use in an interactive mining system [236]. During the interactive mining process, users may change the threshold of support in response to the rules

produced, potentially leading to repetition of the whole mining process. Second, the FP-tree algorithm is not suitable for incremental rule mining [236]. In addition, the size of the tree usually increases exponentially as the number of unique items increases [47].

2.3.2.2 FP-Growth Based Algorithms

A number of FP-tree/FP-growth based algorithms have been developed and brought improvements to the FP-growth algorithm. One such effort is to employ array-based structures to facilitate the searching process. For example, FP-Growth* [78] uses an extra array-based structure to decrease the number of traversals of the tree, which saves time during general traversal of the tree and also enables direct initialization of the next level of the FP-tree.

H-Mine [168] also uses an array-based structure which is constructed in a manner similar to FP-growth. In H-Mine, global frequent items are identified during the first scan and then a hyper-linked data structure (called H-struct) is created from those items in the second scan. The advantage of H-Mine over FP-growth is that it uses the same pattern frame structure with semantics changed through pointer manipulation, rather than the creation of conditional FP-trees as required in FP-growth [47].

ITL-Mine [76] is an optimization of H-Mine. It needs one scan of the database/dataset which creates the underlying structures that are similar to H-struct, except that the header tables maintain all items. These extra links avoid the progressive recalculation of linkage during processing which occurs in H-Mine. A further improvement of ITL-Mine is CT-ITL [201], which uses a compressed

pattern frame structure to reduce the storage space/memory required and also lessen traversal overheads. Although it requires two database/dataset scans, CT-ITL is considered to be more scalable than ITL-Mine, especially as the size of the database increases [202].

Liu et al. [137] proposed a hybrid pattern growth algorithm known as Opportunistic Projection, which opportunistically chooses between array-based or tree-based structures to represent projected transaction subsets. The algorithm heuristically decides to build an unfiltered pseudo-projection or to make a filtered copy according to features of the subsets to achieve the maximized efficiency and scalability. Later, PatriciaMine [171] proposed a compressed trie (Patricia trie) which alleviates the need to swap between trie and array-based data structures based upon dataset density as proposed in H-Mine and Opportunistic Projection.

Wang et al. [216] proposed TD-FP-Growth, a top-down variation to the FP-growth approach. This approach is said to alleviate the need or demand to generate/build conditional pattern bases and physical projections of the trie. Similarly, COFI [59] was proposed to provide an efficient pattern growth algorithm that uses a top-down non-recursive technique.

Lin et al. [131] proposed IFP-growth, which employs an address-table structure to lower the complexity of forming the entire FP-tree and a new structure called FP-tree+ to reduce the need to build conditional FP-trees recursively. By using an address-table and FP-tree+, the proposed algorithm has less memory requirement and better performance in comparison with FP-tree based algorithms.

2.4 Improvements in Rule Generation

One main issue related to association rule mining is that classical techniques produce a high number of rules which are often unusable by the user [86]. Consequently, intensive research has been conducted to reduce the number of extracted rules or improve their quality, for example:

- Interestingness measures. They are either objective or subjective. Objective interestingness measures, such as support and confidence, as well as others measures such as lift/interest, Laplace correction and chi-square statistics, are used to rank the obtained rules to allow users to select rules in which they are more interested [74, 204]. Subjective measures are based on subjective factors controlled by the user. Most of the subjective approaches, such as unexpectedness and actionability involve user participation in order to express which rules are of interest [132].
- Pattern visualization. Visualization techniques are utilized to present mining results in order to exploit the natural human pattern recognition capability [154]. Commercial KDD products and many prototypes have incorporated methods for the visualization of results [183].
- Condensed representations. Algorithms, such as disjunction-free sets [44], deduction rules [45], counting inference [27] and closed itemset algorithms [215, 169], have been developed to produce a reduced result set from which valid patterns can be derived.
- Integration of domain knowledge. Domain knowledge consists of information that is not explicitly presented in the database rather it is made

available from a domain expert. When a set of rules is generated from the dataset, pre-conceived knowledge about the domain can help the user to determine how well these rules match or contradict the user's existing knowledge. Less interesting rules can be ignored, thus reducing the number of rules in focus [77, 229, 220, 114].

Among those available techniques, here the focus is on constraint-based association rule mining techniques and the reduction of redundant rules.

2.4.1 Constraints-Based Association Rule Mining

Constraint-based association rule mining allows the user to impose a set of constraints over the content of the discovered rules, and therefore only generate those association rules that are interesting to them individually. These constraints can be knowledge type constraints, data constraints, dimension/level constraints, interestingness constraints and rule constraints [86].

Rule constraints define the form of rules to be mined and can be specified using a high level declarative data mining language. For example, Shen et al. [192] proposed meta query, a technique to specify the form of rules to be discovered in data mining. Furthermore, several data mining query languages, such as DMQL [85] and MSQL [99, 100] have been proposed for this purpose.

There have been several approaches to applying rule constraints to the mining process. One of these is meta-rule guided mining [67]. A meta-rule is one kind of constraint that is based on the user's experience, expectations or intuition regarding the data, or can be generated from the data schema. A meta rule is a rule template of the form

$$P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$$

where $P_i (i = 1, \ldots, l)$ and $Q_j (j = 1, \ldots, r)$ are either instantiated predicates or predicate variables.

Meta-rules allow users to specify the syntactic form of the rules they expect. For an example, a meta-rule can be $X, Y \Rightarrow Z$, where X, Y and Z represent any items in the database. According to this meta-rule, only frequent 3-itemsets can produce this kind of rule, which in turn makes the algorithm more efficient as early pruning can be done.

Another approach was proposed by Bayardo et al. [29] where consequent constraint is applied to the consequent of all the rules to a certain itemset and minimum improvement constraint ($minimp$) is used to prune uninteresting rules. In their approach, a proper sub-rule is defined as a simplification of the rule formed by removing one or more conditions from its antecedent. The $minimp$ prunes any rule that does not offer a significant predictive advantage over its proper sub-rules. This increases efficiency and presents the user with a concise set of predictive rules that are easy to comprehend [29].

Furthermore, Ng et al. [151] and Srikant et al. [200] proposed algorithms incorporating item constraints on the process of generating frequent itemsets, from which association rules are derived. The item constraints restrict the items and combination of items that are interesting to the user.

2.4.2 Removing Redundant Rules

Association rule mining may produce a large number of redundant rules, the definition of which varies. For example, Jaroszewicz and Simovici [101] regarded

a rule as redundant if its true confidence is close to the estimate while Bastide et al. [26] defined redundant association rules based on a decision rule that compares the confidence or support of an association rule to similar rules. For instance, rule $X => Y$ is a "minimal non-redundant association rule" if there is no rule $X' => Y'$ with $X' \subseteq$ X, $Y \subseteq$ Y′ such that $supp(XY) == supp(X'Y')$ and $conf(X => Y) = = conf(X' => Y')$. Another definition was given by Zaki [232] based on the concept of frequent closed itemsets. A set is called closed if it has no proper superset with the same support. Given a non-closed set X, any set Y in its closure, and a rule $X => Z$, rules of the form $XY => Z$ and $X => YZ$ are treated as redundant if their frequencies and confidences are identical with the rule $X => Z$.

Redundant rules contain information or knowledge that is less interesting to the user. It becomes a crucial problem when the data is dense or correlated (such as in statistical datasets) [166]. Eliminating redundant rules has received a great deal of attention from various research communities [232, 233, 166, 223, 224, 75].

Zaki [232, 233] proposed an algorithm to remove redundant rules based on the concept of *minimal generators*. A generator X' of an itemset X is a subset of $X(X' \subset X)$ that has the same support as $X(supp(X) = supp(X'))$. A generator of X which has no subset generator of X is called *minimal generator* of X. Then, each *minimal generator* can be the left part or, respectively the right part of a non-redundant association rule.

Further to the work by Zaki, Pasquier et al. [166] introduced two condensed association bases to represent non-redundant association rules: Min-max Approximate Basis and Min-max Exact Basis. Both Pasquier et al.'s and Zaki's

approaches are based on frequent closed itemsets with Zaki using the mathematic framework of Formal Concept Analysis (FCA), while Pasquier et al.'s approach was based on Galois closure.

Xu and Li [223] improved the definitions suggested by Pasquier et al., proposing a condensed representation called Reliable Exact Basis for exact association rules. The rules in the Reliable Exact Basis are not only non-redundant but also more succinct than the rules in Min-max Exact Basis. However, their work is focused on reducing the redundancy in rules that have a confidence value of one, which only offers a small reduction in redundancy for rules which have a confidence of less than one.

Later, Xu, Li and Shaw [224] extended this work by introducing the concept of approximate association rules, which are rules with confidence less than one. In their work, they present a concise representation basis called Reliable Approximate Basis to extract non-redundant approximate rules. They claimed that no information or knowledge is lost and all approximate association rules can be deduced from the Reliable Approximate Basis [224].

Another approach proposed to deal with redundant rules is based on the extraction of non-derivable association rules [75]. The idea behind this approach is that if the lower and upper bounds of a rule coincide and the confidence is uniquely determined by the subrules, the rule can be pruned as redundant, or derivable, without any loss of information. In this approach, proposed redundancy is tested by deriving the absolute bounds of the rule's confidence instead of estimating them as is done in other approaches [75].

2.5 Extensions of Association Rule Mining

With the advances of association rule mining, a variety of extensions have been proposed, such as the mining of spatio-temporal association rules [7, 127, 175, 87, 115], multi-level association rules [54, 66, 84, 190], negative association rules [188, 221, 38] and fuzzy association rules [119]. This section reviews some of the extensions of association rule mining, which are the main interests for this thesis, including quantitative association rule mining, multi-level association rule mining and temporal association rule mining.

2.5.1 Quantitative Association Rule Mining

The initial work on mining association rules introduced by Agrawal et al. [6, 3] targeted databases consisting of categorical attributes only, that is, attributes containing discrete and typically unordered data (e.g., gender, brand). Later, Srikant and Agrawal [199] extended the categorical definition of association rules (which are termed Boolean association rules (BARs)) and introduced quantitative association rules (QARs) which involve either quantitative attributes or categorical attributes. An example of a quantitative association rule would be:

$$\{age = [25,35], gender[female]\} => \{salary = [\$50{,}000, \$85{,}000]\}$$
$$(\sigma = 0.03, \gamma = 0.8)$$

QARs are more expressive and informative than BARs due to their ability to represent a wide variety of real-life attributes. In the last decade, a lot of research has been proposed aiming to improve the efficiency for mining such rules.

One common approach to mining QARs is to transform the task into conventional BAR mining where values of quantitative or categorical attributes are mapped to

Boolean attributes and then algorithms are applied to mine QARs. In this approach, a discretization process is often utilized to partition the values of attributes into intervals and then combine adjacent intervals as necessary [124, 217, 147, 199].

Another QAR mining approach is based on statistical methods, in which the consequent of a rule is a statistical measure (e.g., mean, variance) or an aggregate (e.g., min, max) of a quantitative attribute. This type of rule is derived mainly to provide a statistical view of the attributes, rather than giving the interval information of the attributes. For instance, Aumann and Lindell [14] considered the distribution of continuous data via standard statistical measures and provided a new definition of quantitative association rules where the right-hand side (RHS) of a rule expresses the distribution based measures of interestingness, such as the mean or variance of the values of numeric attributes. One example of such a rule would be:

$$Sex = female \; and \; Bachelor \; Degree = yes => mean \; salary = \$50{,}000,$$

which states that the average salary for females with bachelor degrees is $50,000.

Webb [219] further extended this framework to include other statistical measures such as minimum (min), maximum (max) and count. Furthermore, Zhang et al. [234] introduced the concept of statistical quantitative rules (SQ rules) in which the RHS can be any quantitative statistic that can be computed for the subset of data satisfying the left-hand side (LHS) of a rule.

The third approach of QAR mining is optimization-based where numeric attributes are optimized during the mining process. The term optimization was first used by Fukuda et al. [69], who proposed an optimization criterion called gain. Gain takes

into account both the support and confidence of a rule and is defined as follows:

$$Gain(A => B) = Supp(AB) - minconf \times Supp(A)$$

The authors also defined an association rule R that has the form $A_i \in [l_i, u_i]$ ^ $C_p =>$ C_q, where A_i is the i^{th} numeric attribute in the rule template from the left to the right, and $[l_i, u_i]$ represents the whole domain of the i^{th} numeric attribute. C_p and C_q contain only instantiated conditions. They proposed schemas to determine values for variables l_i and u_i such that the confidence, support or gain of the rules is maximized.

Later, the authors extended their work to handle rules containing two un-instantiated numeric attributes on the LHS of a rule [68]. Furthermore, they addressed the optimized support problem and optimized gain problem to allow association rules to contain up to k disjunctions over uninstantiated numeric attributes [180, 40].

In addition, Mata et al. [143] presented a tool to discover association rules in numeric databases without the necessity of discretizing a priori, the domain of the attributes while Ruckert et al. [184] proposed approaches to represent quantitative association rules based on half spaces. Recently, QuantMiner [186] was introduced which can dynamically discover "good" intervals in association rules by optimizing both the support and the confidence.

2.5.2 Multi-level Association Rule Mining

Traditionally, association rule mining has been performed at the level of a single concept or abstract [84, 83, 82]. However, in practice, transaction databases may contain data with a hierarchical structure. One example is shown in Figure 2.6.

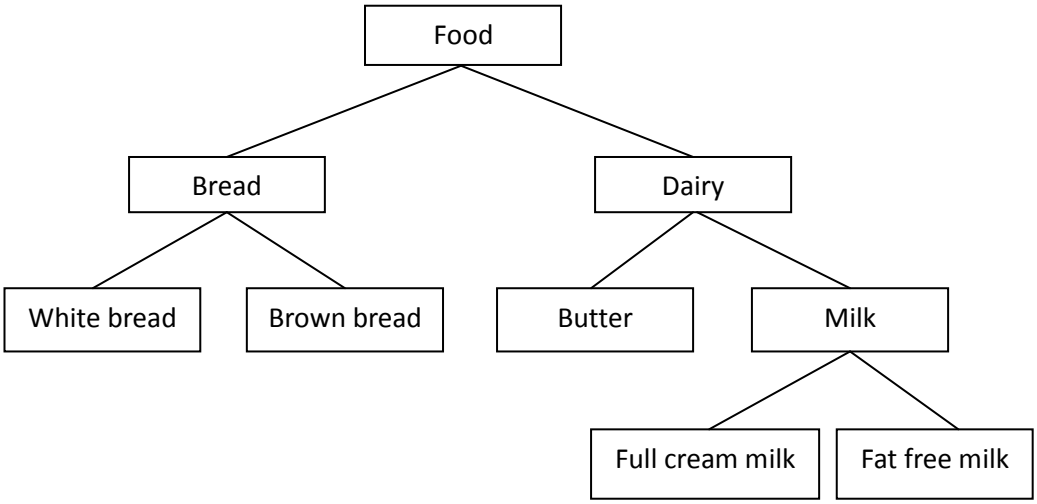

Figure 2.6: Example Food Hierarchical Structure

Given such a food hierarchical structure, it might be difficult to find strong associations among data items at low or primitive levels of abstraction. For instance, a strong association between items “fat free milk” and “bread” may not be found as they may occur in a very small fraction of the transactions. However, consider the generalization of “fat free milk” to “milk”. It might be easier to find a strong association rule like: {milk} => {bread}. Such rules, which reveal relationships between items or attributes at different levels of conceptual hierarchy, are termed multiple-level or multi-level association rules [86]. As shown in the above example, multi-level association rule mining has the potential to discover knowledge which may be ignored by the single-level approach.

There are two main issues regarding multi-level association rule mining. First, it is often difficult to choose the proper support threshold(s) to determine frequent itemsets for each level. One straightforward approach is using a uniform minimum support for all levels. However, since items at lower levels of abstraction are unlikely to occur as frequently as those at higher levels of abstraction, setting the

support too low or too high may result in interesting associations being missed at either high or low abstraction levels [86]. Second, the rules discovered through multi-level association rule mining are dependent on the taxonomy that is built or used. However, it is not trivial to choose or build a suitable ontology/taxonomy for the database [84, 82].

In the literature, there have been a lot of works proposed aiming to improve the efficiency of multi-level association rule mining, including Apriori based approaches [198, 83], FP-growth based approaches [141, 155] and others, such as those techniques based on statistics [160] and fuzzy set theory [97, 96, 107].

Sirkant and Agrawal [198] introduced a simple algorithm called “Basic” to mine multi-level association rules. “Basic” utilizes the Apriori approach by adding the ancestors of each item in a transaction into that transaction and then performing the mining process across the expanded transactions. “Basic” is a slow algorithm and the authors proposed two further algorithms to overcome this deficiency: Cumulate and EstMerge. The Cumulate algorithm only adds ancestors that are in one (or more) of the candidate itemsets being counted in the current pass and the EstMerge algorithm utilizes sampling techniques to estimate the support of the candidate to determine whether it is necessary to calculate their actual support [198].

Srikant and Agrawal’s approach [198] uses the support threshold across all the levels, which may generate uninteresting rules if the threshold is set low or exclude interesting rules at low levels if the threshold is set high [83]. To overcome this drawback, Han and Fu [83] employed a different strategy which applies a variable support count at different levels. In their work, they extended the Apriori algorithm

to generate strong multi-level association rules which are defined as follows [83]:

Definition 2.4 (Strong Multi-level Association Rule) A pattern is frequent in set S at level l if its support is no less than the $minsup$ for the corresponding level. Then a rule "$A => B/S$" is strong for a set S if it satisfies the following conditions:

- all ancestors of A and B are frequent in their corresponding level.
- the support of $A \wedge B$ is frequent in the current level.
- the rule confidence is no less than the threshold in the current level.

The Apriori approaches for multi-level association rule mining suffer the same bottlenecks as the original Apriori [6, 3], as discussed in Section 2.3.1. To overcome the drawbacks, Mao [141] proposed an FP-growth based approach, called Adaptive FP-growth (Ada-FP), which pushes various support constraints into the mining process and is able to discover both inter-level frequent patterns and intra-level frequent patterns. Furthermore, Ong et al. [155] proposed FP'-Tree, which extends FP-growth to mine multi-level association rules with recurrent items. Unlike Ada-FP, this approach builds a separate FP-tree for each concept level that is being mined.

2.5.3 Temporal Association Rule Mining

With the mounting recognition of the value of temporal data, data modelling and databases, an important research area has been how to mine temporal association rules from both static and longitudinal/temporal data. A temporal association rule (TAR) can be represented as a pair < AR, TE >, where AR is an association rule and TE is a time expression belonging to AR [51].

Temporal association rule mining has the ability to mine the behavioural aspects of

(communities of) objects as opposed to simply mining rules that describe their states at a point in time [128]. One example TAR is < "short of breath ~ asthma", Days.Hours (6:10) >, which indicates that patients with symptoms of shortness of breath will have symptoms of asthma from 6 to 10 o'clock every morning.

Several kinds of TARs have been proposed in the literature, some of which are discussed in more detail in the following section.

2.5.3.1 Interval-Based Temporal Association Rule

In large databases, products may not necessarily exist throughout the whole time when a database is gathered. Ale and Rossi [7] introduced the concept of lifetime of members of an itemset. Each rule has an associated time frame, corresponding to the lifetime of the items participating in the rule. Furthermore, Lee, Lin and Chen [120] employed the concept of maximal common exhibition period (MCP) in their Progressive-Partition-Miner (PPM) algorithm to discover general temporal association rules in a publication database. Their algorithm first partitions the publication database in light of exhibition periods of items and then progressively accumulates the occurrence count of each candidate 2-itemset based on its intrinsic partitioning characteristics. In addition, Rainsford and Roddick [175] proposed a method to add temporal features to association rules by associating a conjunction of binary temporal predicates that specify the relationships between the timestamps of transactions.

2.5.3.2 Cyclic Temporal Association Rule

Association rules may also display regular hourly, daily, weekly, etc., variations that have the appearance of cycles. For example, Hanau [90] discovered that every

18-24 months a big supply of pork for low prices was followed by a low supply for high prices in the pig market in Europe in 1927.

Ozden et al. [157] defined a cyclic association rule as a rule which has the minimum support and confidence at regular time intervals. Based on this definition, a cyclic rule does not hold for the entire transactional database, but only for transactional data in a particular periodic time interval. The authors proposed two algorithms to mine cyclic rules. One is a sequential algorithm which uses existing algorithms to discover association rules, utilizing several techniques to reduce the running time, including cycle-pruning, cycle-skipping and cycle-elimination [157]. Another is an interleaved algorithm which consists of two steps [157]. In the first step, the search space for large itemsets is reduced using cycle-pruning, cycle-skipping and cycle-elimination and then in the second phase, the cyclic association rules are calculated using the cycles and the support of the itemsets without scanning the database.

2.5.3.3 Calendar-Based Temporal Association Rule

Ramaswamy et al. [177] considered the discovery of association rules that hold during the time intervals described by a calendar algebraic expression. Later, Li et al. [127] proposed a calendar schema to define a set of simple calendar-based patterns. For example, given a calendar schema (year, month, day), a calendar-based pattern within the schema might be (*, 6, 30), which represents the set of time intervals each corresponding to the 30th day of a June. The notation * is a wildcard denoting every arbitrary integer in the domain of the accordant attribute, in this case, a year.

Based on the definition of calendar schema, a calendric association rule is defined as follows.

Definition 2.5 (Calendric Association Rule) A calendric association rule over calendar schema R is a pair (r, e), where r is an association rule and e is a calendar pattern on R.

Li et al. introduced two classes of calendric association rules [127]: *Temporal association rules w.r.t. full match* which requires the rules to hold during every interval in e, and *temporal association rules w.r.t. relaxed match* which requires the rules to hold during a significant fraction of these intervals. In their work, they extended the Apriori algorithm and developed two optimization techniques to discover both classes of temporal association rules.

2.6 Mining over Association Rules

2.6.1 Overview of Higher Order Mining

The data used for data mining are typically assumed to be primary or raw data captured by some application, cleaned and prepared according to the demands of the mining algorithms [183]. For example, barcode scanning technologies are widely employed to collect transactional data for market data analysis, while with the advances of cloud technology, web browsing history data are stored and utilized for data mining tasks, such as fault detection. However, primary data might not always be available for data mining routines for the following reasons:

- Data ownership. Cooperating institutions that are interested in sharing knowledge may not be willing to disclose their primary data. Thus in some

cases the rules are all that the researchers have on which to operate [183]. For example, Prodromidis et al. [173] pointed out that in fraud detection in financial information systems, financial institutions are not willing to disclose their own proprietary data due to competitive and confidentiality considerations.

- Temporary or transient data. In some applications, primary data are only available for a short time, such as stream data which are not stored. They are encountered, processed in real time and deleted [70].
- Legal or confidentiality restrictions. There might be legal obligations or confidentiality considerations applied to primary data, such as regulations determining when the data records can be disclosed.

Further complexity is added by many data mining routines becoming heavily I/O bound due to the fact that the volume of data requiring analysis is growing disproportionately to the comparatively slower improvements in I/O channel speeds which limit many of the benefits of the technology [128]. Methods of reducing the amount of data have been discussed in the literature and include statistical methods, such as sampling or stratification, reducing the dimensionality of the data by, for instance, ignoring selected attributes, or by developing incremental maintenance methods by analysing the changes to data only [55, 56]. However, these add to the processing complexity and cost thus achieving little beyond transferring the problem to a later stage.

Based on the above observations and motivated by the increasingly frequent need to define and practice data mining without the luxury of primary data, Roddick et

al. [183] introduced the paradigm of higher order mining (HOM) which is a form of data mining that is applied over non-primary, derived data or patterns.

Definition 2.6 (Higher Order Mining) Let P_i be a set of patterns or models derived from a dataset D_i. Given $P = \{P_1, P_2, \dots, P_n\}$, where $n > 1$, higher order mining discovers any new pattern or model P' from P through the use of data mining methods.

As shown in the definition, HOM is the sub-field of knowledge discovery concerned with mining over patterns/models derived from one or more large and/or complex datasets. Since HOM discovers patterns from non-primary data, it thus avoids several problems that traditional data mining techniques encounter and has the following benefits [128]:

- the ability to combine mining strategies through the modular combination of components
- the provision for the development of higher order explanations in describing facts about data, particularly those describing changes over time, location or some other dimension
- the comparatively faster execution time due to reduced volumes of data.

Higher order mining opens a window for changes in perspective for knowledge discovery, from the analysis of data to the analysis of patterns [183]. There have been many advances in this paradigm, including in the areas of pattern clustering [80, 124, 34], pattern classification [11, 65, 72, 134], trend detecting [13, 48, 2, 91, 106, 145, 195, 196, 63], pattern change detection [73, 71, 72, 25] and pattern maintenance [54, 55]. The following sections provide general descriptions of

related works on HOM over association rules.

2.6.2 Clustering Association Rules

Association rules, such as $\{milk\}$ ^ $\{butter\} => \{bread\}$, are derived from transactional databases. For non-transactional data, a record might be in a form like (attribute = value) where the attribute is defined in the database schema and can be either categorical or non-categorical. For example, we might have a rule like

$$\{age = 40\} \text{ ^ } \{salary = \$50{,}000\} => \{own_{home} = yes\}$$

However, when mining association rules from this type of non-transactional data, we may find hundreds or thousands of rules corresponding to specific attribute values. For example, the following three rules

$$\{age = 40\} \text{ ^ } \{salary = \$50{,}000\} => \{own_{home} = yes\}$$

$$\{age = 45\} \text{ ^ } \{salary = \$55{,}000\} => \{own_{home} = yes\}$$

$$\{age = 50\} \text{ ^ } \{salary = \$60{,}000\} => \{own_{home} = yes\}$$

might be better described as

$$\{age = [40 - 50]\} \text{ ^ } \{salary = [\$50{,}000 - \$60{,}000]\} => \{own_{home} = yes\}$$

To handle this case, Lent et al. [124] introduced a clustered association rule as a rule that is formed by combining similar, adjacent association rules to form a few general rules where the set of (attribute = value) equalities are replaced by the set of value ranges using inequalities.

Definition 2.7 (Clustered Association Rule) A clustered association rule is an expression of the form

$$X_c => Y_c$$

where X_c and Y_c are items of the form (attribute = value) or ($bin^i \leq attribute <$

bin^{i+1}) where a bin is the interval between attribute partitions and bin^i denotes the lower bound for values in the i^{th} bin.

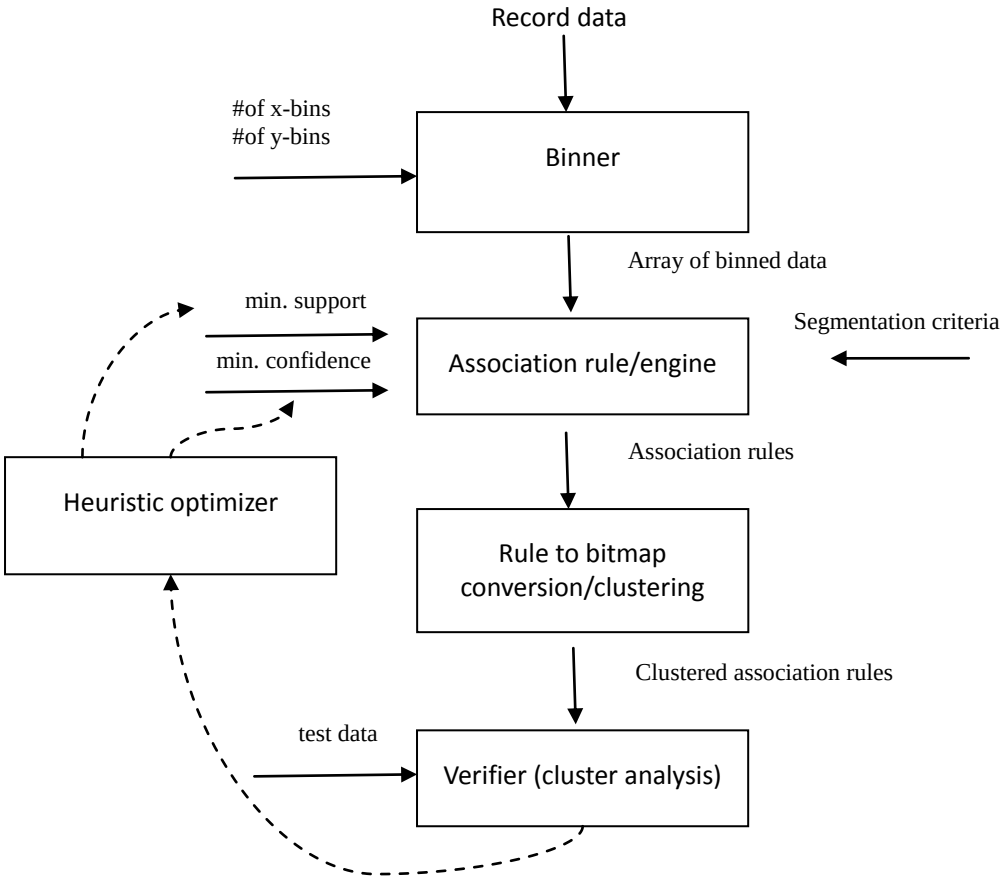

Figure 2.7: Architecture of the Association Rule Clustering System

Figure 2.7 shows the process of clustering association rules [124]. In this approach, source data are taken in tuple form and the values of attributes from a continuous domain are partitioned. Then a set of association rules is generated through a single pass over the data using an association rule engine. Finally, all those two-attribute association rules are clustered where the RHS of the rules satisfies its segmentation criteria [124]. Lent et al.'s approach to the clustering problem is heuristic and produces an efficient linear time approximation to an optimal solution. However,

one drawback of this application is that it is limited to rules with two attributes in the set of antecedents.

Toivonen et al. [210] proposed another approach for clustering association rules, which defines the distance between two association rules as the number of rows where the rules differ. The distance measure is then used to group all the rules into appropriate clusters. In their work, the set of rules can be pruned by forming rule covers, which are subsets of the original set of rules such that for each row in the relation there is an applicable rule in the cover if and only if there is an applicable rule in the original set. One of the limitations of this approach is that the distance measures selected for rule clustering are somewhat arbitrary [211]. Moreover, it is not clear how to describe the rule cluster concisely to the end-user since rules belonging to the same cluster may have substantially different structures.

To improve upon the metric proposed by Toivonen et al. [210], Gupta et al. [80] presented a new distance metric, called the conditional market-basket probability (CMPB) distance, based on which an agglomerative clustering algorithm is introduced for rule clustering. In their approach, the rules are embedded in a vector space by multi-dimensional scaling and clustered using a self-organizing map (SOM).

One of the domains for clustering association rules has been explorative mining. Tuzhilin and Adomavicius [211] utilized the rule clustering techniques in the analysis of microarray data in bioinformatics. In their approach, association rules are clustered based on gene hierarchies (as specified by the domain expert) where every rule cluster is uniquely represented by its aggregated rule (common to all

rules in that cluster). Also, unlike the traditional clustering methods, where the user has only a limited control over the structure and sizes of resulting clusters, a biologist has an explicit control over the granularity of the resulting rule groups [211].

2.6.3 Classification of Association Rules

Associative classification (AC) integrates association rule mining and classification to construct rule-based prediction models. Studies show that associative classification can be used effectively to classify resources and achieve a high precision compared with other sophisticated classifiers, like decision tree (DT), C4.5, naive Bayes (NB) and support vector machine (SVM) [228]. Moreover, many of the rules found by associative classification methods cannot be discovered by traditional classification techniques [134].

In general, an AC algorithm operates in three main phases [227]:

- Rule generation. Association rules are generated using various association rule mining techniques.
- Rule ranking. Rules are ranked according to defined parameters such as confidence and support. The output of the second phase is the set of classification association rules (CARs) which represent the final classifier model.
- Classifier building and rule pruning. The classification model is utilized to predict the class values on new unseen dataset (test data).

2.6.3.1 Rule Generation

Liu et al. [134] proposed one of the first algorithms to bring up the idea of using an association rule for classification, which is called CBA. CBA uses an iterative method which is similar to the Apriori algorithm to generate CARs with consequents restricted to a class attribute [6]. The biggest drawback of this approach is that since the database needs to be scanned many times, so the number of rules increases exponentially, and more system resources are consumed.

Several AC algorithms have been proposed to overcome the deficiencies of CBA. CBA (2) [136] tackles the problem of not generating CARs for minority class labels in the training dataset while ACAC [98] employs support and all-confidence [153] measures to select both frequent and mutual associated itemsets which contribute to classification. ACN [117] extends the Apriori algorithm to mine a relatively large set of negative association rules and then uses both positive and negative rules to build a classifier. Kundu et al. [118] proposed a new CBA-like algorithm called CARGBA, which merges rules generated from two steps. At the first step, a set of high confidence rules of smaller length with support pruning is generated using the Apriori algorithm, while at the second step rules with high confidence and rules of higher length with support below minimum support are also produced by using the Apriori algorithm but in a reverse manner.

Other approaches to improve the efficiency of CARs generation include ACCF [126] which uses the frequent closed itemset approach to improve the searching for frequent itemsets and ACCR [152], a metric measure of rules called “compactness” that stores rule items with low support but high confidence to ensure that high

quality rules are kept. Furthermore, Quinlan and Cameron-Jones [174] proposed the First Order Inductive Learner (FOIL). This learning strategy produces rules for each class of cases in the training data. Yin and Han [228] proposed a greedy AC algorithm called CPAR, which adopts the FOIL algorithm in generating the rules from datasets. CPAR selects multiple literals and builds multiple rules simultaneously. Also, it uses expected accuracy to evaluate rules and the best k rules in prediction.

FP-growth like approaches are also employed in the classified rule generation process. CMAR [125] adopts a variant of FP-growth to find the rules and stores them efficiently in a prefix tree structure, namely the CR-tree. Baralis and Garza [18] proposed a variation of the rule extraction part of the CMAR algorithm, utlizing lazy pruning techniques in their L^3(Live and Let Live) algorithm. Later, they proposed L^3_G which is an extension of L^3 and can provide a wider selection of rules obtained by allowing lower support thresholds [17].

2.6.3.2 Rule Ranking

Ranking generated rules is crucial since usually rules with higher ranks are tested first when predicting test cases and the resulting classifier accuracy depends heavily on rules used during the prediction phase. Most of the AC algorithms order the generated rules using a group of parameters such as confidence and support, where rules with high confidence and support receive higher ranks.

In CBA [134], rules are sorted according to antecedent length as well as confidence and support, that is, if two rules have the same confidence and supports, their antecedent lengths are compared and the rule with lower antecedent length receives

a higher rank. This sorting procedure is utilized by many AC algorithms such as CBA (2) [136], CARGBA [118] and ACCF [126].

In addition to the parameters that are used in the CBA algorithm, Thabtah et al. [207] proposed a new parameter called class distribution which represents the number of times a class occurs in the training dataset. Baralis and Garza [18] introduced a different sorting procedure in their L^3 lazy AC algorithm where the rules are sorted in decreasing length order, which is the opposite of the CBA rule ranking procedure.

2.6.3.3 Building Classifiers and Pruning

Lastly, the classification model is utilized to predict class values on new unseen datasets (test data). In AC, a classifier consists of a set of rules that is built from the training dataset. A major concern about AC algorithms is that they produce a relatively high number of rules that build the classifier [125], which slows the classification process. Also, some of these rules may be useless for the classifier and redundant. Redundant rules need to be discarded to increase the effectiveness, efficiency and accuracy of the classifier.

Liu et al. [134] proposed a database coverage technique to reduce the size of the classifier space. The technique checks whether each rule covers at least one object of the training dataset. If so, the rule is added to the classifier and its corresponding training object is deleted from the training dataset. This method has been utilized by many AC algorithms, including CBA (2) [136], CMAR [125], CAAR [222], ACN [117], and ACCF [126].

In the database coverage technique, since the objects covered by the rules are all

deleted, the selection of other rules derived from the deleted objects may be affected. To overcome this problem, Jiang et al. [103] proposed a rule pruning algorithm based on indiscernibility relationship. Baralis and Garza [18] introduced further improvements in rule pruning where a lazy pruning technique is utilized to discard from the classifier only the rules that do not correctly classify any training case. Antonie and Zaïane [11] introduced the concept of conflicting rules, which is defined as follows. Given two rules $R_1: X => C_1$ and $R_2: X => C_2$, R_1 and R_2 are conflicting rules because they hold the same antecedent (X) and belong to different class labels $C_1$and C_2. In their approach, all these duplicates or conflicting rules are eliminated.

Thabtah et al. [208] introduced the Looking at the Class (LC) prediction method, which selects the class with the highest average confidence value among the set of rules in the classifier for prediction. Abu-Mansour et al. [1] evaluated the correctness of the rule's class with that of the training data when covering the training case. Their test results show that the number of rules generated by the developed pruning procedure is usually less than those of lazy pruning and database coverage heuristics.

2.6.4 Rule Changing Monitoring

The world around us changes constantly. In recent years, methods and techniques have emerged to monitor the changes in association rules over time or location to help businesses to detect, assess and respond to changing conditions rapidly and intelligently.

2.6.4.1 Frameworks

Several frameworks have been proposed that can handle changes in association rules. Ganti et al. [73, 71, 72] presented a framework for measuring changes via two models. The difference between the two models is quantified as the amount of work (e.g., difference in supports) required to transform one model into the other.

Spiliopoulou and Roddick [197] provided a framework for modelling *higher order* association rules as temporal sequences of conventional rules obtained from different mining sessions. Mining sessions are defined as a 6-tuple, providing a signature to which higher order reasoning algorithms can refer. Higher order mining routines are then able to operate over temporal sequences of rulesets.

Later Baron and Spiliopoulou [21] proposed a temporal rule model, generic rule model (GRM), to model both the content and the statistics of a rule as a temporal object. A rule R is a temporal object with the following signature [21]:

$$R = (ID, query, timestamp, stats, body, head)$$

where ID is an identifier, ensuring that rules with the same $body$ (antecedent) and $head$ (consequent) have the same ID. The $query$ is the data mining query, while the $stats$ (statistics) depend on the rule type. For example, an association rule A => B with an identifier ID_i, support $\sigma = 0.10$ and confidence $\gamma = 0.60$ produced by query Q at time stamp τ is modelled as

$$\mathrm{R} = (\mathrm{ID}_i, \mathrm{Q}, \tau, [\sigma = 0.10, \gamma = 0.60], \mathrm{A}, \mathrm{B}).$$

Using the GRM to represent patterns, Baron, Spiliopoulou and Günther [24] proposed a general framework for pattern monitoring and change detection. In their work, the KDD process is divided into two phases: the mining phase and the

monitoring phase. In the mining phase, data from the first period is mined and interesting rules and patterns are identified. In the monitoring phase, rules to be monitored are firstly identified and statistics of those rules are then extracted and compared to predefined thresholds. If the statistics of a rule violate the user given thresholds, it is removed from the rule base [24].

Furthermore, based on the GRM, Baron and Spiliopoulou [22] introduced Pattern Monitor (PAM), a framework for observing changes to the behaviour of a web site's visitors. In PAM, a *change detector mechanism* is employed to identify changes to a rule's statistic which exhibit a particular strength. Statistical significance is used to assess the strength of pattern changes.

Based on the idea of detecting interesting changes in a dataset by analysing the support and confidence of association rules along the time axis, Böttcher et al. [33] presented a framework that pro-actively and automatically discovers interesting trends and stabilities in the support and confidence histories of association rules. In this approach, a time stamped dataset is partitioned into intervals along the time axis. Association rule discovery is then applied to each of these subsets. This yields sequences or histories of support and confidence for each rule, which can be analysed further in three layers: structural analyser, change analyser and interestingness evaluator, respectively [33].

2.6.4.2 Representation and Interpretation of Discovered Rules

The changes in rules can be represented and interpreted in different forms based on the nature of data and the requirements of the data mining task.

Active data mining [4] attempts to represent and query the history pattern of the

discovered association rules which are continuously generated at a desired frequency. The discovered rules from different time periods are collected into a rule base. The history, that is, ups and downs in support or confidence over time, is represented and defined using shape operators. The user can then query the rule base by specifying some history specifications.

Liu et al. [133] attempted to detect "fundamental changes" in a set of association rules, that is, changes that are responsible for all changes seen in the set. The proposed approach first generates rules and in the second phase it identifies changes (rules) that cannot be explained by the presence of other changes (rules). In their work, a statistical X^2 test for homogeneity of support and confidence is employed to evaluate rule changes.

Liu et al. [135] counted the significant rule changes across the temporal axis. The dataset is first partitioned into a few blocks or sub-datasets corresponding to the time periods (e.g., years, months or weeks) in which they were collected. Then association rules are mined from each block. Finally, the supports and confidences of the rules in these time periods are inspected to find various types of important rules, including [135]:

- stable rules - rules that do not change a great deal over time and, thus, are more reliable and can be trusted
- trend rules - rules that indicate some underlying systematic trends of potential interest.

Similarly, Au and Chan [13] defined three types of an association rule based on the changes of support or confidence value of each rule, including:

- a changed rule, if its support (confidence) in the period is different from its support (confidence) in the previous period
- a perished rule, if its support and/or confidence become less than the user-specified thresholds in the period, and
- an added rule, if its support and confidence become greater than or equal to the user-specified thresholds in the period.

In their work, they used linguistic variables and linguistic terms to represent the changes in discovered association rules and used fuzzy decision trees to discover the changes [13]. The fuzzy decision trees can then be converted to fuzzy meta-rules which are used to predict any change in the association rules in the future.

Chen and Petrounias [52] focused on the identification of valid time intervals for previously discovered association rules. They proposed a methodology that finds all adjacent time intervals during which a specific association holds, and furthermore all interesting periodicities that a specific association has.

2.6.5 Rule Maintenance

Data change over time, such as the continuous changes due to addition, deletion and modification of the contained data. Therefore, rules at a point in time may become invalid while new rules may come into existence and wait to be detected.

It is challenging to maintain and update discovered association rules if the database where the rules are generated is updated. Algorithms for efficiently updating the association rules have been proposed in the literature [55, 56, 187, 209, 25, 139]. These algorithms take the set of association rules in the old database into account and use this knowledge to remove itemsets that no longer exist in the updated

database and to add new itemsets which were not in the set of old transactions but now exist in the updated database.

Updating association rules was first introduced by Cheung et al. [54, 55]. They proposed the FUP (Fast UPdate) algorithm to deal with insertion of new transaction data. FUP is based on the Apriori algorithm and achieves significant efficiency because it avoids re-computations for itemsets which were already found to be large during mining of the database.

There are two main drawbacks of FUP. One is that it can only handle insertion of new transaction data and the other is it scans a database multiple times. The first issue has been ameliorated by Cheung et al. [56], who extended the work from FUP to handle deletion as well as addition. In order to reduce the number of database scans, Ayan et al. [15] proposed the Update With Early Pruning (UWEP) algorithm which scans the existing database at most once and the new database exactly once. It employs a dynamic look-ahead pruning strategy in updating the existing large itemsets by detecting and removing those that will no longer remain large after the contribution of the new set of transactions. This results in a much smaller number of candidates in the computation of new large itemsets.

When to update rules is an important question. Lee and Cheung [121] proposed an algorithm to estimate the difference between the association rules in a database before and after it is updated. The estimated difference can then be used to determine whether to update the mined association rules. If the estimated difference is sufficiently large, then it is time to update the mined association rules to discover and learn the new rules and discard the old ones. If the estimated difference is

considered small, then there is no need to spend the resources to update the rules, as the rules in the original database are still a good approximation for those in the updated database. Similarly, DELI (Difference Estimations for Large Itemsets) determines when to update rules using approximate upper/lower bounds on the amount of changes in the set of newly introduced association rules, where a low bound denotes small changes in association rules which require no maintenance [122].

Rainsford et al. [176] proposed a temporal windowing technique for incremental maintenance of association rules, which regards transactions outside a user-defined time window as too old and thus uninteresting. Their approach finds strong and near-strong association rules based on definitions of strong support and near-strong support threshold levels as well as the corresponding strong and near-strong confidence levels. Those near-strong rules might be become strong association rules during the next time window.

2.7 Summary

This chapter described the Knowledge Discovery in Databases (KDD) process and data mining. More particularly, it focused on association rule mining and higher order mining techniques.

Association rule mining is a data mining technology used to discover knowledge about patterns and associations between items of transactions in a database. Association rule mining is a two-stage process: the frequent itemset discovery stage and the rule generation stage. This chapter detailed different algorithms developed for rapid and efficient frequent itemset generation. Also, various methods have

been proposed in the literature to increase the efficiency of the rule generation process. Two of them were studied in this chapter: constraint-based association rule mining and redundancy rule reduction. Furthermore, this chapter reviewed some of the extensions of association rule mining, including quantitative association rule mining, multi-level association rule mining and temporal association rule mining.

Being a sub-field of data mining, HOM is concerned with mining over patterns/models derived from one or more large and/or complex datasets. This chapter explored HOM techniques briefly with a focus on works related to mining over association rules, including rule clustering, rule classification, rule change monitoring and rule maintenance.

HOM opens a new window for knowledge discovery from mining from the source to mining from the patterns/models. However, the overall potential of HOM is still largely unexploited and worthy of further research [183].

Chapter 3

Ruleset Pattern and Horace

Since the search for rules that can inform business decision making is the ultimate goal of data mining technology, problems such as the interpretation of interestingness for discovered rules is an important issue. However, as discussed in Chapter 1, association rules are commonly supplied in a low, instance-level format. Such low-level rules, while useful, provide knowledge only about the coincidence of elementary values and can be termed *zero-order* rules. *Higher order* semantics can be derived when sets of rules are inspected to determine patterns of interest between rules.

This chapter provides formal definitions of patterns in discovered association rules and presents Horace, a novel approach for ruleset pattern discovery.

3.1 Preliminaries

Given a rule r_i with antecedent X, consequent Y, support σ and confidence γ, it is denoted as:

$$r_i: X => Y(\sigma, \gamma)$$

where r_i is the *name* of the rule and $X \cup Y$ the *itemset* of r_i. For brevity, there are the following notations:

- $r_i.ac$: the antecedent of r_i, i.e., $r_i.ac = X$
- $r_i.cs$: the consequent of r_i, i.e., $r_i.cs = Y$
- $\sigma(r_i)$: the support of r_i
- $\gamma(r_i)$: the confidence of r_i
- $P(X \cup Y)$: the number of transactions containing the *itemset* of r_i

Definition 3.1 (Parent and Sibling Rule) Given two rules r_i and r_j, if $r_i.cs = r_j.cs \wedge r_i.ac \cap r_j.ac = \emptyset$, that is, they have the same consequent but disjointed antecedent, r_i is a sibling of r_j and vice versa. These are denoted as $Sib(r_i, r_j)$. Given multiple sibling rules, they are denoted as $Sib(r_1, r_2, \dots, r_n)$.

If $r_i.cs = r_j.cs \wedge r_i.ac \subset r_j.ac$, that is, they have the same consequent but the antecedent of r_j contains the antecedent of r_i, r_j is the parent of r_i, and r_i a child of r_j, which are denoted as $Par(r_j, r_i)$.

Given a parent rule r_p with a set of sibling rules as its children, $Sib(r_1, r_2, \dots, r_n)$, they are denoted as $Par(r_p, Sib(r_1, r_2, \dots, r_n))$.

Example 3.1 To illustrate, consider the ruleset below:

$$r_1: \{a\} => \{c\} (\sigma = 0.60, \gamma = 0.80)$$
$$r_2: \{b\} => \{c\} (\sigma = 0.70, \gamma = 0.75)$$
$$r_3: \{a, b\} => \{c\} (\sigma = 0.10, \gamma = 0.60)$$
$$|D| = 1000$$

Figure 3.1: Sample Ruleset

Since $r_1.cs = r_2.cs = \{c\}$, $r_1.ac \cap r_2.ac = \{a\} \cap \{b\} = \emptyset$, r_1 and r_2 are two siblings. Also, since $r_1.\mathrm{cs} = r_3.\mathrm{cs} = \{\mathrm{c}\}$, $r_1.\mathrm{ac} = \{\mathrm{a}\} \subset r_3.\mathrm{ac} = \{\mathrm{a}, \mathrm{b}\}$, r_3 is the parent of r_1. Similarly, we find that r_3 is a parent of r_2. Therefore, the three rules in Figure 3.1 can be denoted as $Par(r_3, Sib(r_1, r_2))$.

Definition 3.2 (Relative Support) Given a set of sibling rules $R = Sib(r_1, r_2, \dots, r_n)$, the relative support[1] ρ of rule $r_i \in R$ is defined as follows:

$$\rho(r_i) = \frac{P(r_i.ac \cup r_i.cs) - Q(r_i.ac \cup r_i.cs)}{|D|}$$

where $Q(r_i.ac \cup r_i.ac)$ denotes the number of transactions containing the antecedent and consequent of other rules in R in all transactions containing the antecedent and consequent of r_i.

Relative support represents the occurrence of the antecedent of a sibling rule without the existence of other sibling rules' antecedents, when occurring together with their consequent. To illustrate, let us take the following example.

Example 3.2 Consider the relative support of the two sibling rules r_1 and r_2 in the example shown in Figure 3.1. According to the definition of support, we have:

[1] This concept builds on the work by Shillabeer and Pfitzner[193]

$$\sigma(r_1) = \frac{P(r_1.ac \cup r_1.cs)}{|D|} \quad (1)$$

$$\sigma(r_2) = \frac{P(r_2.ac \cup r_2.cs)}{|D|} \quad (2)$$

$$\sigma(r_3) = \frac{P(r_3.ac \cup r_3.cs)}{|D|} \quad (3)$$

Thus, we have

$$\begin{aligned} P(r_1.ac \cup r_1.cs) &= P(\{a, c\}) \\ &= \sigma(r_1) \times |D| \\ &= 0.60 \times 1000 = 600 \end{aligned} \quad (4)$$

Similarly,

$$P(r_2.ac \cup r_2.cs) = 700 \quad (5)$$

$$P(r_3.ac \cup r_3.cs) = 100 \quad (6)$$

Result (4) shows there are 600 transactions containing items a and b, which are the antecedent and consequent of r_1 respectively. In order to calculate $\rho(r_1)$, we need to calculate $Q(r_1.ac \cup r_1.cs)$ which is the number of transactions containing the antecedent and consequent of its sibling, r_2, from result (4). Since $r_2.ac = \{b\}$, $r_2.cs = \{c\}$ and result (4) contains all transactions with $\{a, c\}$, it is clear that

$$Q(r_1.ac \cup r_1.cs) = P(\{a, b, c\}) = 100 \quad (7)$$

and therefore,

$$\rho(r_1) = 0.5 \quad (8)$$

$$\rho(r_2) = 0.6 \quad (9)$$

That is, item a occurs in 50% of transactions together with item c without the existence of item b, and b occurs in 60% of transactions together with item c without the existence of item a.

3.2 Defining Patterns in Rules

Let $R = \{r_1, r_2, \dots, r_n\}$ where $n > 1$ be a set of rules. A pattern in R is denoted as

$$RP = \{Rset|P\},$$

where $Rset = \{r|r \in R\}$, P is the condition(s) the $Rset$ holds. For brevity, we call RP a ruleset pattern.

A condition in a ruleset pattern is in the following form:

$$< attribute > op < attribute >, \text{ or}$$
$$< attribute > op < constant >$$

The $< attribute >$ is the *attribute* of the participating rule, including its name, the antecedent, consequent, support, confidence, and relative support. op is normally one of the operators $\{=, <, >, \geq, \leq, \neq\}$. The $< constant >$ is a constant value defined by end users or domain experts. Clauses can be arbitrarily connected by the Boolean operators AND, $OR$and NOT to form a general selection condition.

Definition 3.3 (Competitor Pattern) Given user-specified thresholds $minH$ and $maxL$, where $minH \geq maxL$, a competitor pattern is denoted as:

$$CoPatt = \{Par(r_p, sib(r_i, r_j))|$$
$$\rho(r_i) \geq minH,$$
$$\rho(r_j) \geq minH,$$
$$\sigma(r_p) \leq maxL,$$
$$\sigma(r_p) < \sigma(r_i) \times \sigma(r_j)\}$$

As shown in the above definition, a competitor pattern contains a parent rule r_p with two children rules r_i and r_j. The pattern requires that the relative support of r_i and r_j is higher than or equal to the user-specified threshold $minH$ and the support of the parent rule is lower than or equal to threshold $maxL$. It also requires that the itemsets of rules r_i and r_j are statistically negatively correlated.

The competitor pattern illustrates the relationship between the antecedents of rules

r_i and r_j, where one suppresses the other when occurring together with their consequent, resulting in unexpected low support. To illustrate, let us take the following example.

Example 3.3 Given $minH = 0.4$, $maxL = 0.2$ and take the three rules from Figure 3.1. Since $\rho(r_1) = 0.5 > minH$, $\rho(r_2) = 0.6 > minH$ and $\sigma(r_3) = 0.1 < maxL$, the first three conditions are satisfied. Furthermore, we have $\frac{\sigma(r_3)}{\sigma(r_1)\times\sigma(r_2)} = \frac{0.1}{0.6\times0.7} = 0.238 < 1$, thus $\sigma(r_3) < \sigma(r_1) \times \sigma(r_2)$. So the last condition is also met and we have found a matched instance which reveals that items a and b suppress each other when occurring together with c.

Definition 3.4 (Twoway-Catalyst Pattern) Given user-specified thresholds $minH$ and $maxL$, where $minH \geq maxL$, a twoway-catalyst pattern is denoted as:

$$Ca2Patt = \{Par(r_p, sib(r_i, r_j))|$$
$$\rho(r_i) \leq maxL,$$
$$\rho(r_j) \leq maxL,$$
$$\sigma(r_p) \geq minH,$$
$$\sigma(r_p) > \sigma(r_i) \times \sigma(r_j)\}$$

As shown in the above definition, the first three conditions require that $\rho(r_i)$ and $\rho(r_j)$ are lower than or equal to threshold $maxL$ and the support of the parent rule r_p is higher than or equal to threshold $minH$. The last condition requires that the itemsets of the two sibling rules r_i and r_j should be statistically positively correlated.

The twoway-catalyst pattern is similar to the competitor pattern except that it illustrates a positive relationship between the antecedents of the rules r_i and r_j,

where one facilitates the other when occurring together with their common consequent.

Definition 3.5 (Threeway-Catalyst Pattern) Given user-specified thresholds $minH$ and $maxL$, where $minH \geq maxL$, a threeway-catalyst pattern is denoted as:

$$Ca3Patt = \{Par(r_p, sib(r_i, r_j, r_k))|$$
$$\rho(r_i) \leq maxL,$$
$$\rho(r_j) \leq maxL,$$
$$\rho(r_k) \leq maxL,$$
$$\sigma(r_p) \geq minH,$$
$$\sigma(r_p) > \sigma(r_i) \times \sigma(r_j) \times \sigma(r_k)\}$$

Threeway-catalyst patterns illustrate the relationship between the antecedents of three sibling rules, which seldom occur individually, but more commonly occur together with their consequent.

Table 3.1: Sample Ruleset Patterns

Pattern	Ruleset	Description
Competitor Pattern	$\{cola\} => \{chips\}$ $\{lemonade\} => \{chips\}$ $\{cola, lemonade\} => \{chips\}$	Customers tend to buy $chips$ and $cola$ or $chips$ and $lemonade$ individually, but they seldom buy $chips$, $cola$ and $lemonade$ together.
Twoway -Catalyst Pattern	$\{milk\} => \{bread\}$ $\{butter\} => \{bread\}$ $\{milk, butter\} => \{bread\}$	When customers buy $bread$, they tend to buy $milk$ and $butter$ together but not individually.
Threeway -Catalyst Pattern	$\{turkey\} => \{Christmas\ cards\}$ $\{crackers\} => \{Christmas\ cards\}$ $\{ham\} => \{Christmas\ cards\}$ $\{turkey, crackers, ham\} => \{Christmas\ cards\}$	$Turkey$, $crackers$ and ham are frequently bought together with $Christmas\ cards$.

These three types of patterns in rulesets widely exist in the real world. Table 3.1 represents a descriptive list of the patterns. Ruleset patterns may be exhibited within many domains, for instance, in human resources where the productivity of a team can be affected by internal strife between its members. Also, in medicine, exposure to different conditions may result in an increased probability of illness. In addition, patterns in rulesets may combine to form more complex patterns. The conjunction of ruleset patterns is outside the scope of this thesis.

3.3 The Horace Approach

A straightforward approach to finding ruleset patterns is to search for such patterns from rules generated from other data mining techniques. However, this approach is inefficient for the following reasons.

Firstly, association rule mining often generates a large number of rules. To tackle the problem, support and confidence are used to prune those rules which do not meet the thresholds. However, setting too high or too low thresholds of support and confidence may not obtain satisfactory rules. If the threshold is set too high, the number of rules may be insufficient and some important associations may be filtered out. On the other hand, if the threshold is set too low, a huge number of rules will be generated and the rule generation process becomes intractable.

Secondly, the support or confidence might not be required in ruleset patterns. For example, confidence is not specified in patterns defined in this chapter. Therefore, when searching a ruleset for such patterns, the ruleset is incomplete as it only contains rules which satisfy the threshold of minimum confidence, resulting in inaccurate results.

3.3.1 Overview of Horace

To overcome the above drawbacks, this thesis proposes a novel tree-based approach, called Horace, for efficient and effective ruleset pattern discovery.

Frequent pattern or prefix trees, also known as tries, are generally used for frequent itemset generation [47], such as FP-tree [88, 89], COFI tree [59] and Patricia trie [171]. Since the frequent pattern or prefix trees are (generally speaking) isomorphic with the resulting ruleset, it is possible and more efficient to search such data structures directly for patterns.

Firstly, there is no need to generate all rules as a preliminary step, which significantly reduces the computational complexity and temporal overhead. Secondly, once the trees have been built, there is no pruning process required to remove rules which do not satisfy the threshold support and confidence. Information held in the trees is complete, ensuring the completeness and accuracy of searching results.

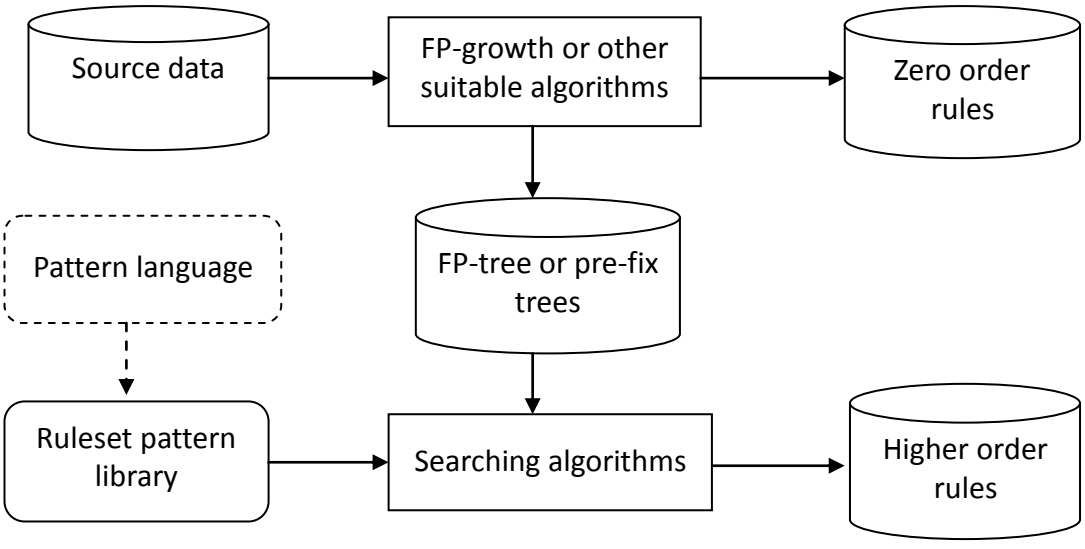

Figure 3.2: Overview of Horace

As shown in Figure 3.2, there are three key parts to the Horace framework: the FP-tree or pre-fix trees, a ruleset pattern library with an associated pattern language and a set of pattern search algorithms.

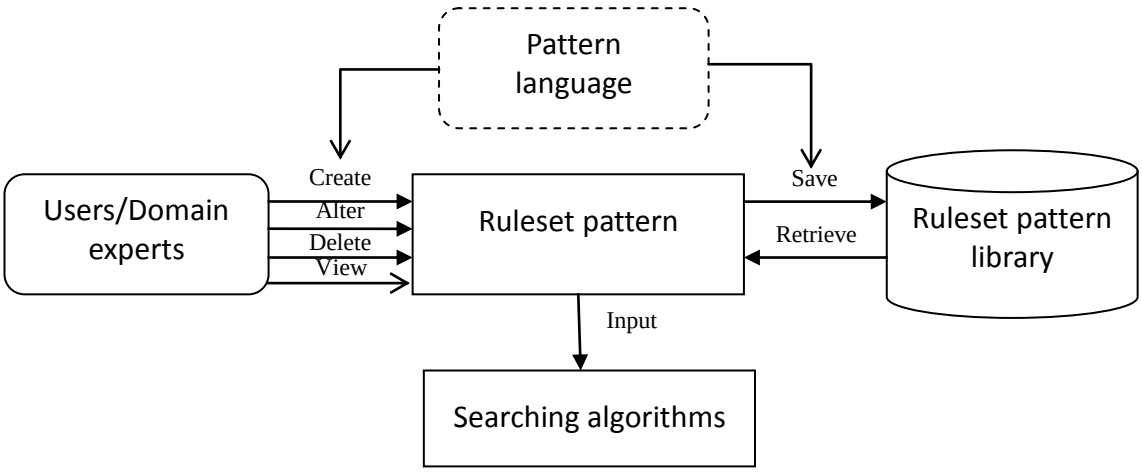

Figure 3.3: Overview of Ruleset Pattern and the Pattern Language

In this thesis, FP-tree [88] is employed for Ruleset Pattern searching[2]. The ruleset pattern library and its associated pattern language, as shown in Figure 3.3, play an important part in Horace. The ruleset pattern library stores a set of ruleset patterns, which can be defined, retrieved and maintained by end users or domain experts through a pattern language. The pattern language, which consists of a rule pattern definition language (RPDL) and a ruleset pattern query language (RPQL), provides the following functionalities:

- Pattern creation. Users can define patterns in rules based on their own definition of interesting.
- Pattern retrieval. Patterns can be retrieved from the ruleset pattern library

[2] The author of the thesis believes other tree-structures can also be utilized, although further experiments are needed to confirm this.

for viewing, maintenance or pattern searching.

- Pattern maintenance. This includes functions such as updating existing ruleset patterns or deleting unused ruleset patterns.

At the core of Horace, there is a set of searching algorithms to find all matches of a given ruleset pattern held in the frequent pattern or prefix trees. Horace represents patterns in rulesets using a novel ruleset pattern tree (RP-tree). A RP-tree has a set of prefix subtrees as the children of the root. Each node consists of three fields: the item, a node link and a count, where the node link points to the next node containing the same item in the RP-tree. Given the structure of the two trees, RP-tree and FP-tree, the search algorithms search the FP-tree for all matches with the RP-tree.

3.4 Related Work

With a few notable exceptions, data mining research has largely focused on the extraction of knowledge directly from the source data [128]. Current work directly in the area of this thesis is relatively limited.

Roddick et al. [183] discussed higher order mining more generally, proposing a list of higher order patterns. The work presented in this thesis differs from their work in the following aspects. Firstly, higher order patterns describe patterns from the previously induced patterns and models, such as association rules, clusters, classification rules and so on. In contrast, this thesis focuses on patterns in association rules only. In addition, in their work, higher order patterns are indicative and loosely described. In contrast, this thesis provides formal definitions

for patterns in association ruleset. Also, a tree-based approach for ruleset pattern discovery which cannot be handled by their work is proposed in this thesis.

Association rule mining can be performed under the guidance of various kinds of constraints provided by the user [67, 29, 159, 151, 199, 200]. For example, meta-rules allow users to specify the syntactic form of the rules they expected. The work presented in this thesis is different from constraint-based association rule mining. Firstly, ruleset patterns reveal interesting relationships in the antecedents or consequents of participating rules. They are not constraints to be applied to the rule generation process. Secondly, in constraint-based association rule mining, association rules are generated based on constraints specified, while in the process of ruleset pattern discovery, the phase of association rule generation is not required.

Rule templates were first studied by Klemettinen et al. [108]. In their work, a rule template is defined by users to specify what items are in a rule, and what restrictions are applied on support and/or confidence. Those rules which do not match the template are uninteresting and thus are pruned out. Klemettinen's work is different from the work presented in this thesis as rule templates are employed to find interesting rules from source data, while this thesis defines patterns of interest between sets of rules.

Teng [206] studied disassociation rules, which are rules that capture the negative relationship between two set of items. For example, the presence of X and Z is not a good predictor for the presence of Y. Dissociation rules are close to the competitor pattern as this chapter defined it. However, Teng's work focuses on the mechanism for learning dissociation from source data while the work presented

here aims to define and facilitate searching of patterns in rules.

In data mining, the usefulness of association rules is strongly limited by the huge amount of delivered rules. Post-processing methods, like pruning, summarizing, grouping, or visualization, are proposed to improve the selection of discovered rules [142, 16, 237]. The work presented in this thesis is different from those post-processing methods as ruleset patterns are normally defined before the mining process starts. Also, in the work presented herein, Horace searches the frequent pattern or prefix trees for ruleset patterns and therefore rule generation and post-processing are not involved.

3.5 Summary

Association rules are commonly supplied in a low-level or instance-level format. Higher order patterns can be discovered in a set of rules based on users' definitions of interesting. This chapter provides formal definitions for patterns in rules, named as ruleset patterns.

A straightforward approach to finding ruleset patterns is to search from association rules generated from other data mining techniques. However, it is problematic due to two factors. One is the complexity of the process of rule generation and the other is that the completeness of rulesets is heavily affected by the setting of support and confidence. To overcome these shortcomings, this chapter proposes a proof-of-concept system, Horace, which incorporates a tree-based approach for efficient and effective ruleset pattern discovery. The basic idea behind it is that since frequent pattern or prefix trees contain all of the information represented by the (larger) rulesets generated from them, it is firstly possible and secondly more efficient to

search such an intermediate data structure for patterns.

There are three key parts to the Horace framework: the frequent pattern or prefix trees, an RP library with an associated pattern language and a set of pattern search algorithms. A brief overview of the key components of Horace is provided in this chapter. Detailed implementation of the searching algorithms and experiment results are presented in Chapter 4 and the ruleset pattern language is presented in Chapter 5.

Chapter 4

Searching Ruleset Patterns Using FP-Trees and RP-Trees

Chapter 3 outlines Horace, a framework for ruleset pattern discovery. The core component of Horace is an efficient searching mechanism to find matches of a ruleset pattern. Horace represents ruleset patterns using a novel ruleset pattern tree (RP-tree). The searching algorithms then search the FP-tree for all matches of the RP-tree.

The following section reviews the FP-tree and Section 4.2 provides a detailed description of the new data structure (RP-tree) to represent ruleset patterns. Section 4.3 presents two searching algorithms and the experiment results are discussed in Section 4.4. Section 4.5 concludes the chapter.

4.1 FP-Tree

FP-tree [88] is a compact data structure that contains the complete set of information held in a database relevant to frequent pattern mining. FP-tree stores a single item at each node, and includes a support count and additional links to facilitate processing. These links start from a header table and link together all nodes in the FP-tree which store the same item. Details of the FP-tree construction were discussed in Section 2.3 in Chapter 2.

4.2 The Ruleset Pattern Tree (RP-Tree)

Horace represents ruleset patterns using a novel tree structure, ruleset pattern tree (RP-tree). To illustrate the construction of a RP-tree, let us first examine an example.

Example 4.1 Given a database D, user-specified thresholds $minH$ and $maxL$ and a competitor pattern $CoPatt$ containing three rules as follows:

$$r_i: \{b\} => \{a\}$$
$$r_j: \{c\} => \{a\}$$
$$r_p: \{b, c\} => \{a\}$$

then the pattern $CoPatt$ might be:

$$CoPatt = \{Par(r_p, Sib(r_i, r_j))|$$
$$\rho(r_i) \geq minH,$$
$$\rho(r_j) \geq minH,$$
$$\sigma(r_p) \leq maxL,$$
$$\sigma(r_p) < \sigma(r_i) \times \sigma(r_j)\}$$

A compact data structure can be designed based on the following observations:

- A list of itemsets in $CoPatt$ can be obtained after we scan it. We can then generate an ordered list of items in those itemsets based on the number of their occurrences (which can be registered as *frequencies*).
- For common items which exist in two itemsets, we might be able to merge them using one prefix structure if items in $CoPatt$ are sorted in their *frequency* descending order.

With the above observations, we construct a RP-tree for $CoPatt$ as follows. Firstly, we scan the pattern once to obtain the list of itemsets in the ruleset pattern and denote the list as *L*, where *L* = [{c, a}, {b, a}, {b, c, a}]. Also, we generate the set of ordered items in *L* based on their frequency in descending order and denote it as *I*, i.e., *I* = {a(3), b(2), c(2)}, where the number in () indicates the frequency of the item in the list of itemsets.

Secondly, the root of the RP-tree is created with a label "T". Then, we go through each itemset in *L* and insert them as follows. Firstly, we insert the itemset of the parent rule, which is {a, b, c}, resulting in the first branch of the tree, as shown in Figure 4.1(a). Then, we insert the second itemset in *L*, which is {a, b}. Since all items in {a, b} share a common prefix (a, b) with the existing path (a, b, c), no new branch is created as shown in Figure 4.1(b). Finally, the process proceeds to the third itemset, namely, {a, c}. Since only item a in this itemset share a common prefix with the existing path (a, b, c), a new branch is created. In addition, to facilitate tree traversal, a node link is created, which points to nodes with the same item-name, that is, the two "c" nodes. The resultant RP-tree is shown as Figure 4.1(c).

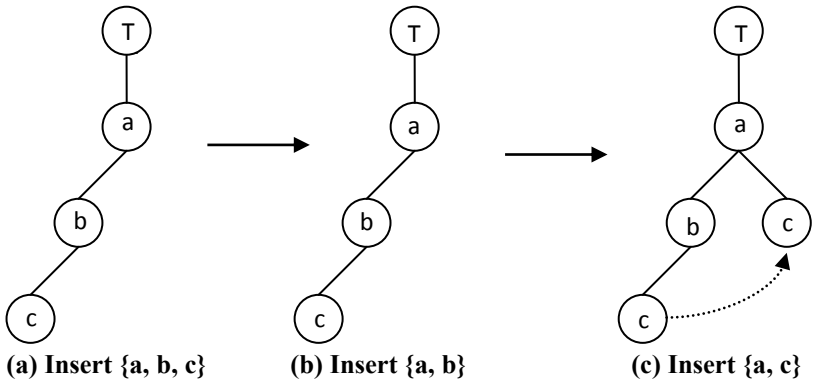

Figure 4.1: Sample RP-Tree Construction

Based on this example, a RP-tree can be defined as follows:

Definition 4.1 (RP-Tree) A RP-tree is a tree structure consisting of one root which is labelled "T", and a set of prefix subtrees as the children of the root. Each node in the prefix subtree consists of three fields: the item, a node link and a count, where the node link points to the next node containing the same item in the RP-tree.

Algorithm 4.1: RP-Tree Construction

1: **Input**: A ruleset pattern p
2: **Output**: A RP-tree
3: scan each rule in p, do the following:
4: a) set L to be the list of itemsets in p
5: b) set I to be the sorted items in L in their frequency count descending order
6: create root T
7: **for each** itemset s in L, starting from the itemset of the parent rule, **do**
8: insert-tree(s, T, 0)
9: **end for**
10: insert-tree(itemset s, treeNode n, int pos)
11: **if** $pos < s.length$ **then**
12: **if** $n.child == null$ or $n.child$.item-name $\neq s[pos]$.item-name **then**
13: create new node n', set $n'.item = s[pos]$, $n.child = n'$ and the node-link of n' points to the nodes with the same item-name of n'
14: **end if**
15: insert-tree(s, $n.child$, $pos++$)
16: **end if**

Based on this definition, the RP-tree construction algorithm is outlined in Algorithm 4.1. As shown in line 3 to line 5, there is only one scan required to build

a RP-tree which generates the list of itemsets (L) in the ruleset pattern and the ordered list of items (I) in L. Starting from the itemset of the parent rule, each itemset in L is then inserted through a function called insert-tree (as shown in lines 10 to 16).

Given an itemset s, the function inserts each item based on their order in I. If the node the item will be inserted into has a child and the item-name of the child node is the same as that of the inserted item, it proceeds to the next node. Otherwise, a new child node is added with a node-link pointing to the nodes with the same item-name. The function is recursively executed until all itemsets in L are inserted.

After construction, a language is required to describe the RP-tree. A RP-tree is defined as a collection of tuples <node, parent, [Nchildren]>, where Nchildren is the collection of the node's children from left to right. In addition, the set of conditions the ruleset pattern holds and a label (denoted as *desc*), which is an instantiated description of the pattern, can be imposed over the definition. Thus the tuples which are constructed for all non-leaf RP-tree nodes of the RP-tree in Figure 4.1(c) are:

$$< T, , [a] >$$
$$< a, T, [b, c] >$$
$$< b, a, [c] >$$
$$\rho(\{a, b\}) \geq minH,$$
$$\rho(\{a, c\}) \geq minH,$$
$$\sigma(\{a, b, c\}) \leq maxL,$$
$$\sigma(\{a, b, c\}) < \sigma(\{a, b\}) \times \sigma(\{a, c\}),$$
$$desc: \text{"Competitor pattern: "} + b + \text{","} + c$$

4.3 Algorithm Development

Given the structure of the two trees, RP-tree and FP-tree, in this section, two Searching Ruleset Patterns using FP-tree (SRPFP) algorithms are proposed to search the FP-tree for matches of the RP-tree.

4.3.1 The *SRPFP-a* Algorithm

The *SRPFP-a* algorithm (as shown in Algorithm 4.2) provides a mechanism for tree searching, which substitutes RP-tree nodes with the items from the FP-tree header table.

Algorithm 4.2: *SRPFP-a* Algorithm

1: **Input**: FP-tree fp, RP-tree rp
2: **Output**: Set of matched instances of rp $RPset$
3: set I to be the items in $fp.header$ in their support descending order
4: RP-mine ($rp.root.child$)
5: RP-mine (RPtreeNode $node$)
6: **for each** item i in I **do**
7: $node$.item-name = i.item-name
8: **if** $node.link \neq null$
9: $node.link$.item-name = i.item-name
10: **end if**
11: **if** $node.child == null$ **then**
12: calculate support count for nodes in branches of $node.root.child$ from fp
13: **if** !isValid ($node$.root.child, $rp.conditions$) **then**
14: return
15: **else**
16: $RPset$.Add($node$.root.child)
17: **end if**
18: **else**
19: RP-mine ($node.child$)
20: **end if**
21: **end for**

As the RP-tree's leftmost branch contains all items in the parent rule, *SRPFP-a* requires a single pass over this branch for search purposes. For each node in this parent rule branch, all other nodes representing the same item are accessible

through the node links. The collection of itemsets that terminate with a specific item is efficiently obtained through this structure.

During the process to calculate support, *SRPFP-a* visits the relevant FP-tree branches by following the link of the specific item in the header table. If a visited branch contains the parent rule's itemset, the support of the itemset increases by adding the support count of the visited node in the branch. If there is no parent rule's itemset in the branch and it contains a child rule's itemset, the relative support of the itemset increases by adding the support count of the visited node. If there is a parent rule's itemset in the branch and the branch contains more than one child rule's itemsets, those itemsets are not independent of each other in that branch so the support count of the visited node will not be counted according to the definition of relative support.

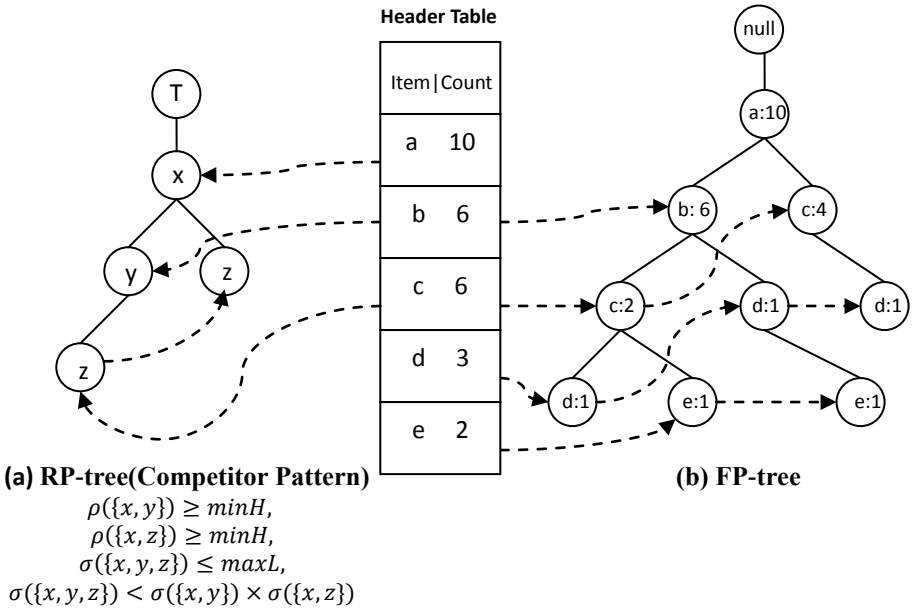

Figure 4.2: Searching Algorithm Illustration (*SRPFP-a*)

Example 4.2 To illustrate, consider the example shown in Figure 4.2, which contains a competitor pattern tree (RP-tree) and an FP-tree. Given that $minH = 0.4$, $maxL = 0.2$ and $|D| = 10$, the algorithm starts with node 'x' in the RP-tree. Each node in the RP-tree will be replaced by its corresponding value from the FP-tree header table.

Starting with item 'a', only one branch ends with 'a', and a support count of 10 is obtained from the header table. The process continues, moving to the next node in the traversal, 'y'. The node 'y' is replaced by all items in the header table, except 'a'. Starting with item 'b', a branch is generated from the RP-tree that ends with 'b' and also contains 'a', which is {a, b}. Following the FP-tree link from 'b', all branches that contain {a, b} are found. The support count, which is 6, is thus obtained. The process advances to 'z', which is replaced with each item in the FP-tree header table except 'a' and 'b'.

Starting with 'c', it generates a list of branches in the RP-tree that contain 'c' and have 'a' or 'b' in their parental path, specifically {a, b, c} and {a, c}. The support count of {a, b, c} and {a, c} are obtained, which are 2 and 6 respectively. Since it is the last node of the parent rule branch, a condition check will be performed for the three related itemsets, {a, b, c}, {a, b} and {a, c}. The relative support of {a, b} (0.4) and {a, c} (0.4) are calculated. Since $\sigma(\{a,b,c\}) = 0.2 = maxL,\ \rho(\{a,c\}) = 0.4 = minH$, and $\rho(\{a,b\}) = 0.4 = minH$, the first three of the conditions for the competitor pattern are met. Furthermore, since $\frac{\sigma(\{a,b,c\})}{\sigma(\{a,b\})\times\sigma(\{a,c\})} = \frac{0.2}{0.6\times0.6} = 0.56 < 1$, and $\sigma(\{a,b,c\}) < \sigma(\{a,b\}) \times \sigma(\{a,c\})$, all of the conditions of the pattern have been met. Therefore, a matched instance of the RP-tree has been found.

The algorithm then progresses to ‘d’. Branches containing ‘d’ that have ‘a’ or ‘b’ in their parental path are {a, b, d} and {a, d}. The support count of {a, b, d} and relative support of {a, d} are obtained, namely, $\sigma(\{a, b, d\}) = 0.2, \rho(\{a, d\}) = 0.1$. Since $\rho(\{a, d\}) = 0.1 < minH$, it is invalid and thus is pruned. The process continues until all nodes in the leftmost branch of the pattern tree have been substituted.

The complexity of the *SRPFP-a* algorithm is $O(\frac{n!}{(n-k)!k!})$, where n is the number of frequent items in an FP-tree and k is the number of distinct items in a RP-tree. The performance of *SRPFP-a* is heavily affected by the value of n and k. With an increase in n and k, the computational expense becomes high.

In addition, no pruning strategy is employed during the tree node substitution process, which is less efficient. For example, in the above example, when node x and y are substituted, *SRPFP-a* is not able to check the relative support of the itemset represented by the tree path as it cannot be calculated unless all itemsets of the ruleset patterns are identified. Therefore, itemsets with invalid relative support are used for further processing, which adds unnecessary computation time.

4.3.2 The *SRPFP-b* Algorithm

To overcome the drawbacks of the *SRPFP-a* algorithm, *SRPFP-b* is proposed, which provides a more efficient searching mechanism to deal with complex FP-trees and RP-trees. The *SRPFP-b* algorithm (as shown in Algorithm 4.3) consists of two phases: k_p-itemset generation (where k_p is the length of the itemset of the parent rule) and RP-tree instance generation. At the k_p-itemset generation step, *SRPFP-b* finds all valid k_p-itemsets from the FP-tree. At the second step, it

constructs a RP-tree instance for each itemset returned from the previous step which is then validated against the conditions associated with the ruleset pattern.

Algorithm 4.3: *SRPFP-b* Algorithm

1: **Input**: FP-tree fp, RP-tree rp
2: **Output**: Set of matched instances of rp $RPset$
3: set I to be the sorted items in $fp.header$ in their support ascending order
4: **for each** item i in I **do**
5: set S = RP-growth (fp, i, $|rp.parent|$, $rp.parent.condition$)
6: **for each** itemset s in S **do**
7: sort items in s in their support descending order in $fp.header$
8: set rp' = RP-Substitution ($rp.root.child$, s, 0)
9: **if** !isValid(rp', $rp.condition$) **then**
10: return
11: **else**
12: $RPset$.Add($rp'.root.child$)
13: **end if**
14: **end for**
15: **end for**
16: RP-growth (FP-Tree $Tree$, node a, int len, condition c)
17: **if** $Tree$ contains a single path P
18: **for each** combination (denoted as q) of the nodes in the path P **do**
19: generate pattern $q \cup a$ with $|q \cup a| == len$ and condition c is met
20: **end for**
21: **else**
22: **for each** a_i in the header of $Tree$ **do**
23: generate pattern $q = a_i \cup a$ with $|q| == len$ and condition c is met
24: construct q's conditional pattern base and q's conditional FP-tree $Tree\_q$
25: **if** $Tree\_q \neq \emptyset$ **then**
26: call RP-growth ($Tree\_q$, q, $len--$, c)
27: **end if**
28: **end for**
29: **end if**
30: RP-Substitution(node n, itemset s, int $index$)
31: **if** $index > s.length - 1$ **then**
32: calculate support count of s from fp
33: **end if**
34: $n.item = s[index]$
35: **if** $s.link \neq null$ **then**
36: $n.link.item = s[index]$
37: **end if**
38: RP-Substitution ($n.child$, s, $index++$)

4.3.2.1 k_p-Itemset Generation

In this step, *SRPFP-b* employs a function called RP-growth to find all valid k_p-itemsets from the FP-tree. RP-growth is a variation of the FP-growth algorithm [88, 89] with the following two main differences: (1) compared to FP-growth which generates all frequent itemsets, RP-growth only generates k_p-itemsets; (2) FP-growth employs the minimum support threshold to ensure only frequent itemsets are returned. In contrast, RP-growth does not require minimum support and k_p-itemsets might not be frequent. The above differences are reflected by the changes in lines 16, 19 and 23 respectively in Algorithm 4.3.

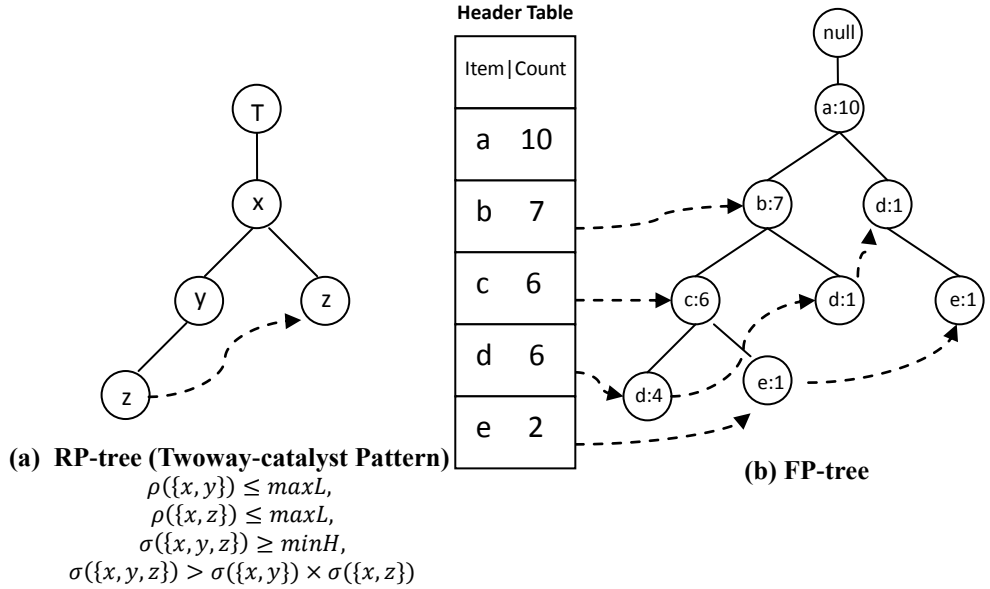

Figure 4.3: Searching Algorithm Illustration (*SRPFP-b*)

Example 4.3 To illustrate, let us take the example shown in Figure 4.3, which contains a twoway-catalyst pattern tree (RP-tree) and an FP-tree. The initial parameter values are $minH = 0.4$, $maxL = 0.2$ and $|D| = 10$. Since the itemset of the parent rule ({x, y, z}) has length 3, the algorithm firstly finds all 3-itemsets

from the FP-tree that satisfy the condition associated with the parent rule ($\sigma(\{x, y, z\}) \geq minH$), that is, the support count should be greater than $minH \times |D| = 0.4 \times 10 = 4$.

Starting from item ‘e’ in the header table, since the support count of ‘e’ is 2 which is less than the minimum support count, the process moves to the next item in the header table, which is ‘d’. Item ‘d’ occurs in two branches of the FP-tree, which are (a, b, c, d: 4) and (a, b, d: 1). Taking ‘d’ as the suffix, the corresponding two prefix paths are (a, b, c: 4) and (a, b: 1) which forms the conditional pattern base. There is one branch in the conditional FP-tree and we obtain all 3-itemsets, namely, {a, b, d}(5), {a, c, d}(4) and {b, c, d}(4). They all are valid as their support counts are greater than or equal to the minimum support count threshold associated with the parent rule.

The process then proceeds to item ‘c’. Similarly, we obtain the branch containing item ‘c’, which is (a, b, c: 6). There is only one branch and we obtain the valid 3-itemset {a, b, c}(6).

Since items ‘a’ and ‘b’ are the first two items in the header table, they are not considered as there will be no branches terminating at these items with length greater than 2. Therefore, we have the following valid 3-itemsets: {a, b, d}(5), {a, c, d}(4), {b, c, d}(4) and {a, b, c}(6).

4.3.2.2 RP-Tree Instance Generation

Given a set of itemsets *L*, *SRPFP-b* constructs a RP-tree instance for each itemset in *L*. For an itemset *s* in *L*, nodes in the leftmost branch of the RP-tree, that is, the parent rule branch, are substituted by items in *s* based on descending order of their

frequency. Nodes in branches other than the leftmost branch of the RP-tree are substituted through the node link in the RP-tree. Since the parent rule branch contains all items in the ruleset pattern, the process ensures that all nodes in the RP-tree are substituted. Validation occurs when the substitution process is completed, where the support and relative support count are calculated through the node link structure in the FP-tree.

Example 4.4 For illustration, consider the RP-tree and FP-tree in Figure 4.3 as an example. As indicated in the previous section, the set of 3-itemsets are {a, b, d}(5), {a, c, d}(4), {b, c, d}(4) and {a, b, c}(6).

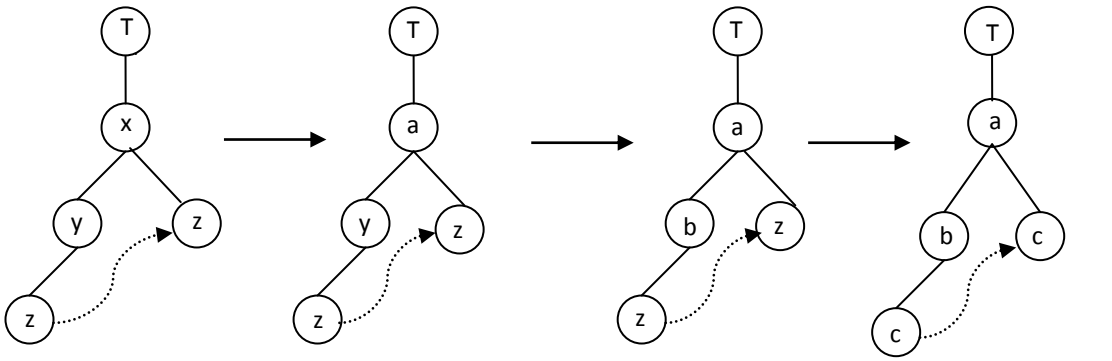

Figure 4.4: RP-Tree Substitution Process

Starting with itemset (a, b, d: 5), node x, y and z in the leftmost branch are substituted with item ‘a’, ‘b’ and ‘d’ respectively. The process is shown in Figure 4.4. As shown in Figure 4.4(d), when the node z in the leftmost branch is substituted with item ‘c’, the node in another branch with the same item-name is substituted by following the node link.

After substituting all nodes in the RP-tree, the process validates the generated RP-tree instance against the conditions associated with the pattern tree. The support

and relative support are calculated from the FP-tree through the node link structure, that is, $\sigma(\{a,b,d\}) = 0.5$, $\sigma(\{a,b\}) = 0.7, \sigma(\{a,d\}) = 0.6$, $\rho(\{a,b\}) = 0.2$ and $\rho(\{a,d\}) = 0.1$. Since $\sigma(\{a,b,d\}) = 0.5 > minH$, $\rho(\{a,b\}) = 0.2 = maxL$, $\rho(\{a,d\}) = 0.1 < maxL$, the first three conditions are met. Also, since

$$\frac{\sigma(\{a,b,d\})}{\sigma(\{a,b\}) \times \sigma(\{a,d\})} = \frac{0.5}{0.7 \times 0.6} = 1.19 > 1,$$

we have

$$\sigma(\{a,b,d\}) > \sigma(\{a,b\}) \times \sigma(\{a,d\}).$$

All conditions have been met and we have found a matched RP-tree instance.

Compared to the step of k_p-itemset generation, the step to construct a RP-tree instance is less complex. On one hand, the RP-tree is usually much smaller than the FP-tree and the substitution of the nodes in the parent rule branch is straightforward. On the other hand, during the validation process, it is more efficient to calculate the support count and relative support count through the node link structure in the FP-tree. Therefore, the entire performance of *SRPFP-b* is determined by the k_p-itemset generation step.

4.4 Experiments and Analysis

To demonstrate the concept, a prototype of Horace was implemented in Java and several experiments were conducted on both synthetic and real datasets. All tests were done on a 2.6 GHz PC with 2GB of main memory running Windows 7. This implementation is shown to be tractable and able to reveal patterns in rulesets of potential interest that would otherwise not be reported.

A synthetic data generator was built based on the work reported by Agrawal and Srikant [6] to produce large quantities of transactional data. Table 4.1 shows the

parameters for data generation, along with their default values and the range of values on which experiments were conducted. Table 4.2 presents an overview of some generated synthetic data.

Three real datasets were used to test *SRPFP-a* and *SRPFP-b*. Their details are shown in Table 4.3. The retail data was sourced from an anonymous Belgian retail supermarket store [37]. The data were collected over three non-consecutive periods between 1999 and 2000. The two datasets BMS-WebView-1 and BMS-WebView-2 were taken from KDDCUP 2000 [113]. They contain several months' worth of click stream data from two e-commerce web sites.

Figure 4.5 shows the patterns tested, while Table 4.4 shows detailed description.

Table 4.1: Synthetic Data Parameters

Name	**Description**	**Default Value**	**Range of Values**
|I|	Number of Items	10	10-100
|T|	Number of Transactions	5,000	5,000-200,000
|P|	Number of Patterns	50	50-200
TS	Average Size of Transaction	5	5-10
PS	Average Size of Pattern	5	5-10

Table 4.2: Synthetic Data

Data	**|I|**	**|T|**	**|P|**	**|TS|**	**|PS|**
Syn1	100	10,000	20	5	5
Syn2	200	50,000	150	10	10
Syn3	300	100,000	250	10	15

Table 4.3: Real Datasets

Data	**Retail-Data**	**BMS-WebView-1**	**BMS-WebView-2**
NumberofTrans	88,163	59,602	77,512
Distinct Items	16,470	497	3,340
Max TransSize	67	267	161
Average TransSize	15	2.5	5.0

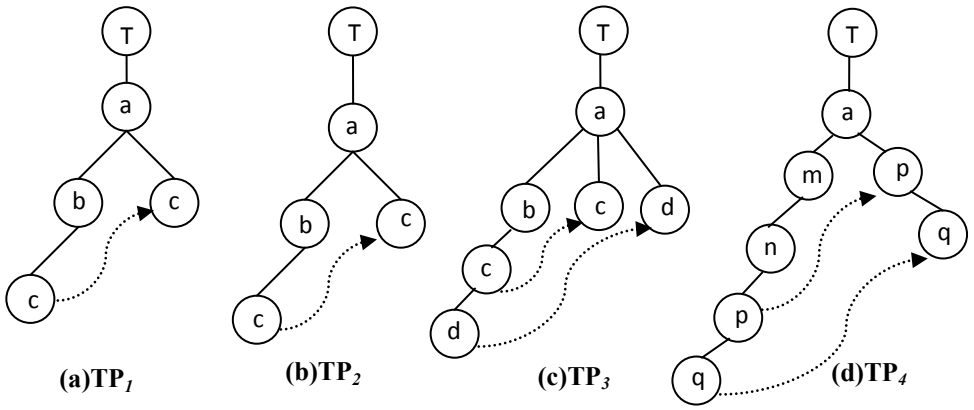

Figure 4.5: Test Patterns

Table 4.4: Description of Test Pattern

Test Pattern	Pattern Type	Conditions	Description
TP_1	Competitor pattern	$\rho(\{a,b\}) \geq minH$, $\rho(\{a,c\}) \geq minH$, $\sigma(\{a,b,c\}) \leq maxL$, $\sigma(\{a,b,c\}) < \sigma(\{a,b\}) \times \sigma(\{a,c\})$	Item b competes with item c when occurring together with item a.
TP_2	Twoway-catalyst pattern	$\rho(\{a,b\}) \leq maxL$, $\rho(\{a,c\}) \leq maxL$, $\sigma(\{a,b,c\}) \geq minH$, $\sigma(\{a,b,c\}) > \sigma(\{a,b\}) \times \sigma(\{a,c\})$	Item b facilitates item c when occurring together with item a.
TP_3	Threeway-catalyst pattern	$\rho(\{a,b\}) \leq maxL$, $\rho(\{a,c\}) \leq maxL$, $\rho(\{a,d\}) \leq maxL$ $\sigma(\{a,b,c,d\}) \geq minH$, $\sigma(\{a,b,c,d\}) > \sigma(\{a,b\}) \times \sigma(\{a,c\}) \times \sigma(\{a,d\})$	Items b, c, d facilitate each other when occurring together with item a.
TP_4	Competitor pattern	$\rho(\{a,m,n\}) \geq minH$, $\rho(\{a,p,q\}) \geq minH$, $\sigma(\{a,m,n,p,q\}) \leq maxL$, $\sigma(\{a,m,n,p,q\}) < \sigma(\{a,m,n\}) \times \sigma(\{a,p,q\})$	Items m and n compete with items p and q when occurring together with item a.

4.4.1 Results and Performance Study

The experimental results demonstrate that both *SRPFP-a* and *SRPFP-b* provide a sound and useful means of finding complex RP-tree patterns within an FP-tree. Test results, as shown in Table 4.5, demonstrate that the two algorithms are able to reveal patterns of potential interest based on users definition of interesting.

Table 4.5: Test Results

Dataset	min-sup	FP-tree Info			TP_1			TP_2			TP_3			TP_4		
		Depth	Branches	Nodes	minH (%)	maxL (%)	Cnt	minH (%)	maxL (%)	Cnt	minH (%)	maxL (%)	Cnt	minH (%)	maxL (%)	Cnt
Retail Data	0.01	12	12,142	31,037	1.0	0.1	6	0.5	0.3	1	1.0	0.5	0	0.5	0.5	48
BMS-WebView1	0.01	31	5,584	16,909	0.5	0.5	11	0.8	0.1	52	0.5	0.5	0	0.8	0.2	10
BMS-WebView2	0.005	28	14,044	48,571	0.5	0.5	5	0.5	0.5	0	0.5	0.5	0	0.8	0.2	8
Syn1	0.2	46	5,842	96,159	1.0	0.5	54	1.0	0.5	32	1.0	0.5	44	1.0	0.5	10
Syn2	0.05	23	17,426	127,463	1.0	0.1	144	0.2	0.2	132	1.0	0.5	3	0.8	0.2	2
Syn3	0.1	20	11,699	52,897	0.5	0.5	97	0.5	0.5	2	0.5	0.5	0	0.5	0.5	270

Patterns TP_1, TP_2 and TP_4 exist in both the real and synthetic datasets, while pattern TP_3 exists in two synthetic datasets[3]. Presented below are some examples discovered from Retail-Data (each item is denoted as a character c plus a number):

$$\{c39, c2925\}\ (\rho = 1.1\%),$$
$$\{c39, c1146\}\ (\rho = 1.1\%),$$
$$\{c39, c2925, c1146\}\ (\sigma = 0.009\%)$$

Description: item c2925 competes with c1146 when they occur together with item c39.

$$\{c14945, c101, c236\}\ (\rho = 0.5\%),$$
$$\{c271, c270, c236\}\ (\rho = 0.8\%),$$
$$\{c14945, c101, c271, c270, c236\}\ (\sigma = 0.005\%)$$

Description: itemset {c14945, c101} competes with itemset {c271, c270} when they occur together with item c236.

$$\{c39, c682\}(\ \rho = 0.24\%),$$
$$\{c48, c682\}(\ \rho = 0.14\%),$$
$$\{c39, c48, c682\}(\ \sigma = 0.53\%)$$

Description: item c39 facilitates item c48 when they occur together with c682.

[3]The author of the thesis believes TP_3 exists in the real word, although further experiments with more real datasets are needed to confirm this.

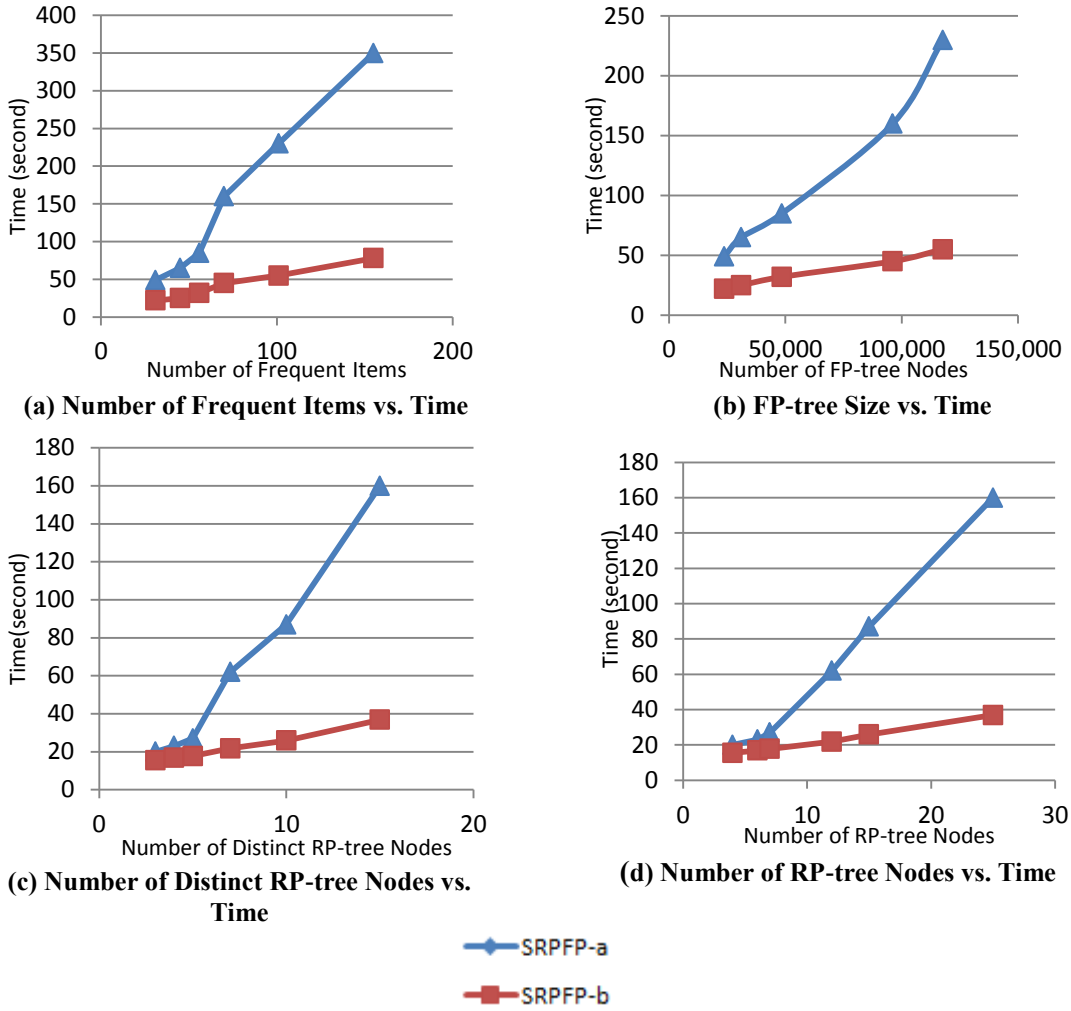

Figure 4.6: Performance Comparison

In the experiments, two groups of tests were conducted to investigate the effect of the size of the FP-tree and RP-tree on the execution time of the two searching algorithms. In the first group of tests, a set of synthetic datasets were generated with the number of transactions ranging from 10k to 150k. The number ranges of the frequent items and tree nodes of the resultant FP-trees were from 30 to 250 and from 20k to 150k respectively. Then pattern TP_1 was used to search for its matches from those FP-trees. The test results are presented in Figure 4.6(a) and 4.6(b).

In the second test group, two synthetic datasets (Syn1 and Syn2) and a set of test

ruleset pattern trees are used where the number of distinct tree nodes and total tree nodes varies from 3 to 12 and from 4 to 22 respectively. The test results are shown in Figure 4.6(c) and 4.6(d).

The test results of the two groups of tests show that the execution time of *SRPFP-a* and *SRPFP-b* increased with the increase of the size of the FP-tree and RP-tree. However, as shown in Figure 4.6(a) and Figure 4.6(c), the computation time of *SRPFP-a* sharply increased with the increase of the number of frequent items of the FP-tree and the number of distinct items of the RP-tree. Similar results are shown in Figure 4.6(b) and 4.6(d), revealing that *SRPFP-b* scales much better than *SRPFP-a* when the size of the FP-tree and RP-tree increases.

SRPFP-b is more efficient than *SRPFP-a* in handling large and complex FP-trees and RP-trees for the following two reasons:

Firstly, since *SRPFP-a* substitutes RP-tree nodes with items from the FP-tree header table, its performance is heavily affected by the number of frequent items in the FP-tree and the number of distinct items in the RP-tree. When the number of frequent items in the FP-tree or the number of distinct items in the RP-tree is large, the computation time of the algorithm becomes expensive. In contrast, *SRPFP-b* provides a different approach for pattern searching which does not involve the heavy tree node substitution process. As discussed in Section 4.3, the entire performance of *SRPFP-b* is mainly determined by the step of k_p-itemset generation. This process is performed on a pattern-base B_{ai} by constructing a conditional FP-tree for B_{ai}. Since B_{ai} is usually much smaller than its original FP-tree and the conditional FP-tree is usually much smaller and never bigger than B_{ai}, each

subsequent mining process/step works on a set of usually much smaller pattern bases and conditional FP-trees [88]. Therefore, *SRPFP-b* provides a faster approach to finding matched k_p-itemsets.

Secondly, *SRPFP-a* does not prune invalid itemsets until the parent rule branch of the RP-tree has been processed, which is less efficient. In contrast, *SRPFP-b* prunes out itemsets which do not meet the conditions associated with the parent rule of the ruleset pattern or have a length that is not equal to k_p during the searching process. This greatly reduces the computation overhead.

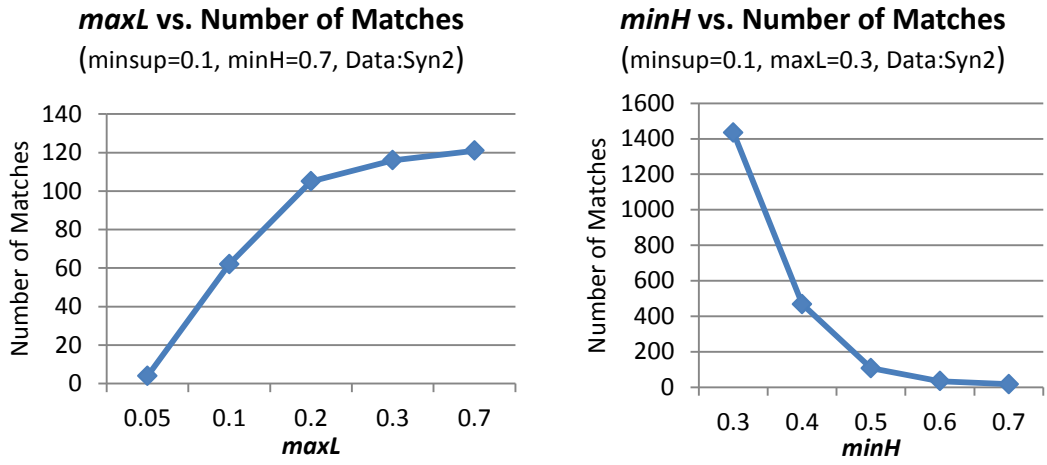

Figure 4.7: Effect of *minH* and *maxL*

The experiments also examined the effect of setting *minH* and *maxL* on the searching results. A set of tests were conducted on dataset Syn2 and pattern TP_1 using various values of *minH* and *maxL*. As shown in Figure 4.7, the number of matched instances is affected by the setting of *minH* and *maxL*. The higher the *minH* or the lower the *maxL*, the more itemsets with lower support are pruned out, and therefore the fewer matched instances found. Similarly, the lower the *minH* or the higher the *maxL*, the more itemsets with lower support are included, and

therefore, more matches can be identified.

4.5 Summary

This chapter presents a novel approach to finding matches for a defined ruleset pattern through using two data structures: the FP-tree and the RP-tree. The RP-tree is a tree structure which consists of one root and a set of prefix subtrees as the children of the root. An associated language is created to describe a RP-tree and to impose the description and conditions of a ruleset pattern over the tree.

Given the FP-tree and RP-tree, two algorithms (*SRPFP-a* and *SRPFP-b*) have been proposed to search the FP-tree for matching RP-tree instances. *SRPFP-a* provides a mechanism for tree searching by substituting RP-tree nodes with the items from the FP-tree header table. The computational complexity of *SRPFP-a* is determined by the number of frequent items in an FP-tree and the number of distinct items in a RP-tree. Therefore, it is less efficient when performing pattern searching which involves complex FP-trees and RP-trees.

The *SRPFP-b* employs a different approach for pattern searching which consists of two steps. At the first step, all valid k_p-itemsets are generated from the FP-tree using an FP-growth like algorithm, RP-growth. At the second step, a RP-tree instance is built and the conditions associated with the RP-tree are validated to ensure that only valid RP-tree instances are returned.

A prototype has been built to demonstrate the feasibility and efficiency of the two algorithms. The experimental results have demonstrated the capacity of both *SRPFP-a* and *SRPFP-b* to find patterns of potential interest. However, *SRPFP-b*

uses less computation time and is more efficient when dealing with large and complex FP-trees and RP-trees.

Chapter 5

RPL: A Ruleset Pattern Language

One essential part of Horace is a pattern library and its associated pattern language to enable users to define, retrieve and update ruleset patterns based on their needs. This chapter presents the ruleset pattern language (RPL), which consists of a ruleset pattern definition language (RPDL) and a ruleset pattern query language (RPQL). RPL provides a tool for end users to define patterns based on their own definition of interesting, to maintain patterns in the ruleset pattern library, and to retrieve patterns efficiently.

5.1 Towards a Ruleset Pattern Language - RPL

RPL consists of a rule pattern definition language (RPDL) and a ruleset pattern query language (RPQL). RPDL is a language for defining ruleset patterns, providing users with the ability to create, alter or delete ruleset patterns, while RPQL is a language for retrieving patterns from a ruleset pattern library. The RPL

language is defined in an extended BNF grammar, where square brackets [] around an element represent zero or one occurrence and {} represents zero or more occurrences. Also, given a ruleset pattern *p*, there is a set of notations that can be used in RPL (as shown in Table 5.1).

Table 5.1: Notations of Ruleset Pattern *p*

Notation	Description
p.name	The name of pattern *p*
p.rules	The set of rules in *p*
p.rules[i]	The i^{th} rule in *p*
p.rules.cs	The set of consequents of all rules in *p*
p.rules.ac	The set of antecedents of all rules in *p*
p.constraints	The set of constraints of *p*
p.constraints [i]	The i^{th} constraint in *p*
p.constraints [i].name	The name of the i^{th} constraint in *p*
p.constraints [i].expression	The expression of the i^{th} constraint in *p*

5.1.1 Ruleset Pattern Definition Language (RPDL)

RPDL was developed to create, alter, rename or delete ruleset patterns.

Create Statement. The syntax of the create statement is as follows:

CREATE *pattern* <pattern_name> (
{<rule_spec>},
{<constraint_spec>})
<rule_spec>::= *rule* <rule_name> (<antecedent>, <consequent>)
<antecedent>:: = {<itemset>}
<consequent>::= {<itemset>}
<constraint_spec> ::= *constraint* <constraint_name> (<expression_spec>)

In RPDL, the statement <rule_spec> is the specification of the class of rules to be included in a pattern. The word *rule* is a key word, indicating a single rule. <rule_name> is the name of the rule, which can include any character or characters. Each antecedent and consequent is a set of more or less tightly specified itemsets.

Similar to <rule_spec>, the statement <constraint_spec> specifies the constraints in a ruleset pattern. <constraint_name> is the name of a constraint and <expression_spec> specifies the content of a constraint, which is represented by one or more mathematical statements. A constraint is defined by end users or domain experts based on their own definition of interesting. Apart from the two commonly used measurements of interestingness, support and confidence, other quality metrics, such as relative support, might be utilized to define a constraint in a ruleset pattern.

For illustration, given user defined thresholds *minH* and *maxL*, a ruleset pattern is created in the following query.

Query 1: *create a pattern called Competitor*

CREATE *pattern* 'Competitor' (
rule r_1({x},{z}),
rule r_2({y},{z}),
rule r_3({x,y},{z}),
constraint c_1 $(\rho(r_1) \geq minH)$,
constraint c_2 $(\rho(r_2) \geq minH)$,
constraint c_3 $(\sigma(r_3) \leq maxL)$,
constraint c_4 $(\sigma(r_3) < \sigma(r_1) \times \sigma(r_2)))$

Query 1 contains three rules (r_1, r_2 and r_3) and four constraints (c_1, c_2, c_3 and c_4) where the first two conditions involve the quality metric relative support.

Alter Statement. The alter statement modifies an existing ruleset pattern. It has the following syntax:

```
ALTER pattern <pattern_name>
{
  SET {
            antecedent = {any_character}
          | consequent = {any_character}
          | constraint = <expression_spec>
      }
  | ADD{<rule_spec> | < constraint_spec>}
  | DELETE{rule <rule_name> | constraint <constraint_name>}
}
```

There are three commands to update a ruleset pattern: SET, ADD and DELETE. Their usages are illustrated with some examples as follows.

The command SET updates a rule or constraint in a ruleset pattern as shown in the following two example queries.

Query 2: *update the antecedent of rule r_1 of pattern 'Competitor' to {w}*

```
ALTER pattern 'Competitor'
SET r1.ac = {w}
```

Query 3: *update constraint c_1 of pattern 'Competitor' to be '$\sigma < minH$'.*

```
ALTER pattern 'Competitor'
SET c1.expression = 'σ < minH'
```

The command ADD is used to add a rule or constraint to a ruleset pattern. One example is shown in Query 4.

Query 4: *add rule r_1: w=>z and constraint $c_1(\sigma > minH)$ to ruleset pattern 'Competitor'*

```
ALTER pattern 'Competitor'
ADD rule r1({w},{z}),
      constraint c1 (σ > minH)
```

Finally, the command DELETE removes a rule or a constraint from a ruleset pattern. Query 5 and 6 provide two examples of its usage.

Query 5: *delete rule r_1 from ruleset pattern 'Competitor'*

ALTER *pattern* 'Competitor'
DELETE *rule* r_1

Query 6: *delete constraint c_1 from pattern 'Competitor'*

ALTER *pattern* 'Competitor'
DELETE *constraint* c_1

RENAME Statement. The rename statement renames a ruleset pattern. It has the following syntax:

RENAME *pattern* <pattern_name>
SET *name* = {any_character}

Query 7 shows an example of the usage of the rename statement.

Query 7: *updates the name of pattern 'Competitor_A' to be 'Competitor_B'*

RENAME *pattern* 'Competitor_A'
SET *name* = 'Competitor_B'

DROP Statement. The last RPDL statement is the DROP statement which is used to delete a ruleset pattern. Its syntax is:

DROP *pattern* <pattern_name>

The usage of the DROP statement is straightforward as shown in Query 8.

Query 8: *delete a pattern called 'Competitor'*

DROP *pattern* Competitor

5.1.2 Ruleset Pattern Query Language (RPQL)

RPQL allows users to retrieve patterns from a ruleset pattern library, denoted as

Patternbase. The structure of a basic RPQL query is:

SELECT p
FROM *Patternbase* p
WHERE <conditional_expression>

The SELECT clause selects one variable over its corresponding *Patternbase* that satisfies the conditions of the WHERE clause.

A condition in the WHERE clause is defined in one of the following forms.

- {<pattern_name> | <rule_name> | <constraint_name> | <expression_spec> } *op* const

  The symbol *op* is one of the notations In, $InEqual$ and $Equal$, which are related to the standard relational operators $\subset$, $\subseteq$ and $=$ respectively. const can be textual or numeric or a mathematical expression.

- [ALL|ANY] {<antecedent> | <consequent>} *op* <itemset>

  ANY and ALL are two key words used to specify whether the condition should be applied to any or all of the antecedents or consequents in a ruleset pattern. For example, given a ruleset pattern p, the statement "ALL p.rules.cs $InEqual$ {z}" requires that the consequents of all rules in p should contain itemset {z}.

- Count(R) *relation_op* const

  Count (R) is a function returning the number of elements in R, where R is a list of rules or constraints.

Similar to SQL, more than one simple condition can be joined together to form a complex <conditional_expression>. It also supports negations. Some examples of the basic RPQL queries are provided as follows.

Query 9: *find patterns named 'Competitor'*

SELECT p
FROM *Patternbase* p
WHERE p.name = 'Competitor'

Query 10: *find patterns where all of its rules' consequents are {z}*

SELECT p
FROM *Patternbase* p
WHERE ALL p.rules.cs $Equal$ {z}

Query 11: *find patterns with more than one rule and constraint and the first rule's antecedent contains {x}*

SELECT p
FROM *Patternbase* p
WHERE Count(p.rules) > 1
AND Count(p.constraints) > 1
AND {x} $InEqual$ p.rules[1].ac

Query 12: *find patterns with more than one rule, all of which have consequent {z} and at least one rule whose antecedent is {x}.*

SELECT p
FROM *Patternbase* p
WHERE COUNT(p.rules) > 1
AND ANY p.rules.ac $Equal$ {x}
AND ALL p.rules.cs $Equal$ {z}

The above four queries show that RPQL is a small, yet expressive and powerful language that allows a rich variety of queries about the ruleset patterns.

5.2 Implementation of RPQL

The user-defined ruleset patterns are stored in a database, where each ruleset

pattern is assigned a unique ID. The manipulation of RPDL, including alteration, deletion and updating, is in line with SQL syntax. This section focuses on the evaluation of RPQL queries.

One straightforward method to retrieve ruleset patterns is to scan all patterns in the *Patternbase* for matches. This approach is applicable if the size of the *Patternbase* is small. In order to cope with a huge number of patterns and complex queries, RPQL employed the techniques of indexing to ensure evaluation efficiency.

5.2.1 Indexing

Inverted lists are a common indexing method used in text information retrieval, where each document is regarded as a set of keywords or items. Inverted lists are very efficient for set-oriented operations and also fast to build. Since the antecedent and consequent of a rule are sets of items, inverted lists are a suitable indexing technique when querying rules in ruleset patterns.

Inverted lists indexing works as follows: given a *Patternbase* PB, an antecedent inverted list (denoted as aL) and a consequent inverted list (denoted as cL) are created to index items in the antecedent and consequent of all rules in PB respectively. The generated inverted lists index has two parts: a vocabulary and a list. The vocabulary contains the distinct items in the antecedent or consequent, while the list stores pattern IDs, each of which is paired to an integer representing the position of the rule where the item belongs.

The inverted list for an item i in aL or cL is denoted as $aL(i)$ or $cL(i)$. The example below shows a consequent inverted list for item *w* in Table 5.2:

cL(w): {(2,1),(2,2),(2,3),(2,4),(3,1),(3,2)(3,3),(5,1),(5,2),(5,3)}

Each pair between the two curly braces in cL(w) shows a pattern ID and the position of the rule where item *w* appears. For example, (2, 1) indicates item *w* exists in the first rule's consequent of the pattern with ID 2.

The list of IDs in aL or cL for item *i* at the j[th] rule is denoted as $aL(i,j)$ or $cL(i,j)$ respectively. For example, cL(w, 1): [2, 3, 5] shows that item *w* exists in the first rule's consequent of three patterns with ID 2, 3 and 5 respectively.

Table 5.2: Sample Ruleset Patterns

Pattern ID	1[th] Rule		2[nd] Rule		3[rd] Rule		4[th] Rule	
	antecedent	consequent	antecedent	consequent	antecedent	consequent	antecedent	consequent
1	{x}	{z}	{y}	{z}	{x,y}	{z}		
2	{x}	{w}	{y}	{w}	{z}	{w}	{x,y,z}	{w}
3	{x,y}	{w,z}	{p,q}	{w,z}	{x,y, p,q}	{w,z}		
4	{x,y}	{z}	{m }	{z}	{n}	{z}	{x,y,m,n}	{z}
5	{x,y,v}	{w,z}	{m}	{w,z}	{x,y,v,m}	{w,z}		
6	{p}	{z}	{q}	{z}	{p, q}	{z}		

Note: for brevity, pattern name and constraints have been omitted.

5.2.2 Evaluating Queries

Algorithm 5.1 illustrates the process of RPQL query evaluation. As shown in lines 3 and 4, two inverted lists (aL and cL) are built and one variable $IDList$ is created, which holds all ruleset patterns' ID in the *Patternbase*. With the usage of the inverted lists, lines 6 to 15 update $IDList$ if the antecedent or consequent appears in the WHERE clause. Lines 17 to 21 scan the *Patternbase* to find those patterns whose ID is in $IDList$ and also satisfy other conditions.

Algorithm 5.1: RPQL Query Evaluation

1: **Input**: *Patternbase* pb, RPQL query q
2: **Output**: $AnswerSet$
3: build inverted list aL and cL
4: set $IDList$ = list of all patterns' ID from pb
5: set $ruleConditions$ = new List()
6: **for each** $condition\ c$ in $q.conditions$
7: **if** $c.rule \neq null$ **then**
8: **if** $c.rule.ac \neq null$ **then**
9: updateID(aL, $c.rule.ac$, $c.rule.pos$)
10: **else if** $c.rule.cs \neq null$ **then**
11: updateID (cL, $c.rule.cs$, $c.rule.pos$)
12: **end if**
13: $ruleConditions$.Add(c)
14: **end if**
15: **end for**
16: set $otherConditions = p.conditions$.Except($ruleConditions$)
17: **for each** pattern p in pb
18: **if** *IDList*.contains($p.ID$) ^ p.isValid($otherConditions$) **then**
19: $AnswerSet$.Add(p)
20: **end if**
21: **end for**
22: updateID(invertedList L, itemset I, int pos)
23: **for each** item i in I
24: **if** L.contains(i, pos) **then**
25: $IDList = IDList \hat{}\ L(i, pos)$
26: **end if**
27: **end for**

To illustrate, consider the following query against patterns in Table 5.2.

Query 13: *find patterns containing three rules and the antecedent of the first rule contains {x,y} and the second rule's antecedent is {m}.*

SELECT p
FROM *Patternbase* p
WHERE {x,y} $InEqual$ p.rules[1].ac (1)
AND p.rules[2].ac $Equal$ {m} (2)
AND Count (p.rules) = 3 (3)

To evaluate Query 13, we first set $IDList$ to contain the list of IDs for all patterns,

$$IDList = [1, 2, 3, 4, 5, 6]$$

We then evaluate the first condition: {x,y} $InEqual$ p.rules[1].ac. Below shows the inverted antecedent lists for item x and y:

$$aL(x): \{(1,1),(1,3),(2,1),(2,4),(3,1),(3,3)(4,1),(4,4),(5,1),(5,3)\}$$
$$aL(y): \{(1,2),(1,3),(2,2),(2,4),(3,1),(3,3)(4,1),(4,4),(5,1),(5,3)\}$$

Since the condition requires items x and y to be in the first rule, we have:

$$aL(x, 1): [1, 2, 3, 4, 5]$$
$$aL(y, 1): [3, 4, 5]$$

Thus $IDList$ can be updated as

$$\begin{aligned} IDList &= IDList \wedge aL(x,1) \wedge aL(y,1) \\ &= [1, 2, 3, 4, 5, 6] \wedge [1, 2, 3, 4, 5] \wedge [3, 4, 5] \\ &= [3, 4, 5] \end{aligned}$$

The process then goes to the second condition, p.rules[2].ac $Equal$ {m}. Similarly, we have

$$aL(m): \{(4,2),(4,4),(5,2),(5,3)\}$$
$$aL(m, 2): [4, 5]$$

$IDList$ is then updated as

$$IDList = IDList \wedge aL(m,2) = [3, 4, 5] \wedge [4, 5] = [4, 5]$$

Finally, we evaluate the third condition (Count (p.rules) = 3) by scanning those patterns having ID in the $IDList$ and then check each of them to see whether the condition is met. Since pattern 4 has 4 rules, it is pruned out. Thus, only pattern 5 meets all conditions and is returned for the query.

5.3 Experiments

To demonstrate the concept, a prototype of RPL was implemented in C# and

several experiments were conducted on synthetic datasets. All experiments were carried on an Intel Core i5 PC with 2G memory. Database SQL 2008 was used to store user defined ruleset patterns.

A ruleset pattern generator was built to create synthetic test data. Rules in test ruleset patterns are derived from a dataset generated based on the work reported by Agrawal and Srikant [6]. Constraints in test ruleset patterns are in the format:

QM *op* CONST

where QM and CONST represent randomly generated quality measurements and constants respectively. Table 5.3 presents the details of the generated synthetic data and Table 5.4 shows the parameters for data generation.

Table 5.3: Synthetic Data

Data	**|P|**	**|R|**	**|C|**	**|QM|**	**|UC|**
Data A	500	3	3	3	2
Data B	5000	5	5	5	4
Data C	10000	8	8	10	6
Data D	20000	10	10	12	8

Table 5.4: Synthetic Data Parameters

Name	**Description**	**Default Value**	**Range of Values**
|P|	Number of patterns	500	0.5k – 20k
|R|	Average number of rules per pattern	3	2-20
|C|	Average number of constraints per pattern	3	2-20
|QM|	Number of quality measurements	3	1 - 10
|UC|	Number of user-defined constants	2	1 - 10

The performance of the proposed system was tested on the four queries presented in Section 5.1.2. The execution times of these queries are presented in Figure 5.1. Each execution time is the average result for 10 runs for the query, and includes the time for loading patterns from the *Patternbase* and the time for building appropriate indices.

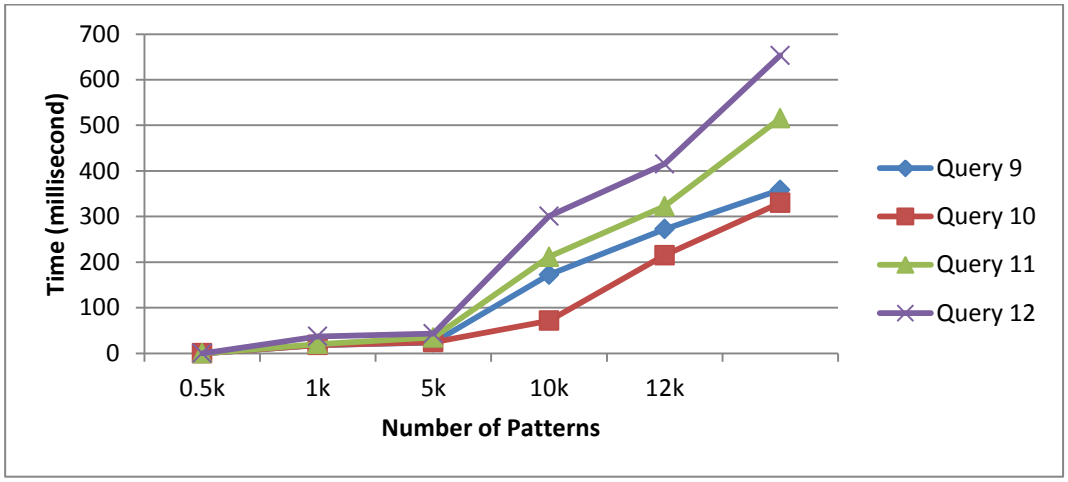

Figure 5.1: Execution Times of the 4 Representative Queries

As shown in Figure 5.1, RPQL queries are executed very efficiently. Query 11 and 12 have longer execution time than the other two as they contain more complex conditions in the WHERE clause. However, it took only 0.65 seconds to process Query 12 over 20,000 patterns. Also, it is interesting to see that Query 9, which is the simplest query, has a greater execution time than Query 10. This is because Query 10 employed inverted lists to speed up query processing while Query 9 was evaluated by scanning the whole *Patternbase*, which is less efficient.

5.4 Related Work

Association rule mining often generates a large number of rules most of which are actually not useful or interesting for specific applications [108]. Making sense of such a large number of rules has become a significant challenge. One approach to tackling this problem is identifying the rules that are of special importance to the user through data mining queries. The query language acts as an interface between the user and the knowledge and database. It allows the user to process data and knowledge and to direct the discovery process.

Agrawal et al proposed a shape definition language, called SDL, for retrieving objects based on shapes contained in the histories associated with the objects [5]. SDL enables users to define ups and downs of supports or confidences of a rule over a number of time periods. SDL is a definition language which focuses on behaviour shapes of the rules. The ruleset pattern language (RPL) presented in this thesis is built on ruleset patterns and it describes the relationships among items in participating association rules. RPL is also comprised of a pattern query language (RPQL) with the capability to retrieve patterns from the ruleset pattern library.

DMQL is a data mining query language proposed by Han et al [85]. With DMQL, users can select different tables (and databases) to mine different types of rules. Similar to DMQL, Meo et al. [146] proposed an SQL-like operator for data mining (MINE RULE) and Shen et al. [192] reported a meta query language for data mining. In addition, dmFSQL was proposed by Carrasco et al. [46] as an extension of FSQL (Fuzzy SQL) for data mining. It extends the SQL language with the capabilities to specify flexible queries to address tables that store vague information using fuzzy attributes [46]. One common characteristic of these query languages is that they are designed for generating rules from source data. In contrast, RPL is designed for users to define, alter and query patterns from a set of rules.

Several techniques that have been proposed to query discovered rules, including MSQL [99, 100] and Rule-QL [212]. MSQL can be used not only for rule generation, but also for querying the discovered rules. Rule-QL queries multiple sets of association rules and provides efficient algorithms for processing the

queries. RPL differs from those techniques in two aspects. It allows users to define patterns among a set of rules, which cannot be handled by MSQL and Rule-QL. In addition, RPQL queries ruleset patterns but not rules. A ruleset pattern is not a single rule but consists of a set of rules and conditions defined by users.

5.5 Summary

This chapter proposed RPL, a ruleset pattern language which consists of a ruleset pattern definition language (RPDL) and a ruleset pattern query language (RPQL). RPL is a small, yet expressive and powerful language that allows a rich variety of queries about the ruleset patterns. It enables end users to create, alter and retrieve ruleset patterns from the pattern library.

RPL is a natural and necessary complement to the ruleset pattern searching algorithms presented in Chapter 4. The emphasis of this chapter is on its expressiveness and functionalities towards end users. The syntax of RPDL and RPQL has been presented with illustrative examples in this chapter. Also, inverted lists are employed as an indexing technique to improve the efficiency of RPQL query evaluation. A set of experiments have been conducted which have demonstrated the efficiency of RPQL.

Chapter 6

Detecting Anomalies in Longitudinal Association Rules

The detection of unusual or anomalous data is an important function in automated data analysis or data mining. This thesis provides a partial solution to this problem by elevating the search for anomalous data in transaction-oriented datasets to an inspection of the rules that can be produced by *higher order* longitudinal/spatio-temporal association rule mining.

The next section discusses research to date in anomaly detection and longitudinal and spatio-temporal knowledge discovery which outlines the context to this work. Section 6.2 then discusses the research in anomaly detection in data as represented through association rules. Section 6.3 provides details of two anomaly detection algorithms and Section 6.4 presents the implementation and experiment results. Finally, Section 6.5 concludes the chapter.

6.1 Motivation and Literature Review

6.1.1 Anomaly Detection

Anomaly detection refers to the problem of finding patterns in data that do not conform to expected behaviour [50]. Such patterns, which are usually called outliers, noise, or novelty in different application domains, often contain useful information regarding the abnormal behaviour of the system described by the data. For example, an anomalous traffic pattern in a computer network could mean that a hacked computer is sending out sensitive data to an unauthorized destination. An anomalous MRI image may indicate the presence of a malignant tumour. Originally studied in the statistics community in the nineteenth century [58], anomaly detection is now a widely researched problem and has found immense use in application domains [162] such as:

- fraud detection: detection of criminal activities occurring in commercial organizations
- intrusion detection: monitoring the events occurring in a computer system or network for unusual behaviour and analysing them for intrusions
- medicine and public health: using unusual symptoms or test results to indicate potential health problems
- image processing: detecting anomalies in an image monitored over time or anomalous regions within an image.

The main anomaly detection approaches can be categorized into the following groups: statistics-based techniques, proximity-based techniques, density-based techniques, classification-based techniques, and clustering-based techniques.

Statistics-based anomaly detection techniques treat data instances occurring in the low probability regions of the statistical distribution as anomalies and those occurring in high probability regions of a statistical distribution as normal [50]. These approaches break the process of anomaly detection into two steps. A statistical distribution is estimated using given data at the first step and statistical inference tests are then applied in the second step to verify whether a test instance belongs to this distribution [93, 226, 225, 12, 185, 105]. Statistics-based anomaly detection techniques are built on standard statistical techniques and thus have a firm foundation. However, since they rely on the assumption that the data is generated from a particular distribution, they are not applicable for detecting anomalies in data with unknown distribution, such as high dimensional data [50]. In addition, it is difficult to select the best statistical method for anomaly detection [149].

Proximity-based anomaly detection techniques define a proximity measure between objects and try to find those objects that are distant from most of the other objects [162]. One of the simplest ways to measure whether an object is distant from most points is to use the distance to the *k*-nearest neighbour [43]. Since originally being used to detect land mines from satellite ground images, the *k*-nearest neighbour method has been extended by researchers with a focus on the definition modifications to obtain the anomaly score of a data instance [60, 10, 235, 32, 110, 111, 112], the use of different distance/similarity measures to handle different data types [156, 116, 161] and improvements in anomaly detection efficiency [28, 178, 144, 60]. Distance-based anomaly detection techniques are simple and easy to implement. However, the approach is too expensive for large

datasets due to the computational complexity ($O(m^2)$, where m is the number of objects). Also, the efficiency of the anomaly detection process is sensitive to the choice of parameters [162].

Density-based anomaly detection techniques compute local densities of particular regions and declare instances in low density regions as potential anomalies [50]. Breunig et al. [35, 36] introduced the concept of a local outlier factor (LOF). For any given data instance, the LOF score is equal to the ratio of average local density of the k-nearest neighbour of the instance and the local density of the data instance itself. Values significantly larger than 1 indicate outliers. Several variants of LOF have been proposed in the literature to enhance the estimation of the local density of an instance [205, 92, 41, 163], handle different types of data [202, 203, 172, 231] or improve its efficiency [57, 104]. Density-based anomaly detection techniques are easy to adapt and purely data driven as they do not make any assumptions regarding the generative distribution for the data [50]. However, like distance-based anomaly detection techniques, these approaches have $O(m^2)$, time complexity. Also, the parameter selection can be difficult for complex data, such as graphs and sequences [162].

Classification-based anomaly detection techniques often build a model for normal (and anomalous) events based on labelled training data and then use it to classify each new unseen event [50]. Those approaches consist of two phases: the training phase which learns a classifier using the available labelled training data and the testing phase which classifies a test instance as normal or anomalous. Association rule mining has also been utilized in building classifiers to capture the normal

behaviour of a system. For example, in the ADAM (Audit Data Analysis and Mining) system [19], association rules are used to gather necessary knowledge about the nature of the audit data. Lee et al. [123] used the association rules and frequent episodes computed from audit data as the basis for guiding the audit data gathering and feature selection processes. Similarly, He et al. [94] proposed an anomaly detection algorithm for categorical datasets in which the anomaly score of a test instance is equal to the number of frequent itemsets in which it occurs.

Clustering techniques which are used to group similar data instances together, have also played an important role in anomaly detection. One of the approaches is based on an assumption that normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster [62, 79, 230]. Some other work assumes that normal data instances lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid [194, 42, 214, 20]. Furthermore, research is being conducted on detecting anomalies based on cluster sizes where large clusters correspond to normal data and the rest of the data points are outliers [167, 60]. Clustering techniques such as K-means have linear or near-linear time and space complexity and therefore, an outlier detection technique based on such algorithms is efficient. However, the set of outliers produced and their scores can be heavily dependent upon the number of clusters used as well as the presence of outliers in the data [162].

6.1.2 Longitudinal and Spatio-Temporal Knowledge Discovery

The popularity of data mining, together with the mounting recognition of the value of temporal and spatial data, spatio-temporal data modelling and databases has

resulted in the prospect of mining spatial and temporal rules from both static and longitudinal/temporal/spatial data[4]. The accommodation of time and location into mining techniques provides a window into the spatio-temporal arrangement of events and affords the ability to suggest cause and effect, which are otherwise overlooked when this component is ignored or treated as a simple numerical attribute. The importance of longitudinal and spatio-temporal data mining is its capacity to analyse activity rather than just states and to infer relationships of locational and temporal proximity. Moreover, temporal data mining has the ability to mine the behavioural aspects of (communities of) objects as opposed to simply mining rules that describe their states at a point in time.

For example, temporal association rule mining accepts a set of keyed, time-stamped datasets and returns a set of rules indicating not only the confluence of events or attribute values (as in conventional association mining [47]) but also the arrangement of these events in time. Such routines can reveal otherwise hidden correlations, even in static rules.

Data mining techniques have been successfully applied in diverse application domains including health, defence, telecommunications, commerce, astronomy, geological survey and security. In many of these domains, the value of knowledge obtained by analysing the changes to phenomena over time and space, as opposed to the situation at an instant or at a single location, has been recognized and a number of temporal and spatial data mining techniques have been developed [182, 61]. For example, spatio-temporal rules can indicate movement, trends and/or

[4] The term "*longitudinal*" is used to mean a set of data ordered in time or space.

patterns that static rules are unable to show. However, apart from the computational complexity involved in introducing any new dimension, a number of challenging problems have arisen, three of which are described below.

The first is the efficient, automated determination of appropriate spatio-temporal intervals. For example, adopting a granularity of a year for a patient's age may result in insufficient support for individual rules while the a priori division of the values into age ranges may result in invalid (or missed) inferences. The problem becomes more severe when the spatial dimension is non-geographic or when cyclic temporal intervals are involved. Other researchers have recognized this problem and solutions to date have included:

- the use of calendric association rules in which various 'calendars' are used to reduce the search space [81, 179]. 'Calendars' in this case refers not only to the many accepted conventions for synchronizing our understanding of an event in absolute time, but also the many conventions relating to relative ages. Although reducing the search space in comparison to a full search, these solutions still suffer from the a priori specification of a set of possible spatial and temporal patterns.
- the use of hierarchical data mining. This allows graduated temporal intervals and spatial regions to be accommodated with the more general being tested when the more specific do not reach the required support thresholds [138, 191]. However, the intervals used at each higher level must subsume those at the level below. Using multiple hierarchies can ameliorate this although this expands the search space in comparison to

the single hierarchy and most algorithms proposed to date suffer from the a priori specification of the spatial and temporal patterns.

- a combination of association rule and clustering algorithms. In this approach, association rules are clustered to determine the appropriate intervals. The approach outlined by Lent et al. [124] creates a 2-D matrix in which the cells are clustered then appropriate minimal description boundaries for the coordinates can be determined.

Secondly, clustering has been applied in spatio-temporal data mining, such as the use of OPTICS for trajectory clustering, aggregation or generalization [150, 181, 9, 8] and the use of DBSCAN for moving data clustering [106, 102, 213]. However, mechanisms for detecting and characterizing changes to cluster boundaries have not received much attention. For example, the spread of many infections, such as HIV, is known to follow distinct spatio-temporal patterns as does the incidence of some pandemic conditions, such as schizophrenia. However, the automated mining of rules that might accommodate such patterns has not been widely investigated.

A third problem is the common, but largely unaddressed issue of detecting statistically-significant anomalies from a series of multiple, large and semantically complex snapshots or single location datasets (such as those that could be collected by an organization as part of routine archival operations or statutory reporting). Efficiently solving this problem would enable a more rapid development of knowledge discovery systems capable of uncovering hidden spatio-temporal trends and correlations which might, in some cases, act as an alerting mechanism. For example, spatio-temporal outlier detection techniques [30, 53] have been proposed

to find spatial outliers over several time periods. Mooney and Roddick [148] tackled this problem by running an association mining algorithm over sets of rules, themselves generated from association rule algorithms.

6.1.3 Motivation

Clearly, anomalies in a single data item can be found using standard statistical techniques. The work presented in this thesis is primarily concerned with whether anomalous transaction data can be detected through an inspection of association rules generated from that data with the following considerations.

First, as discussed in Chapter 2, the primary or raw data might not be always available. For example, organizations (and governments) are willing to provide association rules but unwilling to provide access to source data. Thus in some cases only the rules generated from the source data can be operated upon in research scenarios [183].

Second, the semantics of *second phase mining* are subtly different and, in some cases, lead to more useful information. For example where a (*zero order*) association rule might state that there was a correlation between two (sets of) items, a *higher order* rule might indicate that the strength of the associations was influenced by the presence of a third item. This third item might be deemed to be a *catalyst*. Therefore, the technique to detect anomalies hidden in a set of rules may provide a view of anomalies that is arguably closer to that sought by information analysts.

6.2 Anomaly Detection in Longitudinal Association Rules

Sets of transaction data mined over time are likely to generate rules with the same rule body and thus the form of association rule ($X => Y$) is qualified by the time. In the definitions in this chapter, the syntactic form of a *longitudinal* rule is abbreviated to R_i^τ where R_i is the rule body and τ is the time stamp.

Definition 6.1 (Longitudinal Association Rule Instance) Any given R_i^τ with instantiated $\sigma(R_i)$, $\gamma(R_i)$ and τ values is termed an instance of R_i. For brevity, a specific instance of a rule R_i at time τ is denoted R_i^τ. The support and confidence of R_i^τ are denoted σ_i^τ and γ_i^τ respectively.

Definition 6.2 (Anomalous Rule) Given a set of rules R holding n different instances of R_i, $R = \{R_i^1, \ldots, R_i^n\}$, where $n > 1$, for an instance of $R_i^\tau \in R$, if a rule quality metric such as $\sigma(R_i^\tau)$ and $\gamma(R_i^\tau)$ is significantly different from other instances in R, then R_i^τ is termed anomalous.

Based on the above definitions, anomaly detection in association rules can be stated as the process of identifying those association rule(s) which have significantly different support or confidence values among a large enough number of instances of the same association rule. The main process is categorized into three closely related parts: association rule generation, *CS-set* generation and anomaly detection, as shown in Figure 6.1.

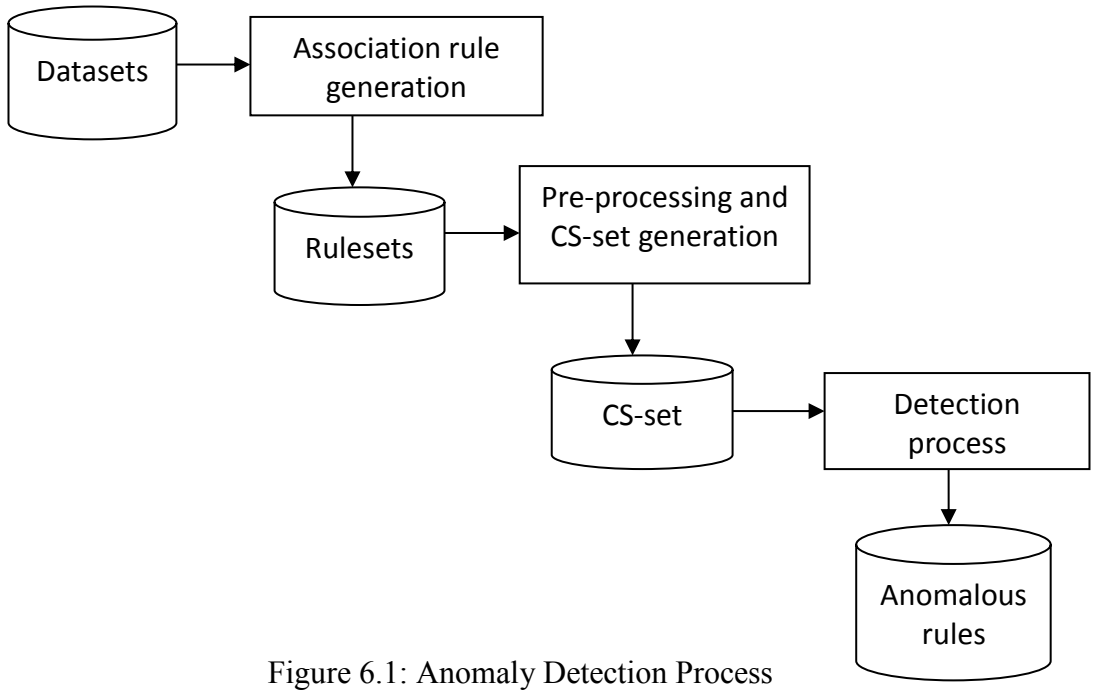

Figure 6.1: Anomaly Detection Process

6.2.1 Longitudinal Association Rule Generation

This part of the process generates association rules from a large amount of input data. Because association mining techniques are relatively mature, there are many widely used algorithms and techniques which can be chosen. The choice of which algorithm is used is not of concern in this work (FP-growth [89, 88] is used herein). Longitudinal sets of rules are commonly generated from a concatenation of multiple individual association rule mining invocations.

6.2.2 Generation of the CS-set

A typical association rules generation run may result in thousands of rules. Moreover, a longitudinal set of rules will typically be two or more orders of magnitude larger. To organize the input rules, create a condensed-sequential set or *CS-set* which brings together the instances of a rule in a form more easily processed by the (potentially third-party) detection algorithms.

For p rules ranging (sparsely) over n time points, the format of the *CS-set* is as shown below:

$$R_1(< \sigma_1, \gamma_1, \tau_1 >, \dots, < \sigma_n, \gamma_n, \tau_n >);$$

...

$$R_p(< \sigma_1, \gamma_1, \tau_1 >, \dots, < \sigma_n, \gamma_n, \tau_n >);$$

Entries are sorted by time within the rule body. This step can also accommodate a pre-processing filter and there is scope for further rule quality metrics to be added.

6.2.3 Detection Process

Using the *CS-set*, the task of detecting anomalies among association rules can be simplified as the detection of anomalous support or confidence values of each association rule R_i in the *CS-set*. This is done by subjecting the rules in the *CS-set* to a series of anomaly detection algorithms (see Section 6.3) which indicate whether the instance is anomalous and if so, a measure of the anomaly's significance. The main detection process can be summarized as shown in Algorithm 6.1.

Algorithm 6.1: Overarching Detection Process

```
1: precondition: CS-set has been generated,
2:                Anomaly thresholds have been defined
3: input: All rules R in the CS-set
4: for all R_i, i = (1 ... n) do
5:     Mark R_i as non-anomalous
6:     for each R_i, τ = (start-time . . . end-time) do
7:         for each algorithm A_i in registry do
8:             Invoke A_i over R_i
9:             if anomalous then
10:                Flag R_i as anomalous at time τ with returned significance θ
11:            end if
12:        end for
13:    end for
14: end for
15:Invoke visualization listing top anomalies
```

6.3 Detection Algorithms

The overarching process described thus far now requires one or more algorithms to detect the anomaly. This chapter presents two algorithms for anomaly detection: *TARMA-a* and *TARMA-b*[5].

The fundamentals for these two algorithms were derived from the Chebyshev theorem that almost all the observations in a dataset will have *z-scores* less than 3. The formula for *z-score* calculation is $z = \frac{x_i - \mu}{sd}$ where μ and sd are the mean and standard deviation of $x_i, (i = 1, \dots, n)$. If $|z_i| \geq 3$, x_i is considered an anomaly. The differences between the two algorithms are:

- For *TARMA-a*, the *z-score* is directly calculated from confidence and support values. It has limited application scope as it can only deal with univariate data. If the variance of the data is unpredictable, the detection accuracy is low.
- For *TARMA-b*, the *z-score* is used to evaluate the expected number of neighbours of each rule instance. It is more robust than *TARMA-a* when dealing with large volumes of arbitrarily varying data.

6.3.1 The *TARMA-a* Algorithm

Based on Definition 6.2, the process of detecting anomalous rules is to identify a significant difference in a rule's confidence or support value with respect to the other time values. *Z-scores* are a good statistical measure of difference amongst large amounts of data. Taking Chebyshev's theorem as the basis the process shown

[5] TARMA - Temporal Association Rule Mining of Anomalies.

in Algorithm 6.2 is applicable.

Algorithm 6.2: *TARMA-a* Algorithm

1: **precondition**: *CS-set* has been generated
2: **input**: All rules R in the *CS-set*
3: **for all** R_i, $i = (1 \ldots n)$ **do**
4: Compute mean support $\mu = \frac{\sigma_1 + \cdots + \sigma_n}{n}$
5: Compute standard deviation $sd = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(s_i - \mu)^2}$
6: **for each** R_i^τ, τ = (start-time . . . end-time) **do**
7: Computer $z_i^\tau = \frac{\sigma_i^\tau - \mu}{sd}$
8: **if**$|z_i^\tau| \geq 3$ **then**
9: Flag R_i^τ as anomalous
10: **end if**
11: **end for**
12: **if** R_i is anomalous **then**
13: Return $\max(z_i^\tau)$ as significance
14: **end if**
15: **end for**

The computational complexity is $O(n)$, indicating that *TARMA-a* is a fast algorithm. Its main failing is that the execution accuracy is heavily reliant on the distribution of the data. It handles univariate data well but its performance is poor when detecting anomalies among highly variate data. Furthermore, it is not a good solution if we wish to consider more advanced temporal aspects.

6.3.2 The *TARMA-b* Algorithm

To overcome the weakness of *TARMA-a*, another more robust algorithm *TARMA-b* was developed, which employs density-based outlier detection techniques. While *TARMA-b* has been specifically designed to detect anomalies in longitudinal association rules, it also works well with rules without such features. *TARMA-b* has been developed based on the idea of density-based outlier detection proposed by Breunig et al. [35], which relies on the local outlier factor (LOF) of each object,

calculated from the local density of its neighbourhood. The neighbourhood is defined by the number of near neighbours. The work presented here takes the essence of this technique and makes some improvements by introducing three new concepts: rX, rY and rXY-neighbourhood.

Definition 6.3 (rX) For a given rule R_i, we have a *CS-set* entry $R_i(< \sigma_1, \gamma_1, \tau_1 >, \ldots, < \sigma_n, \gamma_n, \tau_n >)$ sorted by τ. *rX* is defined as a time span of variable length between τ_1 and τ_n, where $rX \geq 0$ and $rX \leq \tau_n - \tau_1$.

Definition 6.4 (rY) From the *CS-set* calculate the minimum and maximum span of support and confidence σ_{min}, σ_{max}, γ_{min} and γ_{max} respectively. rY is defined as a variable threshold between σ_{min} and σ_{max} or γ_{min} and γ_{max}, where $rY \geq 0$ and either $rY \leq \sigma_{max} - \sigma_{min}$ or $rY \leq \gamma_{max} - \gamma_{min}$.

Definition 6.5 (rXY-neighbourhood) For a given quality metric q (where q is σ, γ or some other metric) for a given rule R_i, for any point $P_n < q_n, \tau_n >$ taken from R_i's CS-set, if there is a point $P'_n < q'_n, \tau'_n >$ that exists such that $|\tau'_n - \tau_n| \leq rX \wedge |q'_n - q_n| \leq rY$, then P'_n is said to be in the rXY-neighbourhood of P_n. The number of neighbours of P_n is represented as $N(P_n, rX, rY)$.

Based on the evaluation of the distribution of data using statistical methods (such as *z-scores*) we can label anomalous points as those with an unusual number of neighbours in their rXY-neighbourhood. This can be done by flagging points that either:

- have fewer than some specified *minpts* number of neighbours,
- have fewer than some number of local neighbours as calculated from the global density for the rule (used in *TARMA-b*),

- have a significant deviation from its consecutive neighbour's count of neighbours,
- have a significant deviation from its the count of neighbours in a larger (but not global) neighbourhood (proposed for *TARMA-c*).

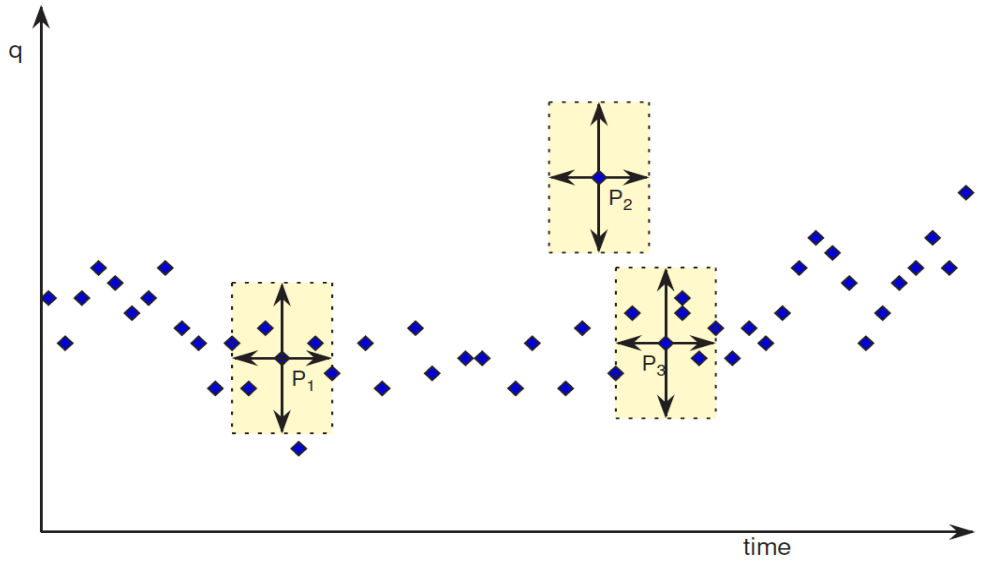

Figure 6.2: An Example of rXY-neighbourhood

These latter two cases take account of datasets that become more or less sparse over time. The use of the concepts rX, rY and rXY-neighbourhood is illustrated in Figure 6.2.

Whichever method is used, rules corresponding to the anomalous points can be identified and returned to the overarching process as potentially suspect. Algorithmically, *TARMA-b* uses *z-scores* to measure the significance of differences to each rule instances neighbour count. That is, the first of the two methods for finding outliers is used. The algorithm is shown in Algorithm 6.3.

Algorithm 6.3: *TARMA-b* Algorithm

precondition: *CS-set* has been generated,
  rX and rY have been defined
for each rule R_i **do**
  for each rule instance R_i^τ **do**
    Calculate rXY-neighbourhood count $N(P_\tau, rX, rY)$
  end for
  Calculate mean μ_i and standard deviation sd_i for R_i
  for each rule instance R_i^τ **do**
    Calculate *z-score* of the number of neighbours for z_i^τ
    if $|z_i^\tau| \geq 3$ **then**
      Flag R_i^τ as anomalous
    end if
  end for
end for

TARMA-b can deal with a variety of arbitrary datasets efficiently. However, the pre-definition of rX and rY is crucial but not trivial. Different rX and rY values may result in different results. A precise definition of rX and rY needs to be based on the complexity of the data and careful study of its distribution, both of which will be discussed in the next section.

6.4 Implemented Prototype and Experiments

In this thesis, a prototype called *TARMAD* has been implemented (screenshots of which are shown in Figure 6.3) and some initial experiments have been completed which show that the concept is sound and feasible in finding outliers.

The prototype design aimed to assess the efficacy of the approach to detecting anomalies in rules produced by *higher order* longitudinal/spatio-temporal association rule mining against traditional statistical data analysis methods. Therefore, *TARMA-a* and *TARMA-b* were designed primarily to be representative algorithms so that the concept could be tested empirically.

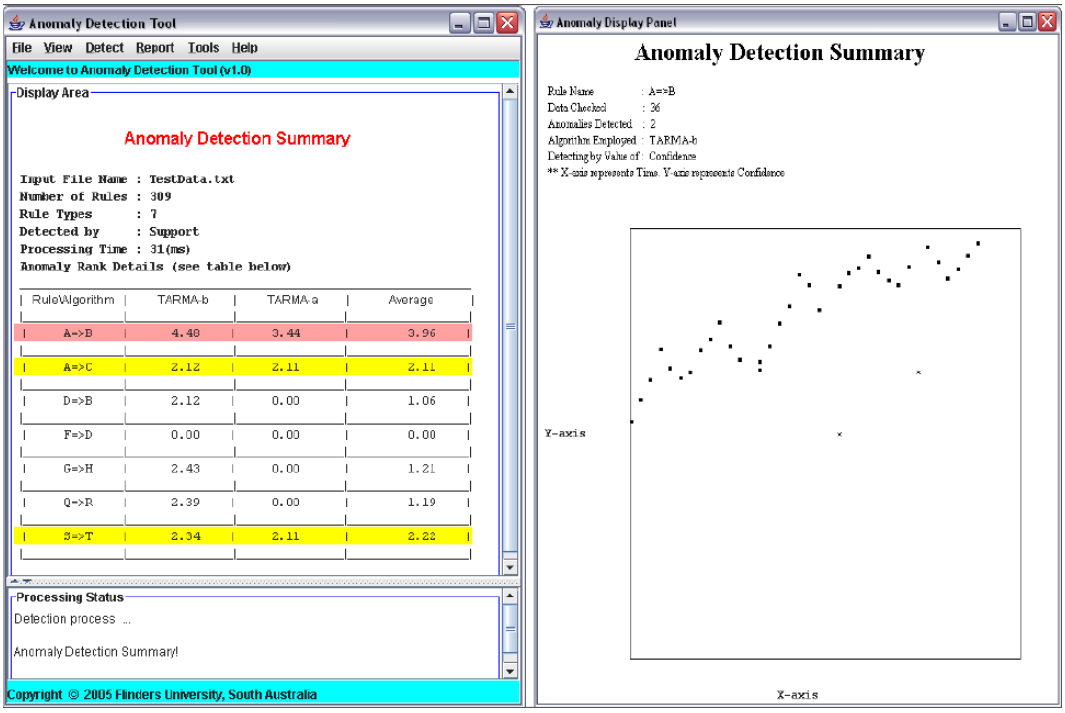

Figure 6.3: Screenshots from *TARMAD* System

The initial prototype system was implemented in Java and the experiments were run on a 2.6GHz PC with 2GB RAM under Window XP. It has the following four main functions:

- Data loading. Data can be loaded with different formats from various sources.
- Anomaly detection. The system currently supports the *TARMA-a* and *TARMA-b* algorithms. Also, based on user-specified requirements, the system can detect anomalies using either confidence or support values.
- Report generation. The system provides a user with detection anomaly details, which includes an anomaly rank to indicate the importance that the detection algorithm believes the anomaly warrants. *TARMA-a* and *TARMA-b* use the *z-score* value to generate the anomaly rank but each algorithm is free to determine its own method of measuring significance.

- Anomaly visualization. The prototype is capable of visualizing the most significant anomalies that have been detected.

In the experiments, both algorithms were tested using both synthetic data (the generator is based on the work reported by Agrawal and Srikant [6] with some modifications to cater for temporal features) and real data (the BMS-WebView-1 and BMS-WebView-2 datasets as used in the KDDCUP in 2000 [113]).

6.4.1 Synthetic Longitudinal Data

A synthetic data generator was built to produce large amounts of longitudinal data which mimic the transactions in a retailing environment. Table 6.1 shows the parameters for the data generation, along with their default values and the range of values on which experiments were conducted. Table 6.2 shows the details of synthetic data that was generated for experiments.

The synthetic data generator has three main steps:

- Step 1: Generate $|T|$ transactions.
- Step 2: Create a time domain $|D|$ which holds n time intervals (Tvl), $|D| = Tvl_1, Tvl_2, \dots, Tvl_n$. $|TF|$ (*Temporizing Factor*) is defined as the number of elements which are randomly chosen from |D|. The mean of transactions during $|TF|$ time intervals is calculated as $\overline{N} = \frac{|T|}{|TF|}$.
- Step 3: the number of transactions to be assigned with Tvl_i is determined from a Poisson distribution with mean equal to $\overline{N}$. A time interval Tvl is then assigned to those transactions. The process repeats until all transactions have been assigned to a time interval.

Table 6.1: Synthetic Data Parameters

Name	Description	Default Value	Range of Values
|I|	Number of Items	10	10-100
|T|	Number of Transactions	5,000	5k-200k
|P|	Number of Patterns	50	50-200
TS	Average Size of Transaction	5	5-10
PS	Average Size of Pattern	5	5-10

Table 6.2: Synthetic Data

Data	|I|	|TS|	|T|	|TF|
I10.TS5.T20.TF40	10	5	20k	40
I50.TS10.T45.TF30	50	10	45k	30
I50.TS15.T100.TF50	50	15	100k	50
I100.TS10.T100.TF30	100	10	100k	30
I100.TS20.T200.TF50	100	20	200k	50

Table 6.3: Real Data

Data	BMS-WebView-1	BMS-WebView-2
Number of Transaction	59,602	77,512
Distinct Items	497	3,340
Maximum Transaction Size	267	161
Average Transaction Size	2.5	5.0

6.4.2 Real Data

BMS-WebView-1 and BMS-WebView-2 contain several months' worth of click stream data from two e-commerce web sites. Each transaction in the two datasets is a web session consisting of all the product detail pages viewed in that session. That is, each product detail view is an item. The details of the two datasets are shown in Table 6.3.

Taking the two datasets, a set of experiments were conducted aiming to discover whether there are any anomalies amongst the associations between products viewed by visitors to the web site. Since there are no timestamps for each unit of click stream data, they are temporized by following steps 2 and 3 in the previous section.

6.4.3 Longitudinal Association Rule Generation

After the temporization of test datasets, the work to generate longitudinal association rules is straightforward. In the work presented here, the ideas from Rainsford and Roddick [175] were employed. Firstly, frequent items are generated from transactions which occurred during the same time interval (Tvl_i) using FP-growth. Longitudinal association rules are then generated by adding temporal semantics (time interval Tvl_i) to each frequent itemset which satisfied the minimum support and confidence value.

Since there is no guarantee that rule R_i will be found at different times and it will be meaningless to detect the significant change of a rule R_i if it has no or only few rule instances in that time domain, the minimum number of rule instances (denoted as min_$N(R_i)$) is defined as a threshold that one rule R_i should satisfy. Those rules that have instances less than min_N are pruned out.

6.4.4 Experimental Results and Evaluation

Experiments over both synthetic and real-world data show that the concept is sound and that outliers in the behaviour of data can be found even if the incidence of the items in the transaction do not change significantly. The results are shown in Table 6.4. The experimental performance showed that (as is the case with many data mining tools) I/O dominates the calculation and the empirical results show a linear correlation with dataset size (as shown in Figure 6.4). Moreover, even including all I/O requirements, *TARMA-b* was able to analyse 100k transactions in 23 seconds.

Table 6.4: Test Results

Dataset	TF	Number of Rules	*TARMA-a*		*TARMA-b*		Deviation Rate
			Anomalies Found	Anomaly Rank	Anomalies Found	Anomaly Rank	
I10.TS5.T20.TF40	40	12,255	348	4.23	335	2.68	0.04%
I50.TS10.T45.TF30	30	12,726	787	3.39	716	2.33	0.01%
I50.TS15.T100.TF50	50	13,204	259	5.15	257	5.10	0.01%
I100.TS10.T100.TF30	30	2,224	59	4.36	52	2.31	0.12%
I100.TS20.T200.TF50	50	11,997	239	5.42	234	5.49	0.03%
BMS-WebView-1	50	492	7	3.48	7	3.26	0.00%
BMS-WebView-1	100	1,221	24	3.00	24	3.26	0.00%
BMS-WebView-1	120	1,676	35	2.92	33	3.26	0.06%
BMS-WebView-2	90	4,163	71	2.86	71	3.26	0.00%
BMS-WebView-2	120	7,397	108	2.60	119	3.26	0.01%

		Count of Transactions('000s)										
	Routine	5	10	15	20	25	35	45	50	60	75	100
Time (secs)	*TARMA-a*	3.06	5.37	9.76	11.43	12.5	13.79	15.34	18.06	19.46	20.9	21.85
	TARMA-b	3.09	5.5	10.04	12.03	12.65	14.01	16.25	18.39	19.48	21.08	22.93

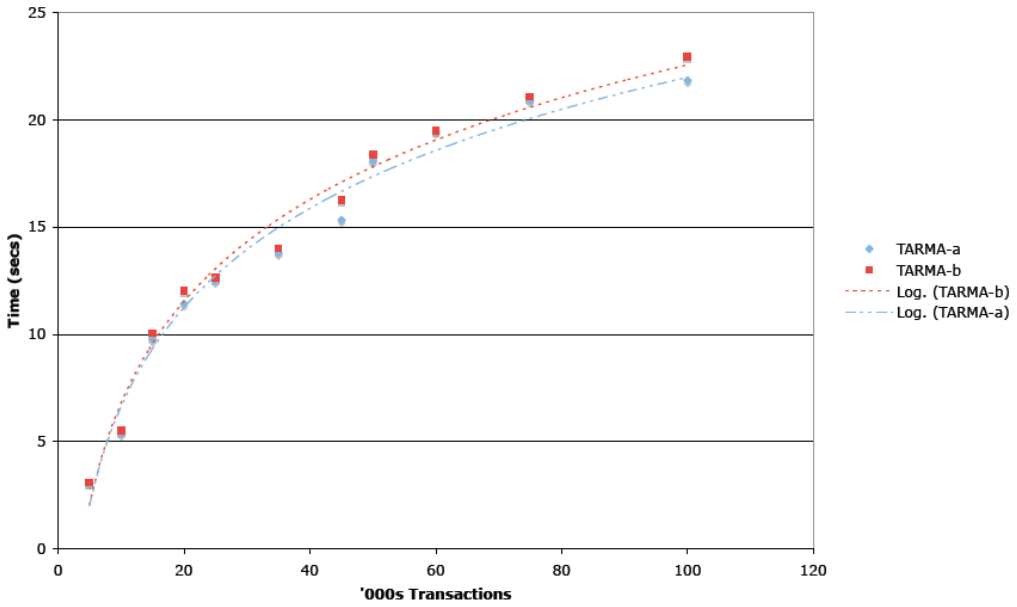

Figure 6.4: Performance – Time vs #Transactions

For the tests, the minimum support, minimum confidence and min_N are defined as 0.20, 0.80 and 10 respectively with synthetic data and 0.01, 0.20, 10 respectively with real datasets. Only the value of support has been taken into account in the process of detecting anomalous rules. To indicate the importance that the detection

algorithm indicates the anomaly warrants, the concept of an anomaly rank is introduced. *TARMA-a* and *TARMA-b* use a *z-score* value to generate the anomaly rank.

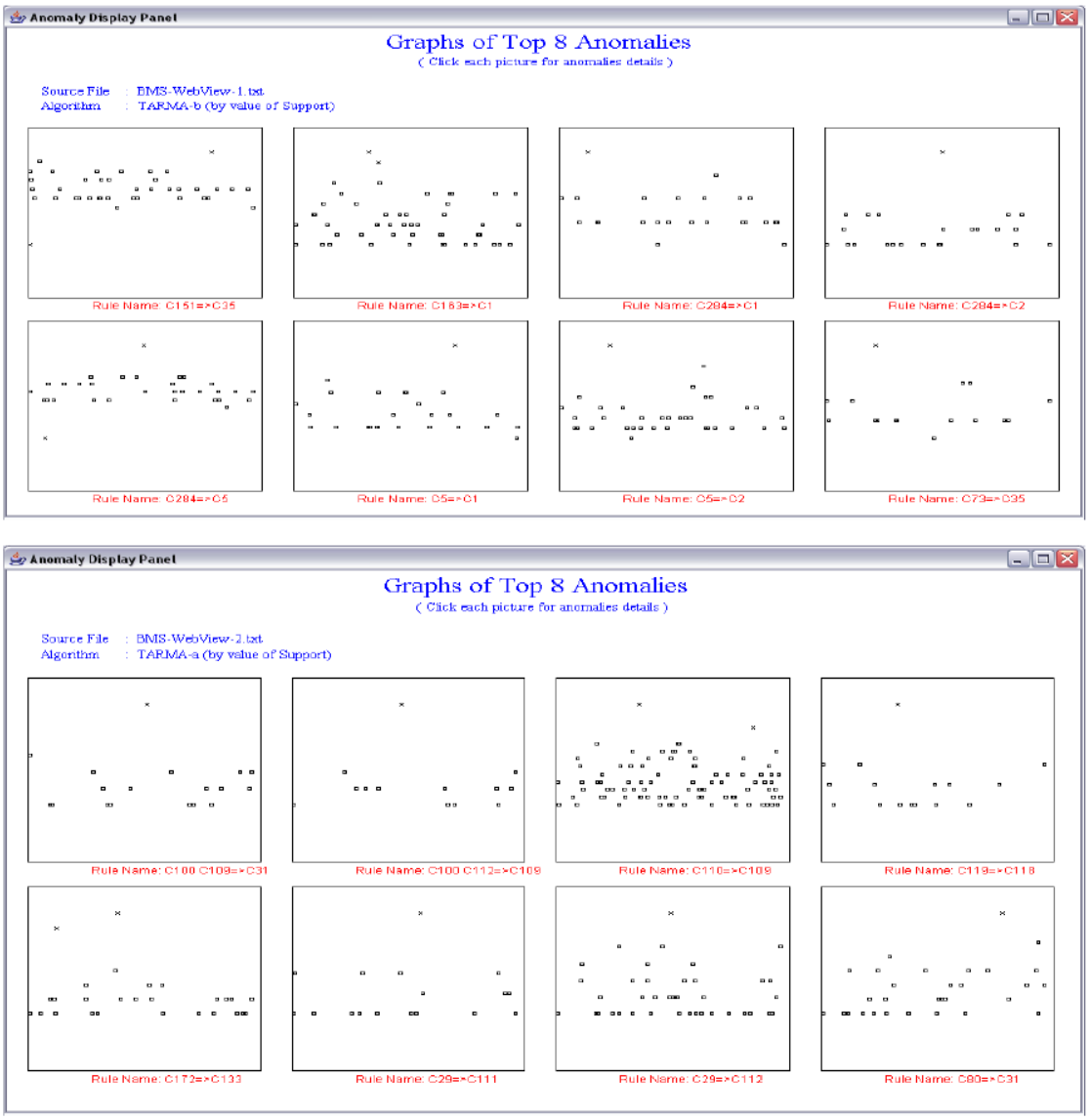

Figure 6.5: Screenshots for Top N Anomalies in Real Data

Both *TARMA-a* and *TARMA-b* have successfully detected anomalies among all test datasets with the size ranging from 20K to 200K and the count of association rules from 500 to 13,200. Although not all anomalies were examined to evaluate the

detection accuracy, the approach presented here has demonstrated its capability to detect anomalies in complex datasets after examinations of the top N anomalies ($N = 10\%$ of the whole amount of anomalies found in the test). Some screen shots of the top 8 anomalies among two real datasets are shown in Figure 6.5. The viewed page is denoted with the character C plus a number.

6.4.4.1 Comparison between *TARMA-a* and *TARMA-b*

Experiment results show that the detection results are similar for the two algorithms. The average deviation rate (the percentage of anomalies found by one algorithm but ignored by another), was as low as 0.05% in synthetic datasets and 0.03% in real datasets.

Although *TARMA-a* has a higher execution speed than *TARMA-b*, *TARMA-b* is more robust than *TARMA-a* in dealing with complex datasets. Further tests were conducted to compare their capacity to detect anomalies hidden among large amounts of data with different densities. Firstly, some association rules which have predefined distribution were generated and then some anomalous points were added into them.

Figure 6.6 shows some of these test data. When the two algorithms were applied with the test data, only *TARMA-b* successfully detected all points (P_1−P_4) as anomalies. The experiments have shown that *TARMA-a* works well with simple data coming from a univariate Gaussian distribution but performs poorly with multi-variate data, that is, data from heavy-tailed distributions.

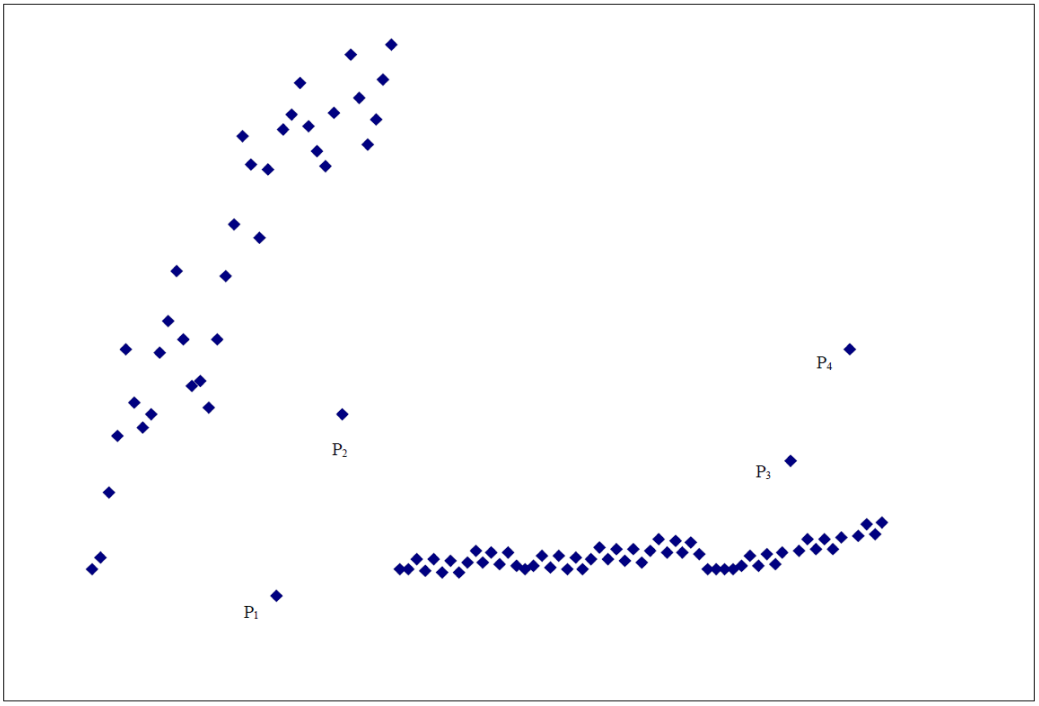

Figure 6.6: Complex Data

6.4.4.2 Effect of rX and rY on Anomaly Detection Results

TARMA-b calculates a *z-score* from rXY-neighbourhood and therefore has great advantages over *TARMA-a*. However, the predefinition of rX and rY is crucial but not trivial. In the experiments, rX is defined as $rX = K_x \times sd_x$, where $K_x \geq 0$ and sd_x is the standard deviation from the sorted time set $T = 1,2, \ldots n$. Similarly, $rY = K_y \times sd_y$, where $K_y \geq 0$ and sd_y is the standard deviation of the selected quality metric.

Testing *TARMA-b* with different values of K_x resulted in the outcome shown in Table 6.5. It is clear that when the rX value is not appropriately defined, the detection accuracy becomes low as the rXY-neighbourhood is sensitive to the value of rX and rY. That is, if the window is too big or too small, the evaluation of the change of number of rXY-neighbourhood using a *z-score* becomes less meaningful.

A more robust algorithm currently is being developed, which has the capability to determine the most suitable rX and rY value automatically, which may lead to the next anomaly detection algorithm, *TARMA-c* (as shown in Algorithm 6.4), to improve detection efficiency.

Table 6.5: Test Results with Different rX

Rule	$rX = 4 \times sd_x$		$rX = 2 \times sd_x$		$rX = 0.5 \times sd_x$		$rX = 0.25 \times sd_x$	
	Average rXY-n'hood count	Anomalies found	Average rXY-n'hood count	Anomalies found	Average rXY-n'hood count	Anomalies found	Average rXY-n'hood count	Anomalies found
A	11	1	9	1	4	1	3	1
B	13	0	13	0	7	3	3	3
C	10	0	9	1	4	4	2	0
D	5	0	5	0	3	0	2	0
E	29	0	24	0	10	1	3	0
F	10	0	10	0	7	2	4	0
G	13	0	12	0	6	2	4	2

Algorithm 6.4: *TARMA-c* Algorithm

1: **precondition**: *CS-set* has been generated
2: rX and rY have been defined
3: **input**: All rules R in the *CS-set*
4: **for each** rule R_i **do**
5: **for each** rule instance R_i^τ do
6: Calculate rXY-neighbourhood count and save as $N(R_i^\tau)$
7: Calculate rXY-window count and save as $W(R_i^\tau)$
8: **end for**
9: Calculate mean μ_i and standard deviation sd_i for $R_i$within its rXY-window
10: **for each** rule instance R_i^τ do
11: Calculate Z_p = *z-score* for the number of neighbours
12: Calculate Z_w = average *z-score* for the window
13: **if** $|Z_p| \leq Z_w$ then
14: Flag R_i as anomalous
15: **end if**
16: **end for**
17: **end for**

The *TARMA-c* algorithm could be (and is being) developed to replace the use of the *z-score* of the number of neighbours across the rule with a *z-score* of the number of proximate neighbours. It calculates a running mean and standard deviation within a

larger window (rX.window × rY.window). In this way, some issues, such as the edge conditions (i.e. problems encountered with the first and last data points) and datasets that vary in the number of collected data points will be catered for more naturally.

6.5 Summary

The work presented in this chapter sought to validate the idea that the inspection of rules as opposed to data could be useful as a tractable method of finding outliers.

Having briefly reviewed related works of anomaly detection and longitudinal and spatio-temporal knowledge discovery, two algorithms, *TARMA-a* and *TARMA-b*, have been proposed for anomaly detection. Both of these algorithms are based on the Chebyshev theorem that almost all the observations in a dataset will have *z-scores* less than 3.

TARMA-a directly calculates the *z-scores* from confidence and support values and can only deal with univariate data. In contrast, *TARMA-b* uses the *z-score* to evaluate the expected number of neighbours of each rule instance. Although *TARMA-a* has a higher execution speed than *TARMA-b*, *TARMA-b* is more robust than *TARMA-a* in dealing with more complex datasets.

While further work needs to be undertaken (including the development of better detection algorithms such as the planned *TARMA-c* algorithm), the work to date has shown the soundness and feasibility of the proposed approach.

Chapter 7

Conclusion and Future Research

Our world is now in an information era. The explosive growth in data and databases generates the need for new techniques and tools that can intelligently and automatically transform that data into useful information and knowledge. Data mining is one technology designed to meet this challenge.

As discussed in Chapter 2, with a few notable exceptions, data mining research has largely focused on the extraction of knowledge directly from the source data. However, in many cases, such mining routines are beginning to encounter problems as the primary or raw data might not always be available. For instance, in some applications, stream data are only available for a short time while some cooperating institutions that are interested in sharing knowledge may not be willing to disclose their primary data. Besides the availability issue of primary data, many data mining routines are becoming heavily I/O bound due to the fact that the volume of data requiring analysis grows disproportionately with the comparatively slower

improvements in I/O channel speeds which limit many of the benefits of the technology. One approach to tackling those issues is to mine over patterns/models derived from one or more large and /or complex datasets, which is generally termed higher order mining (HOM) [183]. This thesis employed the idea of HOM and addressed two important but unanswered issues:

- the discovery of patterns in association rules which represent the *higher order* knowledge sought by users, and
- the discovery of anomalies in association rules that produced by *higher order* longitudinal/spatio-temporal association rule mining.

7.1 Contributions

7.1.1 Discovering Patterns in Association Rules

In order to create useable systems, problems such as the generation and interpretation of interestingness for discovered rules are important considerations and need to be resolved. Unfortunately, since rules are commonly supplied in a low, instance-level format, the rules generated from many algorithms do not correspond to the types of knowledge often being sought by the user. Rather *higher order* knowledge is required which necessitates the construction of complex patterns of data and rules. This thesis developed mechanisms to cater for the tasks of defining and searching such *higher order* patterns based on a user's definition of interesting. In particular, the following contributions were realised.

First, the thesis proposed formal definitions of patterns in rules, or ruleset patterns, which reflect the types of knowledge users are interested in. Based on the

definition of ruleset patterns, a proof-of-concept system, Horace, was presented for efficient ruleset pattern discovery. Horace consists of three key components: the FP-tree or other prefix trees, a pattern library with its associated ruleset pattern language and a set of searching algorithms. Since FP-tree or other prefix trees contain the complete set of information held in a database relevant to frequent pattern mining, Horace employs a tree-based approach to searching for ruleset patterns in the trees instead of rulesets, which has been proved through a prototype system to be possible and more efficient.

In addition, a novel ruleset pattern tree (RP-tree) was introduced to represent ruleset patterns. Given the structures of the FP-tree and RP-tree, two algorithms (*SRPFP-a* and *SRPFP-b*) were developed to search the FP-tree for matches of the RP-tree. *SRPFP-a* substitutes RP-tree nodes with items from the header table of the FP-tree, which only requires one scan of the parent rule branch of the RP-tree thus potentially reducing the processing capacity and time required. However, its computation time was found to be heavily affected by the number of frequent items of the FP-tree and distinct nodes in the RP-tree. To improve the searching efficiency, *SRPFP-b* was further proposed to break the searching task into two steps: the generation of k_p-itemsets and the construction of the RP-tree instances. A variant of FP-growth [88], RP-growth, was introduced to facilitate the fast retrieval of the k_p-itemsets. A complete prototype was built and a comprehensive set of tests have demonstrated the feasibility of both algorithms, with *SRPFP-b* demonstrating greater efficiency when dealing with more complex and large FP-trees and RP-trees.

Finally, a ruleset pattern language (RPL) was developed, which consists of a ruleset pattern definition language (RPDL) and a ruleset pattern query language (RPQL). RPL enables users to create, alter and retrieve patterns from a ruleset pattern library. Also, indexing techniques were employed to further improve the efficiency of RPQL query evaluations. A set of experiments were conducted and reported in this thesis and have demonstrated the efficiency of RPQL.

7.1.2 Discovering Anomalies in Longitudinal Association Rules

The detection of unusual or anomalous data is an important function in automated data analysis or data mining. However, the diversity of anomaly detection algorithms shows that it is often difficult to determine which algorithms might detect anomalies given any random dataset. In this thesis, the idea that the inspection of rules that are produced by *higher order* longitudinal/spatio-temporal association rule mining as opposed to data could be useful as a tractable method of finding outliers was validated. Furthermore it was demonstrated that such a technique may provide a view of anomalies that is arguably closer to that sought by information analysts.

This thesis presents a formal definition of anomalies in longitudinal association rules. Based on the definition, the anomaly detection process in association rules is stated as the process of identifying those association rule(s) which have significantly different support or confidence values among a large enough number of instances of the same association rule. Also, to facilitate the anomaly detection algorithms, the thesis proposed condensed-sequential set or *CS-set* which is extracted from varying longitudinal association rule formats and organized in

sequential order in terms of time of occurrence.

Furthermore, two algorithms were proposed (*TARMA-a* and *TARMA-b*) for anomaly detection. The fundamentals of these two algorithms were derived from the Chebyshev theorem that almost all the observations in a dataset will have *z-scores* less than 3. In *TARMA-a*, the *z-score* is directly calculated from confidence and support values. *TARMA-a* is a fast algorithm but has limited application scope as it can only deal with univariate data. To overcome the deficiencies of *TARMA-a*, *TARMA-b* detects anomalies by evaluating the expected number of proximate neighbors of each rule instance. It is more robust than *TARMA-a* when dealing with large volumes of arbitrarily varying data.

Finally, a set of experiments is reported based on both synthetic data and real world data. Test results have demonstrated that the proposed approach is sound and feasible in finding outliers.

7.2 Future Research

The work presented in this thesis points to several directions for future research.

One of the research directions to be undertaken includes handling the conjunction of various ruleset patterns, that is, *hybrid* ruleset patterns. For example, given a competitor pattern, containing the following three rules: $\{x\} => \{z\}$, $\{y\} => \{z\}$, $\{x, y\} => \{z\}$, and a catalyst pattern containing the following three rules: $\{x\} => \{w\}$, $\{y\} => \{w\}$, $\{x, y\} => \{w\}$, a conjunction of the two patterns might result in a *hybrid* pattern, revealing that with item z items x and y are competitors while with item w, they become catalysts. The challenges of efficiently searching for

matches of *hybrid* ruleset patterns lie in two aspects. First, *hybrid* RP-trees are needed to cater for multiple parent rule branches. Second, more robust search algorithms are required to find matches for such complex *hybrid* RP-trees.

Another research direction is the enhancements of RPL. Future work is planned to add additional features to RPQL, including the UNION operator, the support for GROUP BY, HAVING and ORDER clauses. Also, more efficient query processing methods are under development to cope with more complex RPL queries.

Finally, there is an intention to develop more robust anomaly detection algorithms. As indicated in Chapter 6, a *TARMA-c* algorithm is being developed to replace the use of the *z-score* of the number of neighbours across the rule with a *z-score* of the number of proximate neighbours. This will also address the automatic determination of rX and rY with the static use of the *z-score* of 3.

Appendix

Publications Resulting from This Thesis

The following publications have resulted from material presented within this thesis. Publication 1 relates to material presented in Chapter 6 while publication 2 contains early work of Chapter 3 and 4. Publication 3 contains most of the material presented in Chapter 5.

- P. LIANG and J. F. RODDICK, *Detecting anomalous longitudinal associations through higher order mining*, in K.-L. Ong, W. Li and J. Gao, eds., *2nd International Workshop on Integrating Artificial Intelligence and Data Mining(AIDM 2007)*, Australian Computer Society, Gold Coast, Queensland, 2007, pp. 19-27.
- P. LIANG, J. F. RODDICK and D. DE VRIES, *Searching frequent pattern and prefix trees for higher order rules*, in P. Christen, P. Kennedy, L. Liu,

K.-L. Ong, A. Stranieri and Y. Zhao, eds., *11th Australian Data Mining Conference (AusDM 2013)*, Australian Computer Society, Inc, Canberra, Australia, 2013.

- P. LIANG and J. F. RODDICK, *RPL: A ruleset pattern language*, *International Conference on Artificial Intelligence and Industrial Application(AIIA2014)*, WIT Press, Hong Kong, 2014.

Bibliography

[1] H. ABU-MANSOUR, W. E. HADI, T. MCCLUSKEY and F. THABTAH, *Associative text categorisation rules pruning method*, in RafalRzepka, ed., *Linguistic and Cognitive Approaches to Dialog Agents Symposium (LaCATODA'10)*, UK, 2010, pp. 39-44.

[2] C. C. AGGARWAL, *On change diagnosis in evolving data streams*, IEEE Transactions on Knowledge and Data Engineering, 17 (2005), pp. 587-600.

[3] R. AGRAWAL, T. IMIELIŃSKI and A. SWAMI, *Mining association rules between sets of items in large databases*, ACM SIGMOD Record, 22 (1993), pp. 207-216.

[4] R. AGRAWAL and G. PSAILA, *Active Data Mining*, *KDD'95*, 1995, pp. 3-8.

[5] R. AGRAWAL, G. PSAILA, E. L. WIMMERS and M. ZAIT, *Querying shape of histories, 21st VLDB Conference(VLDB'95)*, 1995, pp. 502-514.

[6] R. AGRAWAL and R. SRIKANT, *Fast algorithms for mining association rules*, in J. Bocca, M. Jarke and C. Zaniolo, eds., *20th International Conference on Very Large Data Bases(VLDB'94)*, Morgan Kaufmann, Santiago, Chile, 1994, pp. 487-499.

[7] J. M. ALE and G. H. ROSSI, *An approach to discovering temporal association rules, the 2000 ACM symposium on Applied Computing*, ACM, 2000, pp. 294-300.

[8] G. ANDRIENKO and N. ANDRIENKO, *Interactive cluster analysis of diverse types of spatiotemporal data*, ACM SIGKDD Explorations Newsletter, 11 (2010), pp. 19-28.

[9] G. ANDRIENKO and N. ANDRIENKO, *Spatio-temporal aggregation for visual analysis of movements*, *IEEE Symposium on Visual Analytics Science and Technology (VAST'08)*, IEEE, 2008, pp. 51-58.

[10] F. ANGIULLI and C. PIZZUTI, *Fast outlier detection in high dimensional spaces*, *Principles of Data Mining and Knowledge Discovery*, Springer, 2002, pp. 15-27.

[11] M.-L. ANTONIE and O. R. ZA ÏANE, *An associative classifier based on positive and negative rules*, *9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, ACM, 2004, pp. 64-69.

[12] A. ARNING, R. AGRAWAL and P. RAGHAVAN, *A linear method for deviation detection in large databases*, *KDD'96*, 1996, pp. 164-169.

[13] W. H. AU and K. C. C. CHAN, *Mining changes in association rules: a fuzzy approach*, Fuzzy Sets And Systems, 149 (2005), pp. 87-104.

[14] Y. AUMANN and Y. LINDELL, *A statistical theory for quantitative association rules*, Journal of Intelligent Information Systems, 20 (2003), pp. 255-283.

[15] N. F. AYAN, A. U. TANSEL and E. ARKUN, *An efficient algorithm to update large itemsets with early pruning*, *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 1999, pp. 287-291.

[16] B. BAESENS, S. VIAENE and J. VANTHIENEN, *Post-processing of association rules*, *DTEW Research Report 0020*, 2000, pp. 1-18.

[17] E. BARALIS, S. CHIUSANO and P. GARZA, *On support thresholds in associative classification*, *ACM Symposium on Applied Computing (SAC'04)*, ACM, 2004, pp. 553-558.

[18] E. BARALIS and P. GARZA, *A lazy approach to pruning classification rules*, *IEEE International Conference on Data Mining (ICDM02)*, IEEE, 2002, pp. 35-42.

[19] D. BARBARÁ, J. COUTO, S. JAJODIA and N. WU, *ADAM: A testbed for exploring the use of data mining in intrusion detection*, ACM SIGMOD Record, 30 (2001), pp. 15-24.

[20] D. BARBARÁ, Y. LI, J. COUTO, J.-L. LIN and S. JAJODIA, *Bootstrapping a data mining intrusion detection system*, *ACM Symposium on Applied Computing(SAC'03)*, ACM, 2003, pp. 421-425.

[21] S. BARON and M. SPILIOPOULOU, *Monitoring change in mining results*, in A. Cuzzocrea and U. Dayal, eds., *Data Warehousing and Knowledge Discovery*, Springer, 2001, pp. 51-60.

[22] S. BARON and M. SPILIOPOULOU, *Monitoring the evolution of web usage patterns*, in B. Berendt, A. Hotho, D. Mladenic, M. van Someren, M. Spiliopoulou and G. Stumme, eds., *Web Mining: From Web to Semantic Web*, Springer, 2004, pp. 181-200.

[23] S. BARON and M. SPILIOPOULOU, *Monitoring the results of the KDD process: An overview of pattern evolution*, in J. Meij, ed., *Dealing with the Data Flood: Mining Data, Text and Multimedia*, STT Netherlands Study Center for Technology Trends, The Hague, Netherlands, 2002, pp. 845-863.

[24] S. BARON, M. SPILIOPOULOU and O. GÜNTHER, *Efficient monitoring of patterns in data mining environments*, *7th East-European Conference on Advances in Databases and Information Systems (ADBIS'03)*, Springer, 2003, pp. 253-265.

[25] I. BARTOLINI, P. CIACCIA, I. NTOUTSI, M. PATELLA and Y. THEODORIDIS, *A unified and flexible framework for comparing simple and complex patterns*, *8th European Conference on Principles and Practice of Knowledge Discovery in Databases(PKDD'04)*, Springer, Pisa, Italy, 2004, pp. 496-499.

[26] Y. BASTIDE, N. PASQUIER, R. TAOUIL, G. STUMME and L. LAKHAL, *Mining minimal non-redundant association rules using frequent closed itemsets*, in J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv and P. J. Stuckey, eds., *International Conference on Computational Logic(CL'2000)*, Springer, London, United Kingdom, 2000, pp. 972-986.

[27] Y. BASTIDE, R. TAOUIL, N. PASQUIER, G. STUMME and L. LAKHAL, *Mining frequent patterns with counting inference*, ACM

SIGKDD Explorations Newsletter, 2 (2000), pp. 66-75.

[28] S. D. BAY and M. SCHWABACHER, *Mining distance-based outliers in near linear time with randomization and a simple pruning rule*, *SIGKDD'03*, ACM, Washington, DC, USA, 2003, pp. 29-38.

[29] R. J. BAYARDO JR, R. AGRAWAL and D. GUNOPULOS, *Constraint-based rule mining in large, dense databases*, *the 15th International Conference on Data Engineering*, IEEE, 1999, pp. 188-197.

[30] D. BIRANT and A. KUT, *Spatio-temporal outlier detection in large databases*, *28th International Conference on Information Technology Interfaces*, IEEE, Cavtat, Dubrovnik 2006, pp. 179-184.

[31] F. BODON, *A fast apriori implementation*, in F. B. Goethals and M. J. Zaki, eds., *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, IEEE Press, Melbourne, 2010.

[32] R. J. BOLTON and D. J. HAND, *Unsupervised profiling methods for fraud detection*, *Credit Scoring and Credit Control VII*, Edinburgh, UK, 2001, pp. 235-255.

[33] M. BÖTTCHER, D. NAUCK, D. RUTA and M. SPOTT, *Towards a framework for change detection in datasets*, *26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Springer, 2006, pp. 115-128.

[34] P. S. BRADLEY and U. M. FAYYAD, *Refining initial points for k-means clustering*, in J. Shavlik, ed., *15th International Conference on Machine Learning(ICML'98)*, Morgan Kaufmann, San Francisco, CA, 1998, pp. 91-99.

[35] M. M. BREUNIG, H.-P. KRIEGEL, R. T. NG and J. SANDER, *LOF: Identifying density-based local outliers*, *ACM SIGMOD Record*, ACM, 2000, pp. 93-104.

[36] M. M. BREUNIG, H.-P. KRIEGEL, R. T. NG and J. SANDER, *Optics-of: Identifying local outliers*, *Principles of Data Mining and Knowledge Discovery*, Springer, 1999, pp. 262-270.

[37] T. BRIJS, G. SWINNEN, K. VANHOOF and G. WETS, *Using*

association rules for product assortment decisions: A case study, *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 1999, pp. 254-260.

[38] S. BRIN, R. MOTWANI and C. SILVERSTEIN, *Beyond market baskets: Generalizing association rules to correlations*, ACM SIGMOD Record, 26 (1997), pp. 265-276.

[39] S. BRIN, R. MOTWANI, J. D. ULLMAN and S. TSUR, *Dynamic itemset counting and implication rules for market basket data*, ACM SIGMOD Record, 26 (1997), pp. 255-264.

[40] S. BRIN, R. RASTOGI and K. SHIM, *Mining optimized gain rules for numeric attributes*, *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, California, USA, 1999, pp. 135-144.

[41] M. BRITO, E. CHAVEZ, A. QUIROZ and J. YUKICH, *Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection*, Statistics & Probability Letters, 35 (1997), pp. 33-42.

[42] S. BUDALAKOTI, A. N. SRIVASTAVA, R. AKELLA and E. TURKOV, *Anomaly detection in large sets of high-dimensional symbol sequences*, Technical report (2006), NASA Ames Research Center.

[43] S. BYERS and A. E. RAFTERY, *Nearest-neighbor clutter removal for estimating features in spatial point processes*, Journal of the American Statistical Association, 93 (1998), pp. 577-584.

[44] A. BYKOWSKI and C. RIGOTTI, *A condensed representation to find frequent patterns*, *20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ACM, Santa Barbara, California, USA 2001, pp. 267-273.

[45] T. CALDERS and B. GOETHALS, *Mining all non-derivable frequent itemsets*, *6th European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, 2002, pp. 74-85.

[46] R. A. CARRASCO, M. A. VILA and F. ARAQUE, *dmFSQL: A language for data mining*, *17th International Workshop on Database and Expert*

Systems Applications (DEXA'06). , IEEE, 2006, pp. 440-444.

[47] A. CEGLAR and J. F. RODDICK, *Association Mining*, ACM Computing Surveys, 38 (2006), pp. 1-42.

[48] S. CHAKRABARTI, S. SARAWAGI and B. DOM, *Mining surprising patterns using temporal description length*, in A. Gupta, O. Shmueli and J. Widom, eds., *24th International Conference on Very Large Data Bases (VLDB'98)*, Morgan Kaufmann, New York, NY, USA, 1998, pp. 606-617.

[49] K. C. CHAN and W.-H. AU, *Mining fuzzy association rules*, *6th International Conference on Information and Knowledge Management*, ACM, 1997, pp. 209-215.

[50] V. CHANDOLA, A. BANERJEE and V. KUMAR, *Anomaly detection: A survey*, ACM Computing Surveys (CSUR), 41 (2009), pp. 1-58.

[51] X. CHEN and I. PETROUNIAS, *An integrated query and mining system for temporal association rules*, *2nd International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2000)*, Springer, 2000, pp. 327-336.

[52] X. CHEN and I. PETROUNIAS, *Mining temporal features in association rules*, *Principles of Data Mining and Knowledge Discovery*, Springer, 1999, pp. 295-300.

[53] T. CHENG and Z. LI, *A multiscale approach for spatio-temporal outlier detection*, Transactions in GIS, 10 (2006), pp. 253-263.

[54] D. W.-L. CHEUNG, V. T. NG and B. W. TAM, *Maintenance of discovered knowledge: A case in multi-level association rules*, *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, AAAI Press, 1996, pp. 307-310.

[55] D. W. CHEUNG, J. HAN, V. T. NG and C. WONG, *Maintenance of discovered association rules in large databases: An incremental updating technique*, in S. Su, ed., *12th International Conference on Data Engineering(ICDE'96)*, IEEE Computer Society, New Orleans, Louisiana,USA, 1996, pp. 106-114.

[56] D. W. CHEUNG, S. D. LEE and B. KAO, *A general incremental*

technique for maintaining discovered association rules, *5th International Conference on Database Systems for Advanced Applications(DASFAA'97)*, 1997, pp. 185-194.

[57] A. L.-M. CHIU and A.-C. FU, *Enhancements on local outlier detection*, *7th International Database Engineering and Applications Symposium* IEEE, 2003, pp. 298-307.

[58] F. EDGEWORTH, *XLI. On discordant observations*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 23 (1887), pp. 364-375.

[59] M. EL-HAJJ and O. R. ZA ÏANE, *COFI-tree mining: A new approach to pattern growth with reduced candidacy generation*, *Workshop on Frequent Itemset Mining Implementations (FIMI'03) in conjunction with IEEE-ICDM*, Melbourne 2003.

[60] E. ESKIN, A. ARNOLD, M. PRERAU, L. PORTNOY and S. STOLFO, *A geometric framework for unsupervised anomaly detection*, *Applications of Data Mining in Computer Security*, Springer, 2002, pp. 77-101.

[61] M. ESTER, A. FROMMELT, H.-P. KRIEGEL and J. SANDER, *Spatial data mining: database primitives, algorithms and efficient DBMS support*, Data Mining and Knowledge Discovery, 4 (2000), pp. 193-216.

[62] M. ESTER, H.-P. KRIEGEL, J. SANDER and X. XU, *A density-based algorithm for discovering clusters in large spatial databases with noise*, *International Conference on Knowledge Discovery in Databases and Data Mining (KDD'96)*, Portland, Oregon, 1996, pp. 226-231.

[63] W. FAN, *Systematic data selection to mine concept-drifting data streams*, *10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'04)*, ACM, Seattle, WA, USA, 2004, pp. 128-137.

[64] U. M. FAYYAD, G. PIATETSKY-SHAPIRO, P. SMYTH and R. UTHURUSAMY, *Advances in knowledge discovery and data mining*, AAAI Press, 1996.

[65] D. H. FISHER, *Knowledge acquisition via incremental conceptual clustering*, Machine Learning, 2 (1987), pp. 139-172.

[66] S. FORTIN and L. LIU, *An object-oriented approach to multi-level association rule mining*, *5th International Conference on Information and Knowledge Management (CIKM'96)*, ACM, Rockville MD, USA, 1996, pp. 65-72.

[67] Y. FU and J. HAN, *Meta-rule-guided mining of association rules in relational databases*, *1st International Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases (KDOOD'95)*, 1995, pp. 39-46.

[68] T. FUKUDA, Y. MORIMOTO, S. MORISHITA and T. TOKUYAMA, *Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization*, ACM SIGMOD Record, 25 (1996), pp. 13-23.

[69] T. FUKUDA, Y. MORIMOTO, S. MORISHITA and T. TOKUYAMA, *Mining optimized association rules for numeric attributes*, *15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM, 1996, pp. 182-191.

[70] M. M. GABER, A. ZASLAVSKY and S. KRISHNASWAMY, *Mining data streams: A review*, SIGMOD Record, 34 (2005), pp. 18-26.

[71] V. GANTI, J. GEHRKE and R. RAMAKRISHNAN, *CACTUS-clustering categorical data using summaries*, in S. Chaudhuri and D. Madigan, eds., *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, San Diego, CA, 1999, pp. 73-83.

[72] V. GANTI, J. GEHRKE and R. RAMAKRISHNAN, *DEMON:Mining and monitoring evolving data*, IEEE Transactions on Knowledge and Data Engineering, 13 (2002), pp. 50-63.

[73] V. GANTI, J. GEHRKE and R. RAMAKRISHNAN, *A framework for measuring changes in data characteristics*, *18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ACM, Philadelphia, PA, 1999, pp. 126-137.

[74] L. GENG and H. J. HAMILTON, *Interestingness measures for data mining: A survey*, ACM Computing Surveys (CSUR), 38 (2006).

[75] B. GOETHALS, J. MUHONEN and H. TOIVONEN, *Mining non-derivable association rules*, *5th SIAM International Conference on Data Mining (SDM'05)*, SIAM, Newport Beach, CA, 2005, pp. 239-249.

[76] R. P. GOPALAN and Y. G. SUCAHYO, *ITL-MINE: Mining frequent itemsets more efficiently*, in L. Wang, S. Halgamuge and X. Yao, eds., *Internatiional Conference on Fuzzy Systems and Knowledge Discovery*, Springer, Singapore, 2002, pp. 167-172.

[77] M. S. GOUIDER and A. FARHAT, *Mining multi-level frequent itemsets under constraints*, International Journal of Database Theory & Application, 3 (2010), pp. 15-34.

[78] G. GRAHNE and J. ZHU, *Efficiently using prefix-trees in mining frequent itemsets*, *FIMI*, 2003, pp. 123-132.

[79] S. GUHA, R. RASTOGI and K. SHIM, *ROCK: A robust clustering algorithm for categorical attributes, the 15th International Conference on Data Engineering*, IEEE, 1999, pp. 512-521.

[80] G. K. GUPTA, A. STREHL and J. GHOSH, *Distance based clustering of association rules*, *Intelligent Engineering Systems Through Artificial Neural Networks(ANNIE 1999)*, ASME, St. Louis, Missouri, USA, 1999, pp. 759-764.

[81] H. J. HAMILTON and D. J. RANDALL, *Data mining with calendar attributes*, in J. F. Roddick and K. Hornsby, eds., *International Workshop on Temporal, Spatial and Spatio-Temporal Data Mining (TSDM2000)*, Springer, Lyon, France, 2001, pp. 117-132.

[82] J. HAN, *Mining knowledge at multiple concept levels*, *4th International Conference on Information and Knowledge Management*, ACM, 1995, pp. 19-24.

[83] J. HAN and A. FU, *Mining multiple-level association rules in large databases*, IEEE Transaction on Knowledge and Data Engineering, 11 (1999), pp. 798-805.

[84] J. HAN and Y. FU, *Discovery of multiple-level association rules from large databases*, *VLDB*, 1995, pp. 420-431.

[85] J. HAN, Y. FU, W. WANG, K. KOPERSKI and O. ZAIANE, *DMQL: A data mining query language for relational databases*, in R. Ng, ed., *ACM SIGMOD Workshop DMKD'96*, Montreal, Canada, 1996, pp. 27-34.

[86] J. HAN and M. KAMBER, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publishers, 2006.

[87] J. HAN, K. KOPERSKI and N. STEFANOVIC, *GeoMiner: A system prototype for spatial data mining*, in J. Peckham, ed., *ACM SIGMOD International Conference on the Management of Data (SIGMOD'97)*, ACM Press, Tucson, AZ, USA, 1997, pp. 553-556.

[88] J. HAN, J. PEI and Y. YIN, *Mining frequent patterns without candidate generation*, in W. Chen, J. Naughton and P. Bernstein, eds., *ACM SIGMOD International Conference on the Management of Data (SIGMOD 2000)*, ACM Press, Dallas, TX, USA, 2000, pp. 1-12.

[89] J. HAN, J. PEI, Y. YIN and R. MAO, *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*, Data Mining and Knowledge Discovery, 8 (2004), pp. 53-87.

[90] A. HANAU, *Die Prognose der Schweinepreise*, Hobbing, Berlin 1928.

[91] S. K. HARMS, J. DEOGUN and T. TADESSE, *Discovering sequential association rules with constraints and time lags in multiple sequences*, *13th International Symposium on Foundations of Intelligent Systems(ISMIS'02)*, Springer, Lyon, France, 2002, pp. 373-376.

[92] V. HAUTAMÄKI, I. KÄRKKÄINEN and P. FRÄNTI, *Outlier detection using k-nearest neighbour graph*, *17th International Conference on Pattern Recognition*, IEEE Computer Society, 2004, pp. 430-433.

[93] D. M. HAWKINS, *Identification of Outliers*, Chapman and Hall, London and New York, 1980.

[94] Z. HE, X. XU, J. Z. HUANG and S. DENG, *A frequent pattern discovery method for outlier detection*, *Advances in Web-Age Information Management*, Springer, 2004, pp. 726-732.

[95] J. HIPP, U. GÜNTZER and G. NAKHAEIZADEH, *Mining association rules: Deriving a superior algorithm by analyzing today's approaches*,

Principles of Data Mining and Knowledge Discovery, Springer, 2000, pp. 159-168.

[96] T.-P. HONG, K.-Y. LIN and B.-C. CHIEN, *Mining fuzzy multiple-level association rules from quantitative data*, Applied Intelligence, 18 (2003), pp. 79-90.

[97] T.-P. HONG, K.-Y. LIN and S.-L. WANG, *Fuzzy data mining for interesting generalized association rules*, Fuzzy Sets and Systems, 138 (2003), pp. 255-269.

[98] Z. HUANG, Z. ZHOU, T. HE and X. WANG, *ACAC: Associative classification based on all-confidence, IEEE International Conference on Granular Computing (GrC)*, IEEE, 2011, pp. 289-293.

[99] T. IMIELIŃSKI and A. VIRMANI, *MSQL: A query language for database mining*, Data Mining and Knowledge Discovery, 3 (1999), pp. 373-408.

[100] T. IMIELIŃSKI, A. VIRMANI and A. ABDULGHANI, *DMajor-Application programming interface for database mining*, Data Mining and Knowledge Discovery, 3 (1999), pp. 347-372.

[101] S. JAROSZEWICZ and D. A. SIMOVICI, *Pruning redundant association rules using maximum entropy principle*, *Advances in Knowledge Discovery and Data Mining*, Springer, 2002, pp. 135-147.

[102] H. JEUNG, M. L. YIU, X. ZHOU, C. S. JENSEN and H. T. SHEN, *Discovery of convoys in trajectory databases*, *VLDB Endowment*, 2008, pp. 1068-1080.

[103] Y. JIANG, Y. LIU, X. LIU and S. YANG, *Integrating classification capability and reliability in associative classification: A β-stronger model*, Expert Systems with Applications, 37 (2010), pp. 3953-3961.

[104] W. JIN, A. K. TUNG and J. HAN, *Mining top-n local outliers in large databases, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Francisco, California USA, 2001, pp. 293-298.

[105] T. JOHNSON, I. KWOK and R. NG, *Fast computation of 2-dimensional*

depth contours, *KDD'98*, 1998, pp. 224-228.

[106] P. KALNIS, N. MAMOULIS and S. BAKIRAS, *On discovering moving clusters in spatio-temporal data*, *9th International Symposium on Spatial and Temporal Databases(SSTD2005)*, Springer, Angra dos Reis, 2005, pp. 364-381.

[107] M. KAYA and R. ALHAJJ, *Mining multi-cross-level fuzzy weighted association rules*, *2nd IEEE International Conference Intelligent Systems*, IEEE, Varna, Bulgaria, 2004, pp. 225-230.

[108] M. KLEMETTINEN, H. MANNILA, P. RONKAINEN, H. TOIVONEN and A. I. VERKAMO, *Finding interesting rules from large sets of discovered association rules*, in N. Adam, B. Bhargava and Y. Yesha, eds., *3rd International Conference on Information and Knowledge Management*, ACM Press, Gaithersburg, Maryland, 1994, pp. 401-407.

[109] W. KLÖSGEN and J. M. ZYTKOW, *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, New York, 2002.

[110] E. M. KNORR and R. T. NG, *Finding intensional knowledge of distance-based outliers*, *25th VLDB Conference*, Edinburgh Scotland, 1999, pp. 211-222.

[111] E. M. KNORR and R. T. NG, *A unified approach for mining outliers*, *1997 Conference of the Centre for Advanced Studies on Collaborative Research*, IBM Press, 1997, pp. 11.

[112] E. M. KNORR, R. T. NG and V. TUCAKOV, *Distance-based outliers: Algorithms and applications*, The VLDB Journal, 8 (2000), pp. 237-253.

[113] R. KOHAVI, C. E. BRODLEY, B. FRASCA, L. MASON and Z. ZHENG, *KDD-Cup 2000 organizers' report: Peeling the onion*, SIGKDD Explorations, 2 (2000), pp. 86-93.

[114] I. KOPANAS, N. M. AVOURIS and S. DASKALAKI, *The role of domain knowledge in a large scale data mining project*, *Methods and Applications of Artificial Intelligence*, Springer, 2002, pp. 288-299.

[115] K. KOPERSKI and J. HAN, *Discovery of spatial association rules in geographic information databases*, *Advances in Spatial Databases*,

Springer, 1995, pp. 47-66.

[116] Y. KOU, C.-T. LU and D. CHEN, *Spatial weighted outlier detection*, in J. Ghosh, D. Lambert, D. Skillicorn and J. Srivastava, eds., *SIAM International Conference on Data Mining*, 2006, pp. 614-618.

[117] G. KUNDU, M. M. ISLAM, S. MUNIR and M. F. BARI, *ACN: An associative classifier with negative rules*, *11th IEEE International Conference on Computational Science and Engineering*, IEEE, 2008, pp. 369-375.

[118] G. KUNDU, S. MUNIR, M. F. BARI, M. M. ISLAM and K. MURASE, *A novel algorithm for associative classification*, *International Conference on Neural Information Processing (ICONIP'07)*, Springer, 2007, pp. 453-459.

[119] C. M. KUOK, A. FU and M. H. WONG, *Mining fuzzy association rules in databases*, ACM SIGMOD Record, 27 (1998), pp. 41-46.

[120] C.-H. LEE, C.-R. LIN and M.-S. CHEN, *On mining general temporal association rules in a publication database*, *IEEE International Conference on Data Mining (ICDM'01)*, IEEE, 2001, pp. 337-344.

[121] S. D. LEE and D. W.-L. CHEUNG, *Maintenance of discovered association rules: When to update?*, in R. Ng, ed., *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'97)*, ACM, Tucson AZ, USA, 1997.

[122] S. D. LEE, D. W. CHEUNG and B. KAO, *Is sampling useful in data mining? A case in the maintenance of discovered association rules*, Data Mining and Knowledge Discovery, 2 (1998), pp. 233-262.

[123] W. LEE, S. J. STOLFO and K. W. MOK, *Adaptive intrusion detection: A data mining approach*, Artificial Intelligence Review, 14 (2000), pp. 533-567.

[124] B. LENT, A. SWAMI and J. WIDOM, *Clustering association rules*, in A. Gray and P.-A. Larson, eds., *13th International Conference on Data Engineering*, IEEE Computer Society Press, Birmingham, UK, 1997, pp. 220-231.

[125] W. LI, J. HAN and J. PEI, *CMAR: Accurate and efficient classification*

based on multiple class-association rules, in N. Cercone, T. Lin and X. Wu, eds., *IEEE International Conference on Data Mining(ICDM'01)*, IEEE Computer Society, San Jose, CA, USA, 2001, pp. 369-376.

[126] X. LI, D. QIN and C. YU, *ACCF: Associative classification based on closed frequent itemsets*, *5th International Conference on Fuzzy Systems and Knowledge Discovery(FSKD'08)*, IEEE, 2008, pp. 380-384.

[127] Y. LI, P. NING, X. S. WANG and S. JAJODIA, *Discovering calendar-based temporal association rules*, Data & Knowledge Engineering, 44 (2003), pp. 193-218.

[128] P. LIANG and J. F. RODDICK, *Detecting anomalous longitudinal associations through higher order mining*, in K.-L. Ong, W. Li and J. Gao, eds., *2nd International Workshop on Integrating Artificial Intelligence and Data Mining(AIDM 2007)*, Australian Computer Society, Gold Coast, Queensland, 2007, pp. 19-27.

[129] P. LIANG and J. F. RODDICK, *RPL: A ruleset pattern language*, *International Conference on Artificial Intelligence and Industrial Application(AIIA2014)*, WIT Press, Hong Kong, 2014.

[130] P. LIANG, J. F. RODDICK and D. DE VRIES, *Searching frequent pattern and prefix trees for higher order rules*, in P. Christen, P. Kennedy, L. Liu, K.-L. Ong, A. Stranieri and Y. Zhao, eds., *11th Australian Data Mining Conference (AusDM 2013)*, Australian Computer Society, Inc, Canberra, Australia, 2013.

[131] K.-C. LIN, I.-E. LIAO and Z.-S. CHEN, *An improved frequent pattern growth method for mining association rules*, Expert Systems with Applications, 38 (2011), pp. 5154-5161.

[132] B. LIU, W. HSU, S. CHEN and Y. MA, *Analyzing the subjective interestingness of association rules*, Intelligent Systems and their Applications, IEEE, 15 (2000), pp. 47-55.

[133] B. LIU, W. HSU and Y. MA, *Discovering the set of fundamental rule changes*, *7th SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2001, pp. 335-340.

[134] B. LIU, W. HSU and Y. MA, *Integrating classification and association rule mining*, *Knowledge Discovery and Data Mining Conference (KDD'98)*, New York, NY, 1998, pp. 80-86.

[135] B. LIU, Y. MA and R. LEE, *Analyzing the interestingness of association rules from the temporal dimension*, *IEEE International Conference on Data Mining (ICDM'01)*, IEEE, 2001, pp. 377-384.

[136] B. LIU, Y. MA and C.-K. WONG, *Classification using association rules: Weaknesses and enhancements*, in V. Kumar, ed., *Data Mining for Scientific and Engineering Applications*, Springer, 2001, pp. 591-605.

[137] J. LIU, Y. PAN, K. WANG and J. HAN, *Mining frequent item sets by opportunistic projection*, *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Edmonton, Alberta, Canada, 2002, pp. 229-238.

[138] Y. LU, *Concept hierarchy in data mining: Specification, generation and implementation*, Simon Fraser University, Simon Fraser University, 1997.

[139] A. MADDALENA and B. CATANIA, *Towards an interoperable solution for pattern management*, *3rd International Workshop on Database Interoperability (INTERDB'07) (in conjunction with VLDB'07)*, Vienna, Austria, 2007.

[140] O. MAIMON and L. ROKACH, *Introduction to knowledge discovery in databases*, *Data Mining and Knowledge Discovery Handbook*, Springer, 2005, pp. 1-17.

[141] R. MAO, *Adaptive-FP: An efficient and effective method for multi-level multi-dimensional frequent pattern mining*, Simon Fraser University, Simon Fraser University, 2002.

[142] C. MARINICA, F. GUILLET and H. BRIAND, *Post-processing of discovered association rules using ontologies*, *IEEE International Conference on Data Mining Workshops (ICDMW'08)*, IEEE, 2008, pp. 126-133.

[143] J. MATA, J.-L. ALVAREZ and J.-C. RIQUELME, *An evolutionary algorithm to discover numeric association rules*, *ACM Symposium on*

Applied Computing, ACM, 2002, pp. 590-594.

[144] A. MCCALLUM, K. NIGAM and L. H. UNGAR, *Efficient clustering of high-dimensional data sets with application to reference matching*, *the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, 2000, pp. 169-178.

[145] Q. MEI and C. ZHAI, *Discovering evolutionary theme patterns from text - an exploration of temporal text mining*, in R. Grossman, R. Bayardo and K. Bennett, eds., *11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD05)*, ACM Press, Chicago, IL, 2005, pp. 198-207.

[146] R. MEO, G. PSAILA and S. CERI, *A new SQL-like operator for mining association rules*, *22nd VLDB Conference (VLDB'96)*, Mumbai, India, 1996, pp. 122-133.

[147] R. J. MILLER and Y. YANG, *Association rules over interval data*, in J. Peckham, ed., *ACM SIGMOD International Conference on the Management of Data*, ACM Press, Tucson, AZ, USA, 1997, pp. 452-461.

[148] C. MOONEY and J. RODDICK, *Mining itemsets-an approach to longitudinal and incremental association rule mining*, in A. Zanasi, C. Brebbia, N. Ebecken and P. Melli, eds., *3rd International Conference on Data Mining Methods and Databases*, WIT Press, Bologna, Italy, 2002, pp. 93-102.

[149] H. MOTULSKY, *Intuitive Biostatistics*, Oxford University Press New York, 1995.

[150] M. NANNI and D. PEDRESCHI, *Time-focused clustering of trajectories of moving objects*, Journal of Intelligent Information Systems, 27 (2006), pp. 267-289.

[151] R. T. NG, L. V. LAKSHMANAN, J. HAN and A. PANG, *Exploratory mining and pruning optimizations of constrained associations rules*, *ACM SIGMOD International Conference on Management of Data(SIGMOD '98)*, ACM, 1998, pp. 13-24.

[152] Q. NIU, S.-X. XIA and L. ZHANG, *Association classification based on*

compactness of rules, *2nd International Workshop on Knowledge Discovery and Data Mining (WKDD'09)*, IEEE, 2009, pp. 245-247.

[153] E. R. OMIECINSKI, *Alternative interest measures for mining associations in databases*, IEEE Transactions on Knowledge and Data Engineering, 15 (2003), pp. 57-69.

[154] K.-H. ONG, K.-L. ONG, W.-K. NG and E.-P. LIM, *Crystalclear: Active visualization of association rules*, *International Workshop on Active Mining (AM-2002) in Conjunction with the IEEE International Conference on Data Mining (ICDN'02)*, IEEE Press, Maebashi City, Japan, 2002.

[155] K.-L. ONG, W.-K. NG and E.-P. LIM, *Mining multi-level rules with recurrent items using FP'-tree*, *3rd International Conference on Information, Communications and Signal Processing*, Singapore, 2001.

[156] M. E. OTEY, A. GHOTING and S. PARTHASARATHY, *Fast distributed outlier detection in mixed-attribute data sets*, Data Mining and Knowledge Discovery, 12 (2006), pp. 203-228.

[157] B. OZDEN, S. RAMASWAMY and A. SILBERSCHATZ, *Cyclic association rules*, *14th International Conference on Data Engineering (ICDE'98)*, IEEE Computer Society Press, Orlando, Florida, USA, 1998, pp. 412-421.

[158] S. OZEL and H. GUVENIR, *An algorithm for mining association rules using perfect hashing and database pruning*, *10th Turkish Symposium on Artificial Intelligence and Neural Networks*, Springer, 2001, pp. 257-264.

[159] B. PADMANABHAN and A. TUZHILIN, *Small is beautiful: discovering the minimal set of unexpected patterns*, *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2000, pp. 54-63.

[160] R. PÁIRCÉIR, S. MCCLEAN and B. SCOTNEY, *Discovery of multi-level rules and exceptions from a distributed database*, *6th SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'00)*, ACM Press, 2000, pp. 523-532.

[161] G. K. PALSHIKAR, *Distance-based outliers in sequences*, *Distributed*

Computing and Internet Technology, Springer, 2005, pp. 547-552.

[162] T. PANG-NING, M. STEINBACH and V. KUMAR, *Introduction to data mining*, *Library of Congress*, 2006.

[163] S. PAPADIMITRIOU, H. KITAGAWA, P. B. GIBBONS and C. FALOUTSOS, *Loci: Fast outlier detection using the local correlation integral*, *19th International Conference on Data Engineering*, IEEE, 2003, pp. 315-326.

[164] J. S. PARK, M.-S. CHEN and P. S. YU, *An effective hash-based algorithm for mining association rules*, *ACM SIGMOD International Conference on Management of Data*, ACM, 1995, pp. 175-186.

[165] N. PASQUIER, Y. BASTIDE, R. TAOUIL and L. LAKHAL, *Discovering frequent closed itemsets for association rules*, *7th International Conference on Database Theory (ICDT99)*, Springer, 1999, pp. 398-416.

[166] N. PASQUIER, R. TAOUIL, Y. BASTIDE, G. STUMME and L. LAKHAL, *Generating a condensed representation for association rules*, Journal of Intelligent Information Systems, 24 (2005), pp. 29-60.

[167] A. PATCHA and J.-M. PARK, *An overview of anomaly detection techniques: Existing solutions and latest technological trends*, Computer Networks, 51 (2007), pp. 3448-3470.

[168] J. PEI, J. HAN, H. LU, S. NISHIO, S. TANG and D. YANG, *H-mine: Hyper-structure mining of frequent patterns in large databases*, *International Conference on Data Mining (ICDM'01)*, IEEE, San Jose, California, 2001, pp. 31-39.

[169] J. PEI, J. HAN and R. MAO, *CLOSET: An efficient algorithm for mining frequent closed itemsets*, *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, pp. 21-30.

[170] W. PERRIZO and A. DENTON, *Framework unifying association rule mining, clustering and classification*, *International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications (CSITeA03)* Rio de Janeiro, Brazil, 2003.

[171] A. PIETRACAPRINA and D. ZANDOLIN, *Mining frequent itemsets*

using patricia tries, in B. Goethals and M. Zaki, eds., *IEEE ICDM Workshop on Frequent Itemset Mining Implementations(FIMI'03)*, Melbourne, Florida, USA, 2003.

[172] D. POKRAJAC, A. LAZAREVIC and L. J. LATECKI, *Incremental local outlier detection for data streams*, *IEEE Symposium on Computational Intelligence and Data Mining*, IEEE, 2007, pp. 504-515.

[173] A. PRODROMIDIS, P. CHAN and S. STOLFO, *Meta-learning in distributed data mining systems: Issues and approaches*, in H. Kargupta and P. Chan, eds., *Advances in Distributed and Parallel Knowledge Discovery*, AAAI press, 2000.

[174] J. R. QUINLAN and R. M. CAMERON-JONES, *FOIL: A midterm report*, *European Conference on Machine Learning(ECML'93)*, Springer, Vienna, Austria, 1993, pp. 3-20.

[175] C. RAINSFORD and J. RODDICK, *Adding temporal semantics to association rules*, in J. Żytkow and J. Rauch, eds., *3rd European Conference on Principles of Knowledge Discovery in Databases (PKDD'99)*, Springer 1999, pp. 504-509.

[176] C. P. RAINSFORD, M. K. MOHANIA and J. F. RODDICK, *A temporal windowing technique for the incremental maintenance of association rules*, *8th International Database Workshop-Data Mining, Data Warehousing and Client/Server Databases*, 1997, pp. 78-94.

[177] S. RAMASWAMY, S. MAHAJAN and A. SILBERSCHATZ, *On the discovery of interesting patterns in association rules*, *24th International Conference on Very Large Data Bases(VLDB'98)* ACM Press, 1998, pp. 368-379.

[178] S. RAMASWAMY, R. RASTOGI and K. SHIM, *Efficient algorithms for mining outliers from large data sets*, *ACM SIGMOD International Conference on Management of Data*, ACM, 2000, pp. 427-438.

[179] D. J. RANDALL, H. J. HAMILTON and R. J. HILDERMAN, *Generalization for calendar attributes using domain generalization graphs*, *5th Workshop on Temporal Representation and Reasoning*, IEEE

Computer Society, Sanibel Island, Florida, USA, 1998, pp. 177-184.

[180] R. RASTOGI and K. SHIM, *Mining optimized support rules for numeric attributes*, *15th International Conference on Data Engineering*, IEEE, 1999, pp. 206-215.

[181] S. RINZIVILLO, D. PEDRESCHI, M. NANNI, F. GIANNOTTI, N. ANDRIENKO and G. ANDRIENKO, *Visually driven analysis of movement data by progressive clustering*, Information Visualization, 7 (2008), pp. 225-239.

[182] J. F. RODDICK and M. SPILIOPOULOU, *A survey of temporal knowledge discovery paradigms and methods*, IEEE Transactions on Knowledge and Data Engineering, 14 (2002), pp. 750-767.

[183] J. F. RODDICK, M. SPILIOPOULOU, D. LISTER and A. CEGLAR, *Higher order mining*, SIGKDD Explorations, 10 (2008), pp. 5-17.

[184] U. RUCKERT, L. RICHTER and S. KRAMER, *Quantitative association rules based on half-spaces: An optimization approach*, *4th IEEE International Conference on Data Mining(ICDM'04)*, IEEE, 2004, pp. 507-510.

[185] I. RUTS and P. J. ROUSSEEUW, *Computing depth contours of bivariate point clouds*, Computational Statistics & Data Analysis, 23 (1996), pp. 153-168.

[186] A. SALLEB-AOUISSI, C. VRAIN and C. NORTET, *QuantMiner: A Genetic Algorithm for Mining Quantitative Association Rules*, *IJCAI*, 2007.

[187] N. L. SARDA and N. SRINIVAS, *An adaptive algorithm for incremental mining of association rules*, *9th International Workshop on Database and Expert Systems Applications*, IEEE, 1998, pp. 240-245.

[188] A. SAVASERE, E. OMIECINSKI and S. NAVATHE, *Mining for strong negative associations in a large database of customer transactions*, *14th International Conference on Data Engineering*, IEEE, 1998, pp. 494-502.

[189] A. SAVASERE, E. R. OMIECINSKI and S. B. NAVATHE, *An efficient algorithm for mining association rules in large databases*, *21th International Conference on Very Large Data Bases*, Morgan Kaufmann

Publishers Inc, 1995, pp. 432-444.

[190] L. SHEN and H. SHEN, *Mining flexible multiple-level association rules in all concept hierarchies*, in G. Quirchmayr, E. Schweighofer and T. Bench-Capon, eds., *9th International Conference on Database and Expert Systems Applications, DEXA'98*, Springer, Vienna, Austria, 1998, pp. 786-795.

[191] L. SHEN and H. SHEN, *Mining flexible multiple-level association rules in all concept hierarchies*, *Database and Expert Systems Applications*, Springer, 1998, pp. 786-795.

[192] W.-M. SHEN, K. ONG, B. MITBANDER and C. ZANIOLO, *Metaqueries for data mining*, in M. F. Usama, P.-S. Gregory, S. Padhraic and U. Ramasamy, eds., *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996, pp. 375-398.

[193] A. SHILLABEER and D. PFITZNER, *Determining pattern element contribution in medical datasets*, *ACSW Frontiers 2007*, Australian Computer Society, Inc., 2007, pp. 233-240.

[194] R. SMITH, A. BIVENS, M. EMBRECHTS, C. PALAGIRI and B. SZYMANSKI, *Clustering approaches for anomaly based intrusion detection*, *Intelligent Engineering Systems through Artificial Eeural Eetworks*, ASME Press, 2002, pp. 579-584.

[195] M. SPILIOPOULOU and S. BARON, *Temporal evolution and local patterns*, *Local Patterns Detection*, Springer-Verlag Berlin Heidelberg, 2005, pp. 190-206.

[196] M. SPILIOPOULOU, I. NTOUTSI, Y. THEODORIDIS and R. SCHULT, *MONIC: Modeling and monitoring cluster transitions*, *12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD06)*, ACM, Philadelphia, USA, 2006, pp. 706-711.

[197] M. SPILIOPOULOU and J. F. RODDICK, *Higher order mining: Modelling and mining the results of knowledge discovery*, in N. Ebecken and C. Brebbia, eds., *2nd International Conference on Data Mining Methods and Databases*, WIT Press, Cambridge, UK, 2000, pp. 309-320.

[198] R. SRIKANT and R. AGRAWAL, *Mining generalized association rules*, *21th International Conference on Very Large Data Bases (VLDB'95)*, 1995, pp. 407-419.

[199] R. SRIKANT and R. AGRAWAL, *Mining quantitative association rules in large relational tables*, *ACM SIGMOD International Conference on Management of Data*, ACM Press, 1996, pp. 1-12.

[200] R. SRIKANT, Q. VU and R. AGRAWAL, *Mining association rules with item constraints*, *KDD'97*, 1997, pp. 67-73.

[201] Y. G. SUCAHYO and R. P. GOPALAN, *CT-ITL: Efficient frequent item set mining using a compressed prefix tree with pattern growth*, in K. DieterSchewe and X. Zhou, eds., *14th Australasian Database Conference*, Australian Computer Society, Inc., Adelaide, Australia, 2003, pp. 95-105.

[202] P. SUN and S. CHAWLA, *On local spatial outliers*, *4th IEEE International Conference on Data Mining (ICDM'04)*, IEEE, 2004, pp. 209-216.

[203] P. SUN, S. CHAWLA and B. ARUNASALAM, *Mining for outliers in sequential databases*, *SIAM International Conference on Data Mining*, SIAM, 2006, pp. 94-105.

[204] P.-N. TAN, V. KUMAR and J. SRIVASTAVA, *Selecting the right interestingness measure for association patterns*, *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'02)*, ACM, 2002, pp. 32-41.

[205] J. TANG, Z. CHEN, A. W.-C. FU and D. W. CHEUNG, *Enhancing effectiveness of outlier detections for low density patterns*, *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2002, pp. 535-548.

[206] C. M. TENG, *Learning from dissociations*, *4th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'02)*, Springer, Aix-en-Provence, France, 2002.

[207] F. THABTAH, P. COWLING and Y. PENG, *MCAR: Multi-class classification based on association rule*, *3rd IEEE International Conference on Computer Systems and Applications*, IEEE, Cairo, Egypt,

2005, pp. 1-7.

[208] F. THABTAH, Q. MAHMOOD, L. MCCLUSKEY and H. ABDEL-JABER, *A new classification based on association algorithm*, Journal of Information and Knowledge Management, 9 (2010), pp. 55-64.

[209] S. THOMAS, S. BODAGALA, K. ALSABTI and S. RANKA, *An efficient algorithm for the incremental updation of association rules in large databases*, *3rd International Conference on Knowledge Discovery and Data Mining (KDD 97)*, ACM Press, New Port Beach, CA, USA, 1997, pp. 263-266.

[210] H. TOIVONEN, M. KLEMETTINEN, P. RONKAINEN, K. HÄTÖNEN and H. MANNILA, *Pruning and grouping discovered association rules*, *ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, Heraklion, Greece, 1995, pp. 47-52.

[211] A. TUZHILIN and G. ADOMAVICIUS, *Handling very large numbers of association rules in the analysis of microarray data*, *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, Edmonton, Alberta, Canada, 2002, pp. 396-404.

[212] A. TUZHILIN and B. LIU, *Querying multiple sets of discovered rules*, *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'02)*, ACM Press, 2002, pp. 52-60.

[213] M. R. VIEIRA, P. BAKALOV and V. J. TSOTRAS, *On-line discovery of flock patterns in spatio-temporal data*, *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, 2009, pp. 286-295.

[214] A. VINUEZA and G. GRUDIC, *Unsupervised outlier detection and semi-supervised learning*, University of Colorado at Boulder, 2004.

[215] J. WANG, J. HAN and J. PEI, *Closet+: Searching for the best strategies for mining frequent closed itemsets*, *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'03)*, ACM, 2003, pp. 236-245.

[216] K. WANG, L. TANG, J. HAN and J. LIU, *Top down fp-growth for*

association rule mining, 6th Pacific-Asia Conference (PAKDD 2002) Springer, Taipei, Taiwan, 2002, pp. 334-340.

[217] K. WANG, S. H. W. TAY and B. LIU, *Interestingness-based interval merger for numeric association rules, KDD*, 1998, pp. 121-128.

[218] J. WAY and E. A. SMITH, *The evolution of synthetic aperture radar systems and their progression to the EOS SAR*, IEEE Transactions on Geoscience and Remote Sensing, 29 (1991), pp. 962-985.

[219] G. I. WEBB, *Discovering associations with numeric variables, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'01)*, ACM, 2001, pp. 383-388.

[220] R. C.-W. WONG and A. W.-C. FU, *ISM: Item selection for marketing with cross-selling considerations, Advances in Knowledge Discovery and Data Mining*, Springer, 2004, pp. 431-440.

[221] X. WU, C. ZHANG and S. ZHANG, *Efficient mining of both positive and negative association rules*, ACM Transactions on Information Systems (TOIS), 22 (2004), pp. 381-405.

[222] X. XU, G. HAN and H. MIN, *A novel algorithm for associative classification of image blocks, 4th IEEE International Conference on Computer and Information Technology*, IEEE, 2004, pp. 46-51.

[223] Y. XU and Y. LI, *Generating concise association rules, 16th ACM conference on Information and Knowledge Management (CIKM'07)*, ACM, 2007, pp. 781-790.

[224] Y. XU, Y. LI and G. SHAW, *A reliable basis for approximate association rules*, IEEE Intelligent Informatics Bulletin, 9 (2008), pp. 25-31.

[225] K. YAMANISHI and J.-I. TAKEUCHI, *Discovering outlier filtering rules from unlabeled data: combining a supervised learner with an unsupervised learner, 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Francisco, California, 2001, pp. 389-394.

[226] K. YAMANISHI, J.-I. TAKEUCHI, G. WILLIAMS and P. MILNE, *On-line unsupervised outlier detection using finite mixtures with discounting*

learning algorithms, 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Boston, Massachusetts, USA, 2004, pp. 320-324.

[227] Y. YE, Q. JIANG and W. ZHUANG, *Associative classification and post-processing techniques used for malware detection, 2nd International Conference on Anti-counterfeiting, Security and Identification (ASID'08),* IEEE, 2008, pp. 276-279.

[228] X. YIN and J. HAN, *CPAR: Classification based on predictive association rules, SIAM Conference on Data Mining (SDM'03),* SIAM, San Francisco,California, USA, 2003, pp. 369-376.

[229] S.-C. YOON, L. J. HENSCHEN, E. PARK and S. MAKKI, *Using domain knowledge in knowledge discovery, 8th International Conference on Information and Knowledge Management,* ACM, 1999, pp. 243-250.

[230] D. YU, G. SHEIKHOLESLAMI and A. ZHANG, *Findout: Finding outliers in very large datasets,* Knowledge and Information Systems, 4 (2002), pp. 387-412.

[231] J. X. YU, W. QIAN, H. LU and A. ZHOU, *Finding centric local outliers in categorical/numerical spaces*, Knowledge and Information Systems, 9 (2006), pp. 309-338.

[232] M. J. ZAKI, *Generating non-redundant association rules, 6th International Conference on Knowledge Discovery and Data Mining (SIGKDD'00),* AAAI Press, Boston, MA, USA, 2000, pp. 34-43.

[233] M. J. ZAKI, *Mining non-redundant association rules,* Data Mining and Knowledge Discovery, 9 (2004), pp. 223-248.

[234] H. ZHANG, B. PADMANABHAN and A. TUZHILIN, *On the discovery of significant statistical quantitative rules, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'04),* ACM, 2004, pp. 374-383.

[235] J. ZHANG and H. WANG, *Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance*, Knowledge and Information Systems, 10 (2006), pp. 333-355.

[236] Q. ZHAO and S. S. BHOWMICK, *Association rule mining: A survey*, Nanyang Technological University, Singapore, Singapore, 2003.

[237] Y. ZHAO, C. ZHANG, L. CAO and I. GLOBAL, *Post-mining of association rules: Techniques for effective knowledge extraction*, Information Science Reference, New York, 2009.