A Study of Frequent Pattern Mining in Transaction Datasets

A thesis presented in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science

at Massey University, Palmerston North,
New Zealand.

Luofeng XU

2011

To my family.

Abstract

Within data mining, the efficient discovery of frequent patterns—sets of items that occur together in a dataset—is an important task, particularly in transaction datasets. This thesis develops effective and efficient algorithms for frequent pattern mining, and considers the related problem of how to learn, and utilise, the characteristics of the particular datasets being investigated.

The first problem considered is how to mine frequent closed patterns in dynamic datasets, where updates to the dataset are performed. The standard approach to this problem is to use a standard pattern mining algorithm and simply rerun it on the updated dataset. An alternative method is proposed in this thesis that is significantly more efficient provided that the size of the updates is relatively small.

Following this is an investigation of the pattern support distribution of transaction datasets, which measures the numbers of times each pattern appears within the dataset. The evidence for the pattern support distribution of real retail datasets obeying a power law is investigated using qualitative appraisals and statistical goodness-of-fit tests, and the power law is found to be a good model. Based on this, the thesis demonstrates how to efficiently estimate the pattern support distribution based on sampling techniques, reducing the computational cost of finding this distribution.

The last major contribution of the thesis is to consider novel ways to set the main user-specified parameters of frequent pattern mining, the minimum support, which defines how many times a pattern needs to be seen before it is 'frequent'. This is a critical parameter, and very hard to set without a lot of knowledge of the dataset. A method to enable the user to specify rather looser requirements for what they require from the mining is proposed based on the assumption of a power-law-based pattern support distribution and fuzzy logic techniques.

KEYWORDS: Data mining, Frequent pattern mining, Dynamic transaction dataset, Incremental mining, Pattern support distribution, Power-law relationship, Fuzzy logic

Declaration

This aims to certify that the research carried out for my Doctorial Thesis entitled "A Study of Frequent Pattern Mining in Transaction Datasets" in the School of Engineering and Advanced Technology, Massey University, Palmerston North, New Zealand is my own work and that no portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Acknowledgements

If any value of this research can be recognized, much of it is due to the endless support that I received from many people. I would like to take this opportunity to express my appreciations to those people, although here I can only mention a few.

First and foremost, my deep gratitude goes to my supervisors and mentors of Dr. Ruili Wang and A/Prof. Stephen Marsland. I thank them for their continuous encouragement, responsiveness, support, considerations, patience and enthusiasm, and for sharing with me their knowledge and experience. This research comes from the helpful discussions with them. It is needless to say that without them, this research would have been impossible.

My gratitude is also expressed to all academic and general staff in our school for always being helpful since the very beginning of this research. Working with them is always so enjoyable. Many thanks to Massey University for financially supporting my PhD study by offering me Massey University Doctoral Scholarship.

Finally, I gratefully acknowledge that I would not be in my present position without the endless love and constant support of my family. Moreover, I would like to thank all of my friends as well, for their presence at times when I most needed them.

I present this research to all of those people and let my warm and sincere blessings accompany them wherever they are!

Greatest thanks to everyone!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A journey of a thousand miles begins with a single step.

— Lao-tzu

In recent years, the amount of data generated and collected by information devices has increased enormously. To analyze these large amounts of data, an interdisciplinary research field, data mining, has appeared. Frequent patterns play a core role in many data mining applications. Thus, it is necessary to design algorithms for mining frequent patterns and learning the characteristics of the target data in order to achieve better algorithmic efficiency and better understanding of the mined results.

This first chapter provides general information about this research. In particular, Section 1.1 discusses what frequent patterns are, and how and why mining them in transaction datasets is useful. A set of aims and objectives for the thesis are derived from the discussion in Section 1.2. The main contributions of the research are listed in Section 1.3, while Section 1.4 gives an overview of the thesis, which gives a brief outline of the research issues and proposed solutions.

1.1 Data Mining

The widespread adoption of information technology in various fields in recent years, such as the popularity of Internet use and bar code scanners collecting customer data, has lead to explosive growth in the volume of generated and stored data. Advanced database techniques collect data in common and consistent formats from many sources, but the fact that the data is stored ready for processing does not mean that it is easy to process. Like a gold mine, there is a lot of potential useful

information and knowledge hidden in the data, which can be of great benefit to users, but extracting the useful nuggets requires a lot of processing.

While the collected data is generally too huge to be manually processed, at the same time computer technology has improved significantly, in both processing power and storage capacity. Using computer technology to discover the information and knowledge in the growing massive data becomes possible, affordable, and extremely important. These factors create a great need for new techniques to automatically transform the huge amount of target data into useful information and knowledge in a reasonable amount of time.

This has led to an emerging research area called *Data Mining* (DM) or *Knowledge Discovery in Databases* (KDD)[1]. Since the beginning of the 1990s, data mining has received a lot of attention, and has expanded into many application areas and specialised conferences and journals. Meanwhile, data mining is also progressively popular in industries such as insurance, financial services and telecommunications.

Data mining can be defined as the *nontrivial discovery of implicit, previously unknown, and potentially useful information from very large volumes of data* [37, 38, 40, 94]. As an interdisciplinary research area, it represents the integration of several research fields, principally statistics, machine learning, information theory, and database systems.

People often wonder what the fundamental differences are between data mining and statistics, since both of them deal with data analysis. There are at least two fundamental differences. One is the data itself. Compared with statistics, data mining often deals with huge amounts of data, which can have a variety of types including multimedia and complex data (Spatial, VRML, XML, etc). The other is that data mining belongs to exploratory analysis, while statistics more commonly deals with confirmatory analysis. Confirmatory analysis firstly sets up a hypothesis, and then uses data to test whether the hypothesis is correct. In exploratory analysis, information can be extracted from data without setting up a relevant hypothesis first. Besides, huge amount of simple or complex data can hide information, which users may never practically build the right hypothesis to test for.

The most common tasks of data mining include:

Association rule mining which aims to discover relationships among variables.

Classification which aims to assign new data into a number of given classes.

[1]Data mining is often referred to as knowledge discovery in databases, or a vital step of KDD, depending on different definitions or views.

Clustering which aims to discover unknown classes in given data. Data in each discovered class is similar in some way.

Regression which aims to discover a model that can describe given data with the least error.

Outlier detection which aims to discover data that significantly deviates from the rest of the given data.

1.1.1 Frequent Pattern Mining

Frequent pattern mining (FPM) has been a vital subject in the field of data mining since the beginning. In general, *frequent patterns* (FPs) are referred to *itemsets, subsequences, or substructures whose appearance frequency in the target dataset is not less than a (user-specified) threshold value*. Frequent pattern mining plays an essential role in many data mining tasks, such as various kinds of association rule mining [81, 118], correlation mining [19, 139], sequential pattern mining [10], emerging pattern mining [32, 121], episode mining [54, 79], partial periodicity [49, 125], structured pattern mining [46], associative classification [70], and frequent pattern-based clustering [126]. Thus, effective and efficient frequent pattern mining has become an important research topic.

Frequent pattern mining has been successfully applied in an extensive range of real world applications for better decision support, including:

In business with the globalization of trade, companies have more and more customers and transactions. Thus, they need to know risks (e.g., can this transaction be fraudulent?) and opportunities (e.g., what product is the customer likely to buy next?). Mining frequent patterns may help in designing promotions, discounts, shelf organization, store layout, special advertisements, storage management, and prediction of potential markets [27].

In healthcare mining frequent patterns can be used to identify abnormity to help manage medicine, support doctors to make decisions about treatments, prevent the outbreaks of diseases, and discover the actions of genes [89, 122].

In education mining frequent patterns can help alter the teaching methods in order to e.g., improve teaching quality by analyzing the situation of students, manage research projects so that scholars can pay more attention to their

research instead of administration, select the contents of teaching according to the ability of students [75].

In disaster prevention mining frequent patterns can help forecast the weather by analyzing different environmental factors such as temperature, humidity and wind, especially for the impending disaster to reduce loss and casualties [138].

In other fields mining frequent patterns can help police investigate crimes and improve public security [11]. It can even help engineers design vehicles [1].

Although some researchers argue that in many application areas frequency is not the best measure to determine the significance of a pattern [71, 107, 109], mining frequent patterns is still very useful. These researchers point out that the significance of a pattern relies on a number of other parameters such as the type of contained items, the length of the pattern, and various attributes associated with each individual item. In such cases, frequent pattern mining can still be used as a pre-processing step to identify a set of candidate patterns for further processing.

1.1.2 Overview of the Research Area

An enormous amount of research has contributed to frequent pattern mining and many remarkable algorithms have been developed. However, there are still some interesting issues that need to be investigated. The first work, published in 1993 by Agrawal et al. [4], handles so-called "transaction datasets" or "supermarket datasets". Since then, transaction datasets have been widely used in frequent pattern mining. A *transaction dataset* is a dataset listing which items are purchased in each transaction. A transaction T is represented by a pair, defined as $T = \langle TID, \text{list of items} \rangle$, where TID is a unique identification number for the transaction and "list of items" is a list of items included in the transaction. Table 1.1 shows an example of a transaction dataset for a grocery store. In general, transaction datasets require a transformation of data from other formats into a single table. Users are then able to mine frequent patterns in this table.

One limitation of previous work on frequent pattern mining is that datasets are very rarely static; rather, they are updated on a regular basis, and so their characteristics evolve. These *dynamic transaction datasets* present different challenges. Most existing algorithms for mining frequent patterns belong to conventional frequent pattern mining, which was initially developed for mining frequent patterns in *static* datasets where data is not expected to vary with time.

TID	List of Items
100	Just Juice, Budget Milk, Molenberg Bread
200	Surf Laundry Powder, NZ Yummy Apples
300	Fresh 'n Fruity Yoghurt, NZ Yummy Apples
400	Fresh 'n Fruity Yoghurt, Budget Milk, Molenberg Bread
500	Molenberg Bread, Surf Laundry Powder, Fresh 'n Fruity Yoghurt
600	NZ Yummy Apples, Budget Milk, Molenberg Bread
700	NZ Yummy Apples

Table 1.1: A sample transaction dataset for a grocery store

For example, in a large organization, data is generally distributed over multiple databases. Each database has its own focus, which stores the data related to some specified subjects, and each database can be asynchronously updated. To mine all data stored in those databases, one can transform all data into a single database and then mine it. Besides lack of flexibility, this method will suffer from huge storage space, heavy IO cost, heavy communication cost, etc. Normally, this mining will not be an economical strategy that any organization would be willing to adopt.

There are historical reasons why conventional frequent pattern mining focuses on static datasets. In the past, data was generated relatively slowly and was also relatively hard to collect. Thus, the size of datasets targeted for final analysis was often not very large and the updating speed of datasets was quite slow. Another possible reason is that at that time researcher were apt to study stable phenomena. We will propose a modification of the standard algorithms so that they work on dynamic datasets as part of this thesis.

With the volume of data increasing rapidly, a better understanding of the characteristics of target datasets in order to improve the performance of frequent pattern mining is important. An important characteristic of real datasets, named the *Pattern Support Distribution* (PSD), can contribute to solving this problem. The *support* of a pattern is the frequency of its appearance in a dataset, and the distribution of patterns against their corresponding supports in a dataset is known as the *pattern support distribution* of the dataset. From the view of probability theory and statistics, a pattern support distribution is a discrete probability distribution, which expresses the probability that the support of an arbitrary pattern in a dataset is equal to some particular value.

A fundamental issue of frequent pattern mining is related to the user-specified threshold value for identifying frequent patterns in a target dataset. Most ap-

proaches to conventional frequent pattern mining have a parameter called the *minimum support*, which directly determines which patterns are common enough to be of interest to users during the mining processes. A common deficiency of those approaches is the assumption that users can specify a suitable data-dependent threshold value for the minimum support. In reality, it is hard for users to know what value is appropriate, especially when they do not have detailed knowledge about the target dataset and the application domain. Besides, different tasks or users may require different data-dependent threshold values for the minimum support, even if the target dataset is the same. Therefore, it is necessary to develop new approaches to overcoming this drawback.

Another fundamental issue of data mining is how to efficiently mine in very large datasets, which usually generates prohibitive computational costs. For example, to find the support of a pattern in a dataset requires scanning the whole dataset once. If there are $|I|$ items in a transaction dataset, to find the support of each pattern in the dataset, one may have to scan the whole dataset $(2^{|I|} - 1)$ times. The computational costs involved adversely affect the effectiveness and efficiency of mining processes. Methods such as sampling and iterative updates can be used to reduce the computational costs, as will be further discussed in this thesis.

1.2 Aims and Objectives

This section first describes the general aims of this research. The aims are then broken down into a set of objectives that the research should meet. These objectives are falsifiable statements, in order that an appropriate scientific investigation of each of the objectives can be implemented [97]. The objectives will also act as the measure for analyzing whether the research is successful.

1.2.1 Aims

We concentrate on only one kind of information—frequent patterns. The goal of this research is to study and improve frequent pattern mining in transaction datasets. To achieve this goal, we address critical areas where our contributions can be made. Technically, the work presented in this thesis includes two distinct areas, each with specific aims.

Designing effective and efficient mining algorithms One area of this research

focuses on designing effective and efficient algorithms for mining frequent patterns in transaction datasets. As mentioned previously, in general, data mining approaches are computationally expensive and time-consuming due to the very large data volumes dealt with. For users, the desired frequent patterns have to be found under an acceptable limitation of execution time. It is necessary to pay special attention to the effectiveness and efficiency of mining algorithms. In this research, we are particularly interested in developing effective and efficient approaches to mining frequent patterns in dynamic transaction datasets and mine frequent patterns in static transaction datasets without data-dependent threshold values specified by users.

Learning the characteristics of datasets The other is to learn the characteristics of datasets. Especially, we concentrate on *Pattern Support Distributions* of transaction datasets. Pattern support distributions can be very useful for designing mining algorithms and tuning the performance of mining algorithms with target datasets.

Theoretical analyses and experimental studies will be utilized to examine the performance of our proposed approaches and formulate the future directions for further improvements.

A set of objectives is given next. These objectives were used to guide the development of our proposed approaches.

1.2.2 Objectives

Based on evaluation of the literature we have identified three important issues in frequent pattern mining, which will be investigated. There are:

1. Datasets are treated as static, but many are updated regularly. We call these dynamic datasets.

2. While many pattern mining algorithms can be relatively efficient, datasets can be very large and at worst the problem is in complexity class NP-hard. To reduce the computational time, approaches by sampling the dataset and/or mining knowledge of structures in the dataset are required.

3. The user-specified minimum support is a crucial input to many pattern mining algorithms. However, it is difficult to set, and minor changes to it affect the

results markedly. Automatic methods based on estimates from the user should improve the effectiveness of the mining processes.

- The desired algorithm for mining frequent patterns in dynamic transaction datasets should:

    1. successfully discover all frequent patterns in a target dynamic transaction dataset with a user-specified data-dependent threshold value
    2. handle the updates (i.e., adding new transactions, deleting existing transactions, or modifying existing transactions) of the target dynamic transaction dataset
    3. scale well with the number of transactions in an update
    4. scale well with the number of transactions in the base dataset
    5. demonstrate its computational efficiency

- The desired approaches of exploring pattern support distributions of real static transaction datasets should:

    1. investigate whether any statistical relationships exist in the pattern support distributions of real static transaction datasets
    2. describe the influence of sampling on the estimation of a power-law-based pattern support distribution
    3. design an effective method to quickly estimate the power-law-based pattern support distribution of a target static transaction dataset

- The desired algorithm for mining frequent patterns in static transaction datasets with a user-specified data-independent threshold value should:

    1. not require that users have any detailed knowledge about the characteristics of the target dataset
    2. allow users to specify data-independent threshold values instead of a data-dependent value for the proposed algorithm, since without detailed knowledge about the characteristics of target datasets it difficult for users to set suitable data-dependent values for their applications

3. generate an appropriate data-dependent threshold value for the minimum support required by most conventional mining algorithms of frequent pattern
4. easily control the number of expected patterns

1.3 Main Contributions

The main contributions of this thesis include:

1. We propose a modification of a conventional mining method to incrementally mine frequent closed patterns in a dynamic transaction dataset.

   Our proposed algorithm is based on the extensions of two tree-like data structures: DB-tree [36] and two-level hash-indexed result tree [120]. The two extended data structures compactly store all information required by the mining processes. Thus, they can reduce the chance of re-scanning the whole updated dataset, and provide the flexibility whenever the user-specified minimum support value is changed (e.g., in interactive mining). Furthermore, since they can also be built piecewise and then merge the pieces together, it can provide the possibility of parallel computing to enhance the performance of mining tasks.

   Our extensive experimental studies show that for mining frequent closed patterns in dynamic transaction datasets, our proposed algorithm has a clear performance advantage over that of rerunning representative algorithms of conventional frequent pattern mining.

   This study demonstrates how to extend conventional mining methods to handle dynamic datasets and also gives an insight into the strengths and limitations of the incremental approach.

2. We study an important characteristic—pattern support distributions—of real transaction datasets. We verify that power-law relationships exist in pattern support distributions of real retail transaction datasets through qualitative appraisals based on visualizations, and statistical goodness-of-fit tests. The self-similarity phenomenon linked to power-law relationships is also discovered in pattern support distributions.

Besides, we investigate the influence of sampling on estimating the power-law-based pattern support distribution of real static transaction datasets in order to shed a light on how to use sampling to reduce the computational costs related to the estimation.

Furthermore, we propose cost-effective approaches to estimating the parameters of the power-law-based pattern support distribution of a real static transaction dataset without requiring all patterns and their corresponding supports in the target dataset. The experimental results demonstrate that the proposed approach is able to achieve quick estimation with relatively high accuracy.

The large number of experimental results and theoretical analyses in our study reveal the connections between power-law relationships in pattern support distributions of real retail transaction datasets and some fundamental mathematical problems, such as the hypergeometric function, which reflects the interestingness and complexity of our real world.

3. We study the problem of mining frequent patterns with a data-dependent threshold value specified by users for the minimum support. Based on power-law-based pattern support distributions and fuzzy logic techniques, we propose a new approach to dealing with this issue in static transaction datasets.

   Our proposed approach requires that users specify two data-independent reference values instead of a data-dependent threshold value for the minimum support. Our approach can then transform the user-specified data-independent reference value to an appropriate data-dependent threshold value for the minimum support demanded by most conventional mining algorithms. In this way, our proposed approach allows users who do not have the detailed characteristics of a target dataset to specify the threshold value for the minimum support.

   Experimental studies demonstrate the effectiveness and efficiency of the proposed approach, which can effectively prevent users from wasting time on searching for an appropriate data-dependent threshold value for the minimum support, and provide users with the power to control the number of mined results. Furthermore, our proposed approach can provide the flexibility to mine frequent patterns in situations where the threshold value for the minimum support may change with time, such as interactive mining and mining in dynamic datasets.

This study is an effective attempt to use fuzzy logic control techniques to process imprecise quantitative user requirements and deal with inconsistency among user requirements, which gives an insight into these issues.

1.4 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2: Incremental mining of frequent closed patterns. Most conventional algorithms for frequent pattern mining assume that the mining process is performed on a static dataset. However, datasets can be dynamic due to updates. In this chapter, we explore the problem of mining frequent closed patterns in a dynamic transaction dataset, with a data-dependent threshold value specified by users. Frequent closed patterns are a condensed representation of frequent patterns. Compared with frequent patterns in a dataset, the corresponding frequent closed patterns have all the information about the frequent patterns in less memory space, which increases the efficiency of the mining process.

First of all, we define dynamic transaction datasets, where updates of adding new transactions, deleting existing transactions, or modifying existing transactions occur. We then analyze the challenges brought by dynamic transaction datasets—previous mined frequent closed patterns may be invalid and new frequent closed patterns may appear. After comprehensively reviewing the current development of frequent pattern mining, including frequent closed patterns in both static and dynamic transaction datasets, we investigate how to extend the conventional mining algorithms to solve this problem.

An algorithm to solve the problem is proposed in detail. Our algorithm is an extension of CLOSET+ [120], one of the most efficient algorithms for mining frequent closed patterns in static transaction datasets. Based on two variant tree-like data structures compactly storing all information required by mining processes, our proposed algorithm efficiently monitors the target dynamic transaction dataset and the mined results. After an update, the previously-mined results are updated by only considering the changes in the dataset. Hence, the frequent closed patterns in the updated dataset can be obtained without re-scanning and re-computing all transactions in the updated dataset.

The performance of our proposed algorithm is compared with CLOSET+ in a number of experiments. The experimental results show that our proposed algorithm has performance improvements for dynamic transaction datasets compared with using conventional mining algorithms designed for static transaction datasets.

Chapter 3: Investigating power-law relationships in pattern support distributions. It is worthwhile to learn the characteristics of datasets in order to design effective and efficient algorithms for mining frequent patterns, to tune the performance of frequent pattern mining, and to better understand the mined results.

In this chapter, we introduce an important characteristic of datasets—the pattern support distribution—which shows how patterns are distributed against their corresponding supports in a static dataset. In this chapter, we are more interested in power-law relationships in pattern support distributions of static transaction datasets. For other types of dataset, a similar exploration can be made using the methods described in this chapter.

This chapter starts with an introduction to power-law relationships and pattern support distributions. Then an overview of related work is presented. After that, the hypothesis that power-law relationships exist in pattern support distributions of static transaction datasets is examined by two methods. One is the qualitative appraisal based on visualization—a power-law relationship shows a roughly straight line in a log-log plot. The other is to use statistical goodness-of-fit tests. The Kolmogorov-Smirnov statistic and p-value are adopted here. After verifying that the existence of power-law relationships in the pattern support distributions of two real retail transaction datasets (i.e., the Retail and BMS-POS datasets) cannot be ruled out at the level of basic distributions, we extend this finding to all real retail transaction datasets by utilizing the bootstrap method and the universality property of power-law relationships. Finally, the self-similarity phenomenon[2] linked to power-law relationships is also discussed.

Chapter 4: Efficiently estimating power-law-based pattern support distributions. In general, to accurately identify the power-law-based pattern

[2]A self-similar object is exactly or approximately similar to a part of itself. In other words, the whole has the same shape as one or more of the parts.

support distribution of a dataset requires knowing all patterns and their corresponding supports in the dataset in advance. However, discovering this can be extremely computationally expensive. Sometimes, it is even impossible in practice.

It seems reasonable that using sample datasets instead of the complete dataset to estimate the power-law-based pattern support distribution may reduce the computational cost without significant detriment to the estimation accuracy. Consequently, we investigate the influence of sampling on estimating power-law-based pattern support distributions of static transaction datasets. Our experimental results and analyses show that, due to the uncertainty brought by sampling, simply using sampling methods cannot reliably and efficiently estimate the power-law-based pattern support distribution of a large transaction dataset.

Therefore, we propose novel approaches to quickly estimating the power-law-based pattern support distribution of a static transaction dataset by using two statistics: the mean support of patterns and the mean number of patterns per transaction. As demonstrated by our theoretical discussion and experimental results, our proposed approaches can quickly estimate the characteristics of the power-law-based pattern support distribution of static transaction datasets with relatively high estimation accuracy.

Chapter 5: Fuzzy frequent pattern mining. Users are usually required to specify a data-dependent threshold value for the minimum support that is a parameter of most conventional mining algorithms for frequent patterns. They are assumed to know precisely what value is suitable for their applications, or that an appropriate value can be easily found. However, this may not be true in all cases. Releasing users from this requirement has become a fundamental research issue of frequent pattern mining.

After an introduction to this issue and a review of related work, we propose a novel approach to handling this issue. Instead of requiring an appropriate data-dependent threshold value for the minimum support, our proposed approach asks users to specify two data-independent reference values as their qualitative expectation for mined results. In this way, users do not need to worry about whether specified values are suitable for mining their target datasets. Based on the power-law-based pattern support distributions introduced in Chapters 3

and 4 and fuzzy logic control techniques, our proposed approach transforms the user-specified data-independent reference values into an appropriate minimum support value, which is then used by the mining processes.

The performance of our proposed approach is examined by experiments. The experimental results demonstrate the effectiveness and efficiency of our proposed fuzzy approach.

Chapter 6: Summary and conclusions. This chapter presents a brief summary of the thesis, discusses the key findings, draws some main conclusions, and points out some directions for future developments.

Chapter 2

Incremental Mining of Frequent Closed Patterns

Just definitions either prevent or put an end to a dispute.

— Nathaniel Emmons

Datasets are not usually static, but they are usually treated as if they are. Data is often accumulated through time, such as stock markets, supermarkets, and bank transactions. The data is regularly updated, most typically through the addition of extra data, although other modifications can occur such as deleting existing data (for example, when goods are returned to a store, or correcting errors). We term these non-static datasets 'dynamic datasets', and believe that they are a key characteristic of many data mining problems.

In this chapter, we focus on the problem of mining frequent closed patterns in dynamic transaction datasets. It is a more challenging problem than that of static transaction datasets, since transactions in a target dynamic transaction dataset can be changed by dataset updates, and so frequent closed patterns and their support values in the dataset may be changed by updates as well.

The problem is generally dealt with by simply rerunning a static algorithm on the updated dataset as if it is completely new. This is likely to be inefficient, and misses out the ability to easily identify the changes in the data. There has been relatively little research in the area of dynamic pattern mining.

In this chapter, we explore 'incremental mining', where updates to the original dataset are mined as they are presented, and the results amalgamated with the frequent patterns mined from the original data. We propose a new incremental mining algorithm of frequent closed patterns that is an extension of CLOSET+ [120],

one of the most efficient and frequently used algorithms for mining frequent closed patterns in static transaction datasets.

This chapter is organized as follows. First, in Section 2.1 the background of the problem of mining frequent closed patterns in a dynamic transaction dataset is introduced, some preliminary concepts are given and the problem is formalized. The problem complexity is also analyzed in this section. After reviewing related work in Section 2.2, our proposed algorithm to solve this problem is presented in detail in Section 2.3, and then our algorithm is experimentally evaluated in Section 2.4. Finally, this chapter is summarized in Section 2.5.

2.1 Frequent Pattern Mining

Frequent pattern mining is the name given to the identification of repeated sequences of items in a dataset. It is one of the core tasks of data mining because of its many applications, such as discovering association rules [130]. As an example, a supermarket such as Pak'n Save may find that most customers who buy beer also tend to buy chips. Supermarket managers may use such information to decide promotional pricing or product layout.

Mining frequent patterns usually suffers from the problem of huge output, i.e., the huge number of frequent patterns that it produces. The number of frequent patterns may grow exponentially with respect to the user-specified threshold value, particularly in dense datasets with many long patterns[1]. Such huge output can include many redundant patterns, which wastes resources for later analysis, and makes it difficult for users to utilize the data [41, 130].

For example, suppose that a transaction dataset contains only two transactions: $\{\{a_1, a_2\}, \{a_1, a_2, a_3, a_4\}\}$ and the specified threshold (i.e., minimum support) value is 1. The complete set of frequent patterns in the dataset has size 15 ($= 2^4 - 1$): $\{\{a_1\} : 2, \{a_2\} : 2, \{a_1, a_2\} : 2, \{a_3\} : 1, \{a_4\} : 1, \ldots, \{a_1, a_2, a_3, a_4\} : 1\}$, where $\{a_{n_1}, a_{n_2}, \ldots, a_{n_{m-1}}, a_{n_m}\} : sup$ represents pattern $\{a_{n_1}, a_{n_2}, \ldots, a_{n_{m-1}}, a_{n_m}\}$ with its absolute support value sup. In the complete set, only *two* patterns $\{a_1, a_2\} : 2$ and $\{a_1, a_2, a_3, a_4\} : 1$ are non-redundant; the other 13 (and their correspondingly support values) can be derived from the two non-redundant patterns.

There are two methods of compressing the pattern data by storing smaller sets:

[1]The density of a given dataset can be judged by the average support of nodes in the FP-tree [120]. For a full definition, see [48, 62, 113].

lossy and lossless compression. The frequent *maximal* pattern set [57] is the most compact set possible, but it is lossy, in the sense that the exact support value of all items is not guaranteed to be recoverable. A frequent pattern is called maximal if there does not exist another frequent pattern that contains it. In the above example, the set of frequent maximal patterns is $\{\{a_1, a_2, a_3, a_4\} : 1\}$. Given only this set, the exact support value of some frequent patterns, such as $\{a_1\}$ and $\{a_1, a_2\}$, cannot be restored.

For cases where all support values could be needed, frequent *closed* patterns [131] can be used, which provide lossless compression. A closed pattern is one that has no super-patterns with the same support value. Though the compression ratio of mining frequent closed patterns is lower than that of mining frequent maximal patterns, the exact support value of every frequent pattern can be derived from the set of frequent closed patterns by finding the largest support value of their corresponding frequent closed super-patterns. In the above example, two patterns $\{a_1, a_2\} : 2$ and $\{a_1, a_2, a_3, a_4\} : 1$ are closed. Generally, the corresponding set of frequent closed patterns is a small portion of the complete set of frequent patterns.

Mining frequent closed patterns in static transaction datasets has been extensively covered in the literature reviewed in Section 2.2.1. In this context, a static transaction dataset commonly stores a huge collection of data that is all accessible, and the mining algorithm can be applied to it as a one-time task.

2.1.1 Dealing with Dynamic Datasets

If the dataset is not static, but consists of a base dataset following by a set of sequential updates over time, then whenever an update to the dataset is provided, new pattern analysis needs to be performed. There are two obvious methods to do this:

The rerunning method which ignores the previous mined results and rerun the chosen algorithm on the entire updated transaction dataset to mine the complete set of frequent closed patterns after a dataset update happens. Figure 2.1 illustrates the rerunning method.

The incremental method which stores the previously mined results and updates them based on mining only the updates, not the entire dataset. Figure 2.2 illustrates the incremental method.

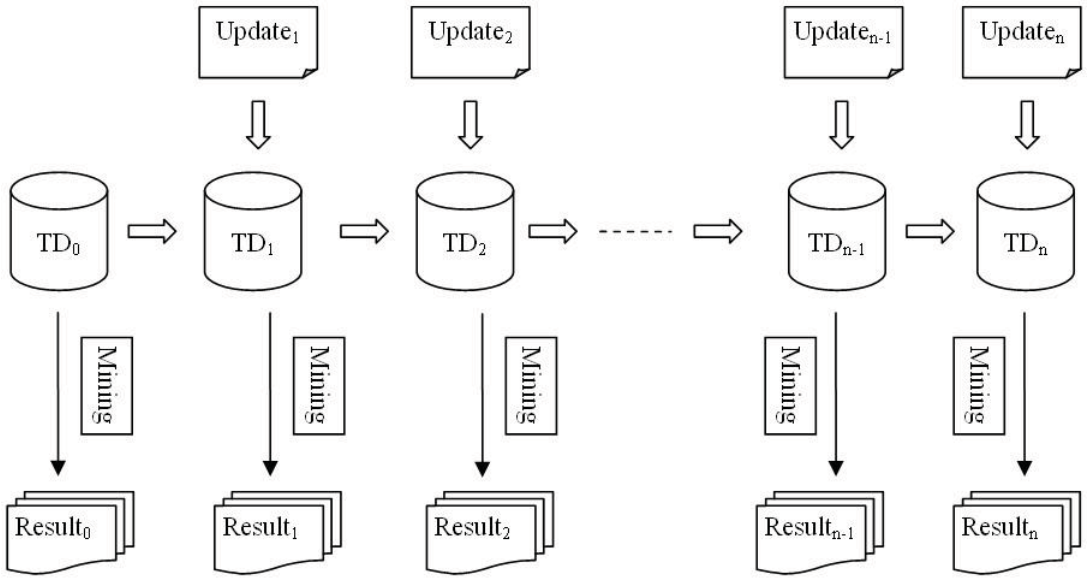


Figure 2.1: The rerunning method

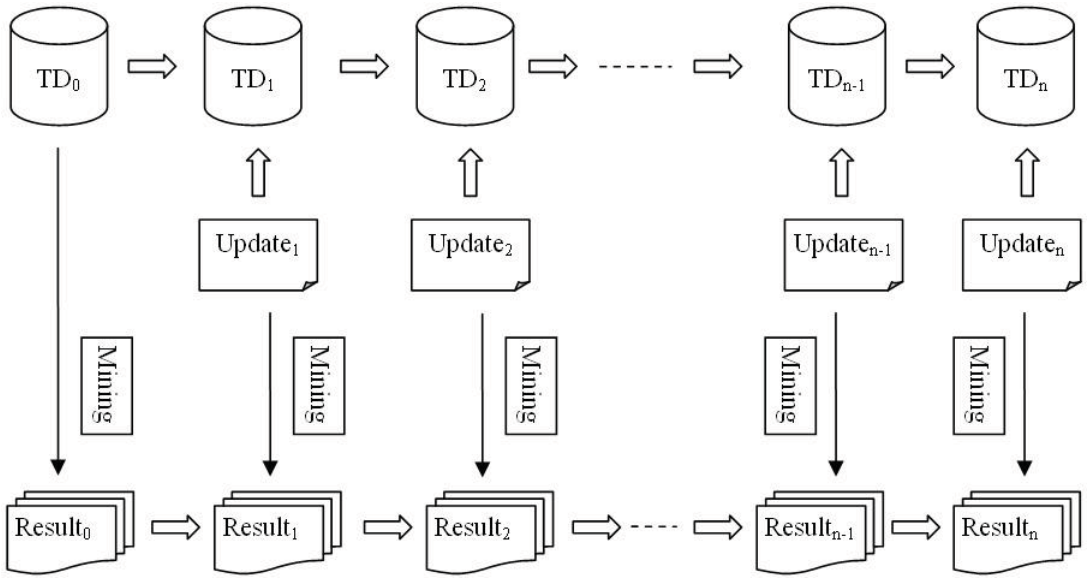


Figure 2.2: The incremental method

While the rerunning method is the most straightforward solution, it is also likely to be very slow for large datasets. It also requires that the entire dataset is stored. By requiring that only the updates to the dataset are mined, the incremental method requires less computational cost at each iteration, and does not require that the entire dataset is stored. The importance of this can be seen for the case of a large supermarket; the volume of daily sales, while large, is trivial compared with the accumulated data over the past year (or longer).

2.1.2 Preliminary Concepts and Problem Statement

The problem of mining frequent patterns was first introduced by Agrawal et al. [4] in 1993, and the problem of mining frequent closed patterns was first introduced by Pasquier et al. [92] in 1999. These works remain the standard references for all mining algorithms of frequent (closed) patterns.

Definition 2.1.1. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of possible *items* in a given dataset. Any item in I is distinct. An *itemset* or *pattern* z is an unordered subset of I, i.e., $z \subseteq I$. It can contain any item of I at most once.

As pattern z is unordered, the patterns $\{a, b, c\}$ and $\{c, a, b\}$ are equal. For ease of writing, patterns will be written as a series of continuous item identifiers and set brackets omitted. For example, pattern $\{a, b, c\}$ will be written as abc and pattern $\{b, d\}$ will be written as bd.

Definition 2.1.2. The *length* $|z|$ of a pattern z is the number of items in z, i.e., $|z| = \sum_{i \in z} 1$.

Definition 2.1.3. A pattern z is called a *sub-pattern* of another pattern z' if $z \subset z'$. Symmetrically, z' is called a *super-pattern* of z.

Definition 2.1.4. A *transaction* T is represented by a pair $\{TID, z\}$, where TID is a unique transaction identification number, and z is a pattern. The length of transaction T is denoted by $|T|$ and is equal to the length $|z|$, and is defined as $|T| = |z| = \sum_{i \in z} 1$.

Definition 2.1.5. A *transaction dataset* TD is a set of transactions. The number of transactions in TD is denoted by $|TD|$, and is defined as $|TD| = \sum_{T \in TD} 1$.

The term 'size of a transaction dataset' is sometimes taken to mean the number of individual items in the dataset. We avoid this term, which is potentially confusing.

Definition 2.1.6. The *absolute support* or *support* of a pattern z in a transaction dataset TD, which is denoted as $sup(z, TD)$ or $sup(z)$, is the number of transactions containing pattern z in TD, i.e., $sup(z) = \sum_{((T \in TD) \bigcap (z \subseteq T))} 1$. The *relative support* or *frequency* of z in TD, denoted as $freq(z, TD)$ or $freq(z)$, is defined as the absolute support value of z divided by the number of transactions in TD, i.e., $freq(z, TD) = \frac{sup(z,TD)}{|TD|}$.

Note that the term *support* has sometimes been used for the *relative support* in the literature.

Definition 2.1.7. Frequent patterns are determined by a pre-specified threshold value. The threshold value can be defined in terms of minimum frequency (denoted as min_freq) or *minimum support* (denoted as min_sup). Given min_freq or min_sup, a pattern z is said to be *frequent* in a transaction dataset TD if and only if the support (or frequency) value of the pattern z is not less than the minimum support (or minimum frequency) value specified by users, i.e., $sup(z, TD) \geqslant min\_sup = min\_freq \times |TD|$. Otherwise, the pattern z is said to be an infrequent pattern.

Note that in the literature, the term *minimum support* has sometimes been used for *minimum frequency* too, since in a static transaction dataset they have the same effect on the mining result. However, when applying them for mining in dynamic transaction datasets, they have different effects.

Definition 2.1.7 leads to the anti-monotone (downward closure) property [5] of frequent patterns, which is that all sub-patterns of a frequent pattern are also frequent. However, a super-pattern of a frequent pattern can be frequent or infrequent. Any super-pattern of an infrequent pattern is infrequent.

Definition 2.1.8. A pattern z is said to be *closed* in a transaction dataset TD if in TD there is no pattern z' satisfying $z' \subset z$ and $sup(z, TD) = sup(z', TD)$. Given min_freq or min_sup, if the closed pattern z is frequent in TD as well, the pattern z is said to be a *frequent closed pattern* (FCP) in TD.

Definition 2.1.9. A *dynamic transaction dataset* (DTD) is a transaction dataset where dataset updates happen. A dynamic transaction dataset is composed of two parts: the base transaction dataset TD_0 and a series of updates U_i $(i = 1, 2, \ldots, n)$. Each update can vary the data of DTD. Transaction dataset TD_n is the dataset after the n^{th} update of a dynamic transaction dataset, i.e.,

$$TD_n = TD_0 \bigcup \left(\bigcup_{i=1}^{n} U_i \right)$$

Mining in dynamic transaction datasets can face the following four different simple cases of dataset update[2]:

[2]Other more complex cases are made up of a combination of these four simple cases.

1. Adding new transactions Δ^+ that contain only items that existed in the original transaction dataset D before the update.

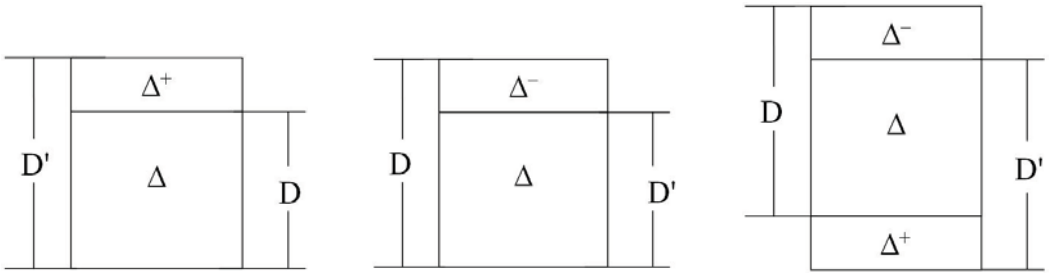
2. Adding new transactions Δ^+ that contain new items that do not exist in the original transaction dataset D.

3. Deleting a number of existing transactions Δ^- from the original transaction dataset D.

4. Modifying a number of existing transactions in D, which includes adding new items, deleting existing items, or replacing some existing items with different items.

These four simple cases are demonstrated in Figure 2.3, where D' denotes the updated transaction dataset after an update, and Δ denotes the *unchanged transactions* existing in both of D and D', i.e., $\Delta = D \cap D'$ during the update. Cases 1 and 2 are shown in Figure 2.3(a). Case 1 is a special case of Case 2, since Case 1 can be regarded as adding new transactions with *zero* new items. Case 3 is shown in Figure 2.3(b). Case 4 can be considered as the combination of Cases 1, 2 and 3. Note that in Case 4, the total number of transactions in the dataset generally remains the same, since the operations in Case 4 operate on the item level instead of the transaction level. However, there is an exception. In an extreme situation where the operation of deleting existing items removes all existing items from some transactions, the number of transactions in the updated dataset becomes less than that in the original dataset, i.e., $|D'| \leqslant |D|$. Figure 2.3(c) illustrates this. Note that in Cases 1 and 2, $D' = \Delta + \Delta^+ = D + \Delta^+$. In Case 3, $D' = \Delta = D - \Delta^-$, while in Case 4, $D' = \Delta + \Delta^+ = D - \Delta^- + \Delta^+$. As Cases 1 and 4 can be thought of special cases of Cases 2 and 3, we only consider Cases 2 and 3 in this chapter.

Given the *original transaction dataset* D before the update, the update of *adding new transactions* Δ^+ or *deleting existing transactions* Δ^- and the specified minimum frequency min_freq, the problem of incremental mining of frequent closed patterns in a dynamic transaction dataset is to mine the complete set of frequent closed patterns and their correspondingly exact support values in the *updated transaction dataset* D' after a update, by using the knowledge about D. The required knowledge of D is at least the complete set of frequent closed patterns, with their correspondingly exact support values in D.



(a) The updating case of adding new transactions

(b) The updating case of deleting existing transactions

(c) The updating case of modifying existing transactions

Figure 2.3: The simple cases of dataset update

Before we consider our incremental mining algorithm in more detail, we remark that, in common with full pattern mining, the problem is NP-hard, as is demonstrated by the following two propositions:

Proposition 2.1.10. *Updating the frequent closed pattern set is* NP-hard *when new transactions are added to the original dataset.*

Proof. Yang in his paper [124] indicated that the problem of counting the number of frequent closed patterns in a static transaction dataset with a specified arbitrary support threshold is #P-complete and the problem of mining (i.e., enumerating) frequent closed patterns is NP-hard. In the worst case, when the original dataset is empty and so the update is equal to the dataset, i.e., $D' = D$, the update reduces to the exact case considered by Yang. □

Proposition 2.1.11. *When an update consists only of deleting some of the existing transactions, the problem of incrementally mining frequent closed patterns is:*

1. *in* P *when the deleting update does not introduce new frequent patterns.*

2. NP-hard *when the deleting update introduces new frequent patterns.*

Proof. Let the complete set of frequent closed patterns in the original transaction dataset D be FCS. The length of the longest pattern in FCS is l_{FCS}, the number of patterns is $|FCS|$, the complete set of frequent closed patterns in the updated transaction dataset D' is FCS', and the length of the longest transaction in D' is $l_{D'}$.

1. When the deleting update does not bring new frequent patterns, FCS' must be a subset of FCS, i.e., $FCS' \subseteq FCS$. For each frequent closed pattern p in

FCS, when the deleting update reduces the support value of p, p can be closed by one of its super-patterns in FCS or even become infrequent; otherwise, p will certainly be in FCS'. Thus, using FCS as a candidate set C for computing FCS', we can first count the support value of each pattern in C by scanning D' once. This step takes at most $O\left(|FCS| \times l_{FCS} \times |D'| \times l_{D'}\right)$ time. Then, FCS' can be obtained by eliminating non-frequent and non-closed patterns in C, which takes at most $O\left(|FCS|^2 \times {l_{FCS}}^2\right)$ time. An algorithm that executes the above steps needs $O\left(\max(|FCS| \times l_{FCS} \times |D'| \times l_{D'},\ |FCS|^2 \times {l_{FCS}}^2)\right)$ time.

2. If and only if applying a minimum frequency value instead of a minimum support value as the threshold value to determine frequent patterns, a deleting update can bring new frequent patterns. When a deleting update does bring new frequent patterns, the problem of mining frequent closed patterns in a static transaction dataset D can be considered as a special case of that of incrementally mining frequent closed patterns in the updated transaction dataset after the update of only deleting some existing transactions from the original transaction dataset. That can be shown in the situation where the complete set of frequent closed patterns in the original dataset is empty, new frequent closed patterns appear after the deleting update, and the updated dataset D' is equal to D.

□

2.2 Related Work

This section reviews those existing algorithms that are most related to our work. These algorithms can be classified into two classes: mining in static transaction datasets, and mining in dynamic transaction datasets.

2.2.1 Mining Frequent Closed Patterns in Static Transaction Datasets

We survey three classes of algorithms for mining frequent closed patterns in static transaction datasets: the Apriori-like algorithms [92, 112], the pattern-growth algorithms [44, 48, 93, 120], and the vertical-format-based algorithms [74, 130, 133].

Apriori-like algorithms

The Apriori-like algorithms employ a generate-and-test strategy, i.e., first generating candidate patterns and then testing whether they are frequent by comparing their support values with the specified threshold value [5]. This relies upon the anti-monotone (downward closure) property that all sub-patterns of a frequent pattern must be frequent as well. Based on this property, Apriori-like algorithms attempt to level-wise optimize the process of mining frequent closed patterns.

The representative algorithm is A-Close, proposed by Pasquier et al. [92]. A-Close was the first mining algorithm of frequent closed patterns and has two main steps. In the first step, it exploits a bottom-up, level-wise (or breadth-first) process, which is similar to that taken by Apriori, to try to produce the set of minimum frequent generators. During the first step, multiple iterations are implemented. In each iteration, the candidate generators are first generated based on the frequent generators discovered in the previous iteration, then a scan over the input dataset counts the support of each candidate generator, and useless generators are pruned. The first step stops when no more generators are generated. The first iteration finds the set of frequent items. Each subsequent iteration k ($k \geqslant 2$) aims to find the set of the minimum frequent generators of length k. In the second step, A-Close determines the closure of the discovered minimum frequent generators to identify the complete set of frequent closed patterns.

As is pointed out in [56, 74], the Apriori-like algorithms generate lots of candidate patterns and scan the complete input dataset $(k + 1)$ times to count the support of each candidate pattern (where k is the length of the longest frequent pattern in the input dataset). This is not a trivial cost, especially as k can be a large number. The huge number of candidate patterns can also make the process of pruning useless patterns quite costly.

Pattern-growth algorithms

The pattern-growth algorithms adopt the divide-and-conquer paradigm. They often start with a short frequent pattern, and then grow the pattern in the transaction space by a depth-first search to discover the frequent closed patterns that take the short frequent pattern as their prefix. The transaction space is often represented by tree-like data structures. The key difference between Apriori-like algorithms and pattern-growth algorithms is that pattern-growth algorithms do not generate candidate patterns at all. Thus, pattern-growth algorithms can generally achieve better performance than Apriori-like algorithms.

CLOSET [93] and CLOSET+ [120] are typical algorithms belonging to this type. They use a tree-like data structure called a 'frequent pattern tree' (FP-tree) [47]. An FP-tree is an extension of the prefix tree [84]. It consists of a *frequent-item header table*, and a set of *item prefix subtrees* with the common root labelled as "null". Each entry in the frequent-item table has two fields: item-name and head of node-link. Each item-name shows which frequent item this entry represents, and its associated head of node-link points to the first node with this item-name in the item prefix subtrees. All frequent items are counted, sorted and stored in support descending order in the header table. Each node of item prefix subtree has three fields: item-name, count and node-link. Each count records the number of transactions sharing the path from the root to this node, while each node-link points to the next node with the same item-name in item prefix subtrees, or null if there is none. Every branch in the item prefix subtrees represents the list of ordered frequent items in a transaction of the input dataset. The ordering is based on the sequence of frequent items in the header table. Thus, an FP-tree can compactly store all frequent items in each transaction of the input dataset in main memory.

For instance, suppose that we have a transaction dataset TD given in the first two columns of Table 2.1 and the specified minimum frequency value is 40% (i.e., $min\_freq = 40\%$). The list f_list of sorted frequent items in TD is $f\_list = \{f : 4, c : 4, a : 3, b : 3, m : 3, p : 3\}$ in descending order of support value. The frequent items in each transaction are sorted based on f_list and listed in the third column of Table 2.1. The FP-tree based on TD is shown in Figure 2.4.

TID	List of Items	Ordered Frequent Items
100	a, f, c, m, b	f, c, a, b, m
200	b, j, f	f, b
300	m, f, p, e, a, c	f, c, a, m, p
400	f, p, c, m, i, a	f, c, a, m, p
500	b, s, c, p	c, b, p

Table 2.1: A transaction dataset with ordered frequent items

Experiments have shown that the reduction ratio for an FP-Tree (compared with the dataset) can reach 165.04 [47]. Generally, the reduction ratio is higher for datasets with more long transactions. The longer the patterns are, the higher the probability that they share prefixes. If they do share a prefix, the longer the patterns are, the higher the probability that the prefix is long. FP-trees store a shared prefix

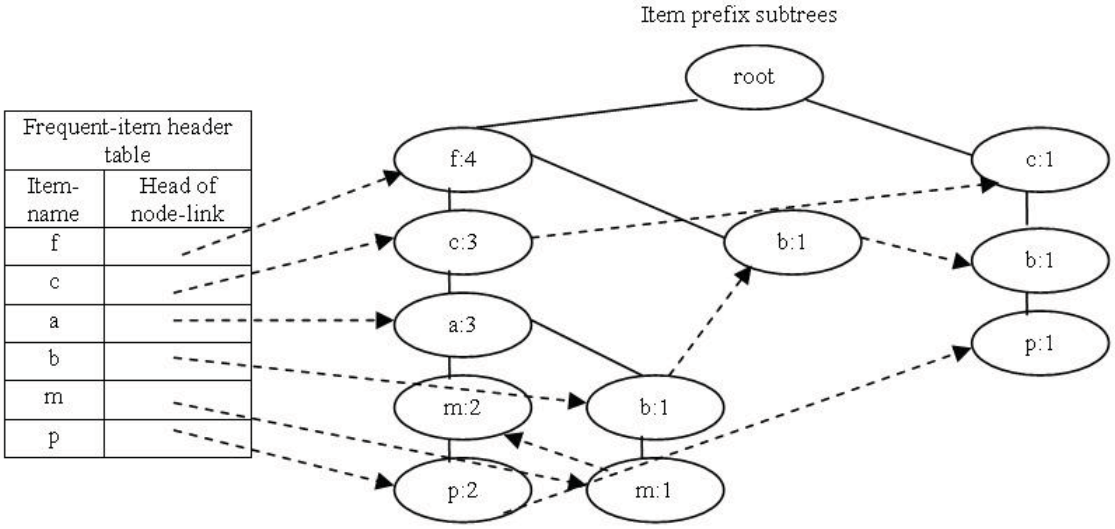


Figure 2.4: The FP-tree based on TD in Table 2.1 when $min\_freq = 40\%$

only once, which is why they can achieve such a high reduction ratio.

Instead of generating and testing candidate patterns, pattern-growth algorithms search the constructed FP-tree in depth-first order to mine frequent closed candidate patterns. During the searching process, the input dataset is not required. After a candidate is mined, they check the closure of the candidate against the previously mined frequent closed patterns to determine whether the candidate is a frequent closed pattern and the previously mined frequent closed patterns are still closed. Combined with the mechanism of partition-based projection, they reduce the memory space cost and enhance the efficiency of the mining process. More details of CLOSET+ are given in Section 2.3.2. To date, CLOSET+ is one of the most popular and powerful algorithms for mining frequent closed patterns in static transaction datasets. In addition, its principal scheme and data structures are suitable to be extended to apply to dynamic transaction datasets. For those two reasons, it is chosen as the base of our proposed algorithm.

Vertical-format-based algorithms

In contrast to the two kinds of algorithm mentioned above, vertical-format-based algorithms focus on a vertical format representation of transaction datasets, where each item is associated with a transaction ID set (TID_set) listing where the item occurs. This vertical format makes counting the support of patterns simple and quick. For example, suppose that we have $TID\_set(a) = \{1, 2, 3, 4\}$ and $TID\_set(d) = \{2, 5\}$ in Table 2.2, then $TID\_set(ad) = \{1, 2, 3, 4\} \cap \{2, 5\} = \{2\}$.

Therefore, the support of pattern *ad* is 1.

Item	Set of transaction IDs
a	1, 2, 3, 4
b	1, 4
c	3, 4
d	2, 5
e	2, 3, 4

Table 2.2: A vertical-format-based transaction dataset

The characteristic algorithm is CHARM [130, 133], which performs a bottom-up, depth-first search strategy on a data structure called the itemset-tidset tree (*IT-tree*) to improve the efficiency of enumerating frequent closed patterns. CHARM generates a frequent pattern each time, and then compares its TID_set with those of the other frequent patterns sharing the same prefix. If a TID_set subsumes another, the associated patterns belong to the same equivalence class and the corresponding nodes of IT-tree are merged. CHARM utilizes a vertical data representation called *diffsets* [132] to store the TID_set associated with the pattern represented by a node. Diffsets record the difference in the TID_set of a pattern from its parent pattern. Thus, diffsets can reduce the memory space required for storing intermediate results and quickly compute the support of patterns. CHARM also applies a hash-table-based approach to eliminating any non-closed frequent patterns discovered from different paths during the closure checking.

All of the algorithms mentioned above are designed to mine frequent closed patterns in static transaction datasets. However, directly applying them to dynamic datasets is not efficient, since they ignore the information discovered previously. Whenever an input dataset is updated, they must restart the mining process from the beginning.

2.2.2 Mining Frequent Patterns in Dynamic Datasets

The process of mining frequent closed patterns in dynamic transaction datasets is more complicated than that of static transaction datasets [21, 105], since dataset updates can introduce new frequent patterns and invalidate some previously mined ones, as was described previously. Some severe problems, such as efficiently checking the status of those frequent closed patterns in the previously mined results after an update happens, only appear when mining in dynamic transaction datasets. Most

incremental mining algorithms are based on algorithms for mining in static datasets.

Apriori-like incremental mining algorithms

The first work on incremental mining seems to be Cheung et al. [24], who applied the incremental method in the mining process, proposing an Apriori-like incremental mining algorithm called the Fast Update Algorithm (FUP) to update the frequent patterns in an input dataset when new transactions are added to the dataset.

The major idea of FUP is to use the previously mined frequent patterns and their supports to generate a relatively small set of candidate patterns that need to be re-examined. FUP has two main steps. One step is to remove the infrequent patterns from the set of frequent patterns mined before the update by checking their supports against the newly added transactions from the update. The other is to mine the newly emerging frequent patterns in the updated dataset. The candidate patterns and their supports are first computed from the newly added transactions of the update, and then the supports of those candidate patterns are updated and checked against the original input dataset to discover new frequent patterns.

Compared with rerunning Apriori [5], FUP has a smaller number of candidate patterns that need to be checked against the updated dataset. The experimental results [24] also shows that FUP improves performance significantly compared with rerunning Apriori and DHP [90] on the updated dataset.

Later, Cheung et al. [23] developed the FUP_2 algorithm to extend FUP for the situation that transactions are added and deleted from a dynamic transaction dataset. Basically, FUP_2 is equivalent to FUP plus the algorithm dealing with the deletion update. It is noted that if an infrequent pattern is frequent in the transaction deleted during a deleting update, it must be infrequent in the unchanged transactions. FUP_2 utilizes this observation to reduce the number of candidate patterns to be checked against the unchanged transactions.

The major weaknesses of FUP and its variations [23] is that they are Apriori-like algorithms. This leads to repeat scans of the dataset once the dataset has been updated, and the number of scans they require is equal to the length of the longest frequent pattern in the updated dataset. These algorithms still suffer from the large number of candidate patterns to be checked. To overcome the weaknesses, two papers [39, 116] suggest using the negative border [117] to indicate when the mined results need to be updated. The negative border of the frequent patterns in a dataset indicates the borderline between the frequent and infrequent patterns. It is a collection of all infrequent patterns whose sub-patterns are all frequent. The

patterns in it can be obtained by using a level-wise mining algorithm such as Apriori. Like FUP, negative-border-based algorithms maintain and use previously-discovered frequent patterns. In addition, they also need to maintain and use the infrequent patterns in the negative border.

If an infrequent pattern becomes frequent after an update, it should have a sub-pattern that is in the negative border generated before the update and becomes frequent after the update[3] [116]. Hence, if this is not found, there is no need to scan the dataset to look for newly emerging frequent patterns. By relying on the negative border, the incremental updating algorithm [116] only needs to scan the whole dataset if the dataset update brings new infrequent patterns to the negative border.

The key advantage of negative-border-based algorithms [8, 105, 116] is that scanning the original dataset is rarely required. Thus, they are more efficient than FUP. Like FUP, negative-border-based algorithms maintain and use the previously-discovered frequent patterns. In addition, they also need to maintain and use the infrequent patterns in the negative border. Thus, the total number of patterns maintained can be huge, which is memory consuming.

Ayan et al. [9] presented an algorithm called UWEP (Update With Early Pruning) that employed a dynamic look-ahead strategy. This strategy was used in updating the existing frequent patterns by detecting and removing infrequent patterns when new transactions were added to the original dataset. It removes the super-patterns of an infrequent pattern that is frequent in the original dataset as soon as it becomes infrequent in the updated dataset.

Relying on this strategy, UWEP minimizes the number of candidate patterns that need to be checked in the updated dataset. Another advantage of UWEP is that it scans the original dataset at most once and the updated dataset exactly once. Consequently, UWEP enhances the efficiency of FUP. However, to limit the number of scans over the dataset to one, UWEP requires large memory to hold all *tidlists*, and the set of candidate patterns. Each tidlist associated with an item is an ordered list of transaction IDs showing where the item exists.

Partition-like incremental mining algorithms

Similar to Apriori-like algorithms, Partition-like algorithms also employ the generate-and-test strategy. However, Partition-like algorithms generate candidate patterns

[3]Here it is assumed that every item is either frequent or in the negative border.

in a different way. The Partition algorithm was proposed by Savasere et al. [106] in 1995 to mine frequent patterns in static transaction datasets by using a partition-based heuristic [66], i.e., given a dataset divided into a number of non-overlapping partitions, if a pattern is frequent in this dataset, it must be frequent in at least one of these partitions. Partition-like algorithms process each partition of an input dataset iteratively to discover local frequent patterns of each partition, and then use them as candidate patterns against the whole dataset to identify global frequent patterns in the input dataset.

A typical Partition-like incremental mining algorithm is the sliding-window filtering (SWF) algorithm [65, 66]. SWF is often adopted in mining data streams as a special kind of dynamic dataset, which mines the data within a fixed period of time rather than all of the past data. That is, when the time window with a fixed length slides, SWF handles new data moving into the window, and obsolete data moving out from the window.

SWF first partitions a transaction dataset into several sequential, non-overlapping partitions, and then uses filtering thresholds related to the minimum support to process each partition one by one. After processing a partition, local frequent 2-patterns are obtained. The previously-discovered local frequent 2-patterns are selectively passed to the process in the following partitions as candidate 2-patterns. After all partitions are processed, SWF uses the identified candidate 2-patterns to generate other candidate patterns. With the second scan over the whole dataset, the frequent patterns in the dataset can be discovered.

Besides maintaining the previously mined candidate 2-patterns and their supports (in common with other kinds of incremental mining algorithms), SWF also maintains the starting partition of each candidate pattern. The starting partition associated with a candidate pattern indicates the partition where the pattern is first identified. When the time window slides, the starting partition associated with candidate patterns allows SWF to easily update the candidate patterns affected by the obsolete transactions. The newly-added transactions can be handled by considering them as a partition. Though SWF eliminates lots of unqualified candidate patterns in the early stage of the mining process, it can still generate unqualified candidate patterns. Moreover, SWF ignores the frequent patterns previously mined from the original dataset before an update. Consequently, to update the supports of the previously mined frequent patterns after an update, SWF has to scan the whole updated dataset instead of the changed partitions of the dataset. Obviously, this is

not efficient.

Pattern-growth incremental mining algorithms

Generally, pattern-growth incremental mining algorithms use tree-like data structures that compactly maintain the input dataset and previously mined patterns in memory in order to avoid generating any candidate patterns and multiple scans over the dataset.

Ezeife et al. [36] proposed the DB-tree and PotFP-tree data structures, which are extensions of the FP-tree data structure. Based on these two special data structures, the required number of dataset scans for incrementally mining frequent patterns in the updated dataset is reduced. To mine frequent patterns, the DB-tree and PotFP-tree algorithms [36] first need to extract an FP-tree from either a DB-tree or PotFP-tree. The DB-tree is used to maintain the input dataset. Thus, the DB-tree algorithm does not require any scan over the original dataset after the update. However, since a DB-tree holds the whole dataset, the DB-tree algorithm may require a lot of memory. Unlike an FP-tree, which stores only frequent items in the dataset and the DB-tree, which stores all items, the PotFP-tree is used to maintain the frequent items and the infrequent items whose supports are at least that of a tolerance parameter value. Therefore, the need to scan the original dataset in order to update the FP-tree after the update is reduced. In essence, the PotFP-tree algorithm uses a tolerance parameter to balance the required memory and the number of scans of the dataset. These two algorithms need to scan the update at most twice before generating any candidate patterns.

Algorithm Moment (Maintaining Closed Frequent Itemsets by Incremental Updates) was designed by Chi et al. [26] to mine frequent closed patterns over a data stream sliding window. They proposed a data structure called CET (closed enumeration tree) to maintain a set of patterns dynamically selected in the sliding window. The set of selected patterns includes all frequent closed patterns and the patterns forming the boundary between the frequent closed patterns and others. Since any pattern that changes from frequent to infrequent or vice versa can affect the boundary, the CET has all necessary information to mine frequent closed patterns in the sliding window. When the window slides, Moment updates the supports of nodes in the CET and may modify the CET as well. For any newly created node in the CET, Moment needs to scan all transactions in the current window to count its support.

The average update cost of the CET relies on the stability of the boundary maintained by the CET. As long as the boundary is relatively stable, the average

update cost of the CET stays small. However, the CET may have a huge number of nodes, including a lot of infrequent patterns, especially when the size of the sliding widow is large, which is memory-consuming and inefficient. Another limitation of Moment is that it was only designed to handle updates with one transaction.

Some data structures, such as the CATS Tree (Compressed and Arranged Transaction Sequences Tree) [25], the CanTree (canonical-order tree) [67] and the CP-tree (Compact Pattern tree) [114], are proposed as extensions of the idea of the FP-tree. Any one of these three tree structures can be constructed with only one scan of a target dataset. A CATS Tree uses a complex tree construction process to store the items of every transaction in a target dataset according to a descending order of local item frequency. A CanTree stores the items of every transaction in a target dataset according to some canonical order, which is predefined by users before the beginning of a mining process. A CP-tree stores the items of every transaction in a target dataset according to a descending order of item frequency in the CP-tree by processing part-by-part the dataset and reordering the CP-tree periodically based on the processed part of the dataset. A CP-tree begins to be built for some transactions in a user-predefined item order. During the building process of a CP-tree, the frequency of each item is found from the CP-tree built. If the descending order of item frequency in the CP-tree is changed, the CP-tree will be reordered based on the latest order.

However, the memory space required by a CATS Tree and a CanTree is more than the one required by a corresponding FP-tree, since a FP-tree generally is the most compact tree representation of a target dataset. Although a CP-tree requires the same amount of memory space as a FP-tree, the efficiency of constructing a CanTree suffers from the reordering processes required. Moreover, none of these three data structures provide the corresponding mechanism of using previously-mined results. That can affect the efficiency of mining algorithms simply based on these three data structures, when dealing with dynamic datasets.

2.3 Algorithm Design and Implementation

We propose a new algorithm for mining frequent closed patterns in dynamic transaction datasets. The main idea behind our proposed algorithm is to utilize the incremental method to extend CLOSET+, which is one of the most efficient algorithms for mining frequent closed patterns in static transaction datasets, to handle

the mining tasks in dynamic transaction datasets. By compactly maintaining the information discovered previously, our proposed algorithm can mine the frequent closed patterns in the updated dataset without rescanning the entire transaction dataset after the dataset is updated.

Our proposed mining algorithm consists of two key procedures: the pre-processing procedure and the incremental update procedure. The pre-processing procedure mines frequent closed patterns in the original transaction dataset D by creating a DB-tree and a result tree before any update happens. The incremental update procedure mines frequent closed patterns in the updated transaction dataset D' by incrementally modifying the previously mined frequent closed patterns based on only the information of the update. The pre-processing procedure is only required once to obtain the frequent closed patterns in the original dataset as a part of the previously-discovered knowledge for the following mining processes. The incremental update procedure is used each time that the dataset is updated.

In fact, the pre-processing procedure is our implementation of CLOSET+ [120], and is also the version used as the comparator in Section 2.4. The key difference between the pre-processing procedure of our proposed algorithm and CLOSET+ [120] is that our proposed algorithm uses a DB-tree instead of the FP-tree used in CLOSET+.

Note that the purpose of our study in this chapter is to explore how to apply the incremental method in the existing algorithms designed for mining frequent closed patterns in static transaction datasets in order to extend them to efficiently mine dynamic datasets. Thus, we only implement the principal data structures and techniques of CLOSET+ in the pre-processing procedure and the incremental update procedure. Adopting other unimplemented techniques of CLOSET+, or even other algorithms, could improve the performance of both procedures, which can be further explored. Our proposed algorithm could serve as the core procedure and the platform for adopting other techniques.

2.3.1 Data structures

Two tree-like data structures are employed in our algorithm: the DB-tree [36] and the two-level hash-indexed result tree [120]. The DB-tree maintains the target transaction dataset in order to reduce the time of scanning the dataset. It is a variant of the FP-tree [47] and has the same structure. However, an FP-tree only stores frequent items of each transaction in the target dataset, while a DB-tree stores all

items in each transaction, whether they are frequent or not. Hence, a DB-tree has more nodes and branches than the corresponding FP-tree and requires more memory space. Since the DB-tree and FP-tree use the same order to store items, the top part of a DB-tree is exactly the corresponding FP-tree.

Similarly to the FP-tree, a DB-tree needs to store all items in the transaction dataset in the header table in descending order of the support. Hence, scanning the original transaction dataset D twice is required to construct a DB-tree. The first scan is to get the set of items and their corresponding support values in the original dataset, while the second scan is to sort the items of each transaction and insert them into the DB-tree in the descending order of support value. After all the transactions have been inserted into the DB-tree, the DB-tree is ready to be used to search for the candidate patterns of frequent closed patterns in the original dataset.

The two-level hash-indexed result tree is for storing the mined frequent closed patterns and closure checking. Its structure is similar to that of the FP-tree. Besides a tree data structure for storing the current mined frequent closed patterns, it has a two-level hash-indexed header table for checking the closure of patterns. Each entry in the first-level hash-indexed header table has two fields: itemID and table-link. Each itemID stores the name of the last item shared by a group of frequent closed patterns as a hash key, and its associated table-link points to the corresponding second-level hash-indexed header table. Entries are stored in descending order of support value of frequent items.

In the second-level hash-indexed table, each entry has two fields: support and node-link. Each support field has the support value of a frequent closed pattern as a hash key, and its associated node-link points to the last item of a frequent closed pattern stored in the result tree. Entries are sorted in descending order of support value of the stored frequent closed patterns.

Each node in a result tree has four fields: itemID, support, the length and the node-link. The itemID of a node records the name of the item shared by some frequent closed patterns. Its associated support field records the maximum support of those patterns sharing this node. The associated length is to record the length of the path from this node to the root node labelled as “null”. The node-link points to the next node with the same itemID if the frequent closed patterns represented by these two nodes have the same support value, or null if there is none. The items in each mined frequent closed pattern are sorted and inserted in the result tree in descending order of support value of frequent items. Note that the sorting orders

used in the FP-tree, the DB-tree and the result tree are the same.

2.3.2 The Pre-processing Procedure

The pre-processing procedure searches the DB-tree to mine frequent closed candidate patterns after a DB-tree has been constructed based on the original transaction dataset. As soon as a frequent closed candidate pattern is mined from the DB-tree, its closure needs to be checked against the previously mined frequent closed patterns stored in the two-level hash-indexed result tree, in order to determine whether the candidate is closed. Generally, the closure check involves superset-checking and subset-checking. The superset-checking is to check whether a frequent closed candidate pattern is a super-pattern of some previously mined frequent closed pattern with the same support value, while the subset-checking is to check whether a frequent closed candidate pattern is a sub-pattern of a previously mined frequent closed pattern with the same support value.

The closure check in the pre-processing procedure only requires subset-checking, as in CLOSET+ [120], since the pre-processing procedure works under the divide-and-conquer and depth-first-search framework.

Some features of closed patterns are indicated by Wang et al. [120]. If a pattern z is a sub-pattern of a pattern z', they must satisfy: 1) $sup(z) = sup(z')$; 2) $|z| \leqslant |z'|$; and 3) z' should contain all items in z. Based on the structure of the two-level hash-indexed result tree, subset-checking can be efficiently implemented by applying these features. Suppose that a frequent closed candidate pattern has been mined from the DB-tree. The items in this candidate are listed in descending order of support value of frequent items. By using the itemID of the last item in the pattern and the associated pattern support value as hash keys and following the corresponding links in the header table of the result tree, the nodes with the same itemID and support value can be found if they exist in the result tree. Then the length feature of closed patterns can be checked by using the values of the length field of the found nodes. Eventually, the closure of the candidate can be checked by following the paths from the found nodes to the root. If the candidate can pass all three checks, it is a frequent closed pattern and is inserted into the result tree.

Lemma 2.3.1. *If a pattern z' exists in every transaction containing a pattern z, then $z \cup z'$ forms a closed pattern if it is not a proper sub-pattern of some closed pattern with the same support value.*

Lemma 2.3.2. *Let pattern z be a frequent closed candidate pattern and pattern z' be a previously mined frequent closed pattern. If $z \subset z'$ and $sup(z) = sup(z')$, z and all of z's descendants in the FP-tree cannot be frequent closed patterns and there is no need to search them.*

Lemmas 2.3.1 and 2.3.2 present two common search space pruning techniques [20, 57, 93, 120, 133] that are also utilized in our proposed algorithm. Lemma 2.3.1 is often used to handle the single prefix path in a mining FP-tree to generate a frequent closed candidate pattern with maximal length. Lemma 2.3.2 is used after checking the closure of a mined candidate pattern.

The pre-processing procedure is outlined below:

Input: A transaction dataset D and a minimum frequency value min_freq.

Output: A DB-tree based on D and a two-level hash-indexed result tree based on the DB-tree.

Method:

1. *Create an empty DB-tree and insert items into the header table. Scan the transaction dataset D once to collect a list L of all items in D and also count the support of each item. Then, sort the items in L and store the sorted items in the header table of the DB-tree in descending order of support.*
2. *Insert transactions into the DB-tree. Scan D once transaction by transaction. Sort the items of each scanned transaction and insert the sorted items into the DB-tree in descending order of item support.*
3. *Create an empty result tree and insert frequent closed patterns mined from the DB-tree into the result tree. Identify the frequent items in the header table of the DB-tree. Based on these frequent items and their support values, sequentially and iteratively project the DB-tree into a number of small FP-trees by using the divide-and-conquer paradigm. Use depth-first search to mine these FP-trees in the top-down manner to discover the frequent closed candidate patterns with at least minimum frequency value min_freq. When a frequent*

closed candidate pattern is found, apply subset-checking. If the candidate is a sub-pattern of a pattern stored in the result tree and they have the same support value, the candidate is not closed and the candidate and all of its descendants in the DB-tree can be abandoned for searching frequent closed patterns. Otherwise, the candidate is inserted into the result tree.

4. *Stop when no more frequent closed patterns can be found from the DB-tree.*

The cost of inserting a transaction T into a DB-tree is $O(|T|)$, while the cost of inserting a pattern z into a result tree is $O(|z|)$. The total number of nodes in the DB-tree is not more than the size of D plus 1, and the total number of nodes in the result tree is not larger than that in the DB-tree.

Given the dataset in Table 2.1, the DB-tree built by the pre-processing procedure is shown in Figure 2.5. Supposing that the specified minimum frequency value is 40% (i.e., $min\_freq = 40\%$), the two-level hash-indexed result tree built by the pre-processing procedure is given in Figure 2.6.

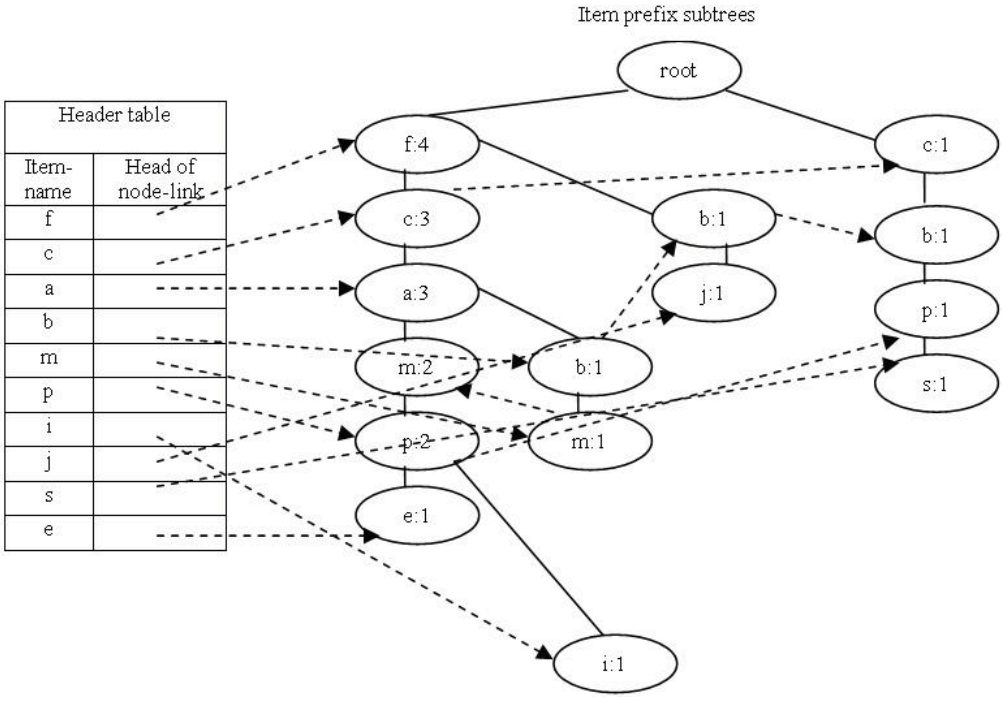


Figure 2.5: The DB-tree based on TD in Table 2.1

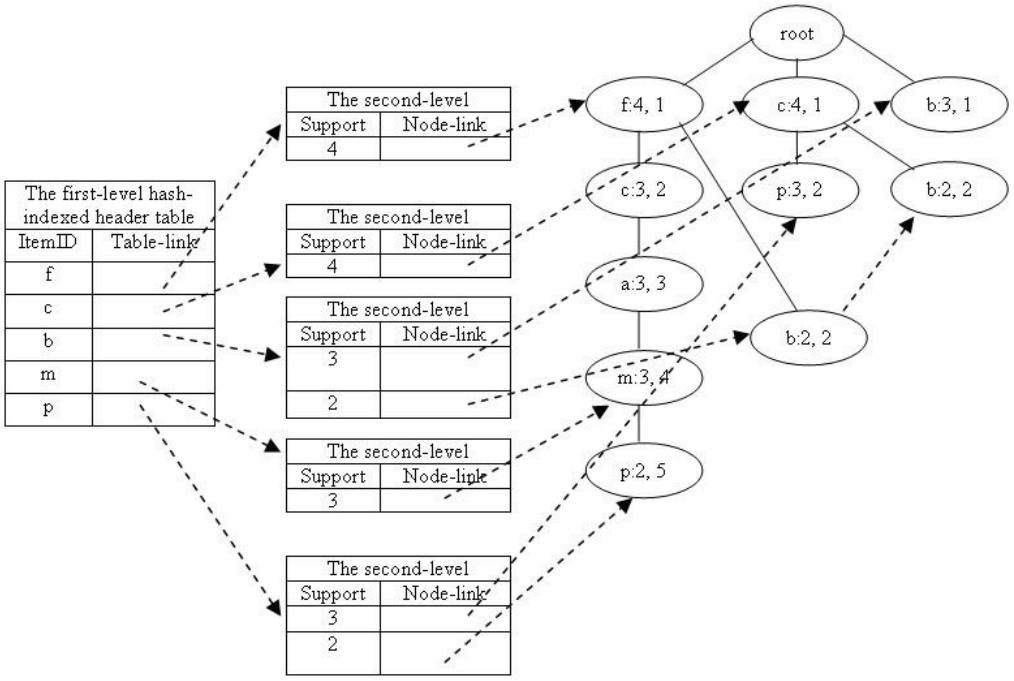


Figure 2.6: The two-level hash-indexed result tree based on TD in Table 2.1 when $min\_freq = 40\%$

2.3.3 The Incremental Update Procedure

After the pre-processing procedure has been used to create the DB-tree and the two-level hash-indexed result tree, when a dataset update is presented, the incremental update procedure first updates the DB-tree based on the dataset update, and then updates the result tree based on the changes in the DB-tree to avoid unnecessary operations in generating candidate patterns and checking closure.

Reordering the DB-tree and the result tree may be required after each dataset update. During each update, the support of each item can be increased or decreased. Based on the structure and building process of the DB-tree, all frequent items in a DB-tree are listed at the top of the tree in descending order of support value. Thus, a dataset update may lead to changes in the position of items in the header table of the DB-tree and the result tree, and the sorting order of items when inserting a transaction or a pattern into the DB-tree and the result tree. Both the DB-tree and the result tree need to be reordered only when the position of any frequent item in the header table of the DB-tree is changed by updates. The purpose of the reordering is to try to reduce the memory consumption and enhance the performance of the incremental update procedure. Compared with the position change of frequent items, the position change of infrequent items in the header table affects this pur-

pose relatively little. Considering the number of infrequent items and the cost of reordering the DB-tree and the result tree, the incremental update procedure does not reorder the DB-tree and result tree if only the position of infrequent patterns has changed.

After a dataset update, the incremental update procedure scans the update once to update the header table of the DB-tree by updating the support of items, deleting items with zero support value, and inserting new items at the bottom of the header table. Then all items in the header table are listed in descending order of support value in an item list i_list. Comparing the sequence of items in i_list with that in the header table of the DB-tree, if the position of a frequent item in i_list is different to its position in the header table, the incremental update procedure will reorder the DB-tree and the result tree.

When reordering a DB-tree, the frequent items that have different positions in i_list are identified in the header table of the DB-tree. The first (fci) and last (lci) frequent items with changed position are found from these identified frequent items in the header table, so that all other frequent items with changed position are between fci and lci in the header table. Note that the item sequence before fci or after lci in the header table is not affected by the update. Thus, only the node representing an item that is between fci and lci in the header table of the DB-tree may need to be reordered in its associated paths.

The reordering process traverses each item from fci to lci in the header table to locate the paths associated with such items through node-links. In each located path, the sub-path that involves the nodes representing the items between fci and lci in the header table is identified, and then the node sequence of the sub-path is checked. If the node sequence follows the item sequence in i_list, the path does not need to be reordered and the reordering process moves onto the next located path. Otherwise, the sub-path and its sub-tree are extracted from the path. After the nodes in the sub-path are sorted based on the item sequence in i_list, the reordered sub-path and its sub-tree are inserted back into the path where they were extracted from. Then, the reordering process checks the next path. After every such path has been checked, the items between fci and lci in the header table are sorted based on the item sequence of i_list. Now, reordering the DB-tree is complete. Note that the sequence of infrequent items in the reordered header table of the DB-tree may not be in descending order of support, but this does not matter, as was explained before. However, all frequent items in the reordered header table are definitely listed at the

top of the header table in descending order of support value.

The result tree needs to be reordered if and only if the DB-tree has been reordered, so that the item sequences used for the DB-tree and the result tree are always the same. However, the process for reordering the result tree is a little bit different from the one for the DB-tree. Every node in a DB-tree is linked by a node-link from the header table of the DB-tree. Only the nodes representing the last items of frequent closed patterns in a result tree are linked by node-links of the header table in the result tree. Thus, if the reordering process only travels from fci to lci in the header table of the result tree to locate the paths, the process can miss some paths. A solution to this problem is to travel along all frequent items before lci in the header table of the result tree in order to locate all paths that are associated with the frequent items between fci to lci. This is the key difference between the reordering processes for the DB-tree and the result tree. After all located paths are checked or reordered, the frequent items in the header table of the result tree are sorted based on the item sequence in the header table of the DB-tree.

Extra node-links are utilized to avoid rechecking the sub-paths that have been previously checked or reordered during the reordering process. A node in any previously checked or reordered sub-path is removed from the node-link of the header table and inserted into a corresponding additional node-link, which links all nodes with same item-name in the previously checked or reordered sub-paths together. With the reordering process, the node-links in the header table, which are associated with items between fci to lci, become shorter and the additional node-links become longer. After all paths associated with items between fci and lci have been checked and reordered, each additional node-link replaces its corresponding node-link in the header table. The cost of reordering a path is $O((\log(|path|) + 2)|path|)$, where $|path|$ is the number of nodes in the path.

For example, suppose an update to the original dataset D shown in Table 2.1 is the addition of a new transaction with $TID = 600$ into the dataset. The updated dataset D' is given in the first two columns of Table 2.3. With the same minimum frequency value 40%, the list of sorted frequent items in the updated dataset becomes $f\_list = \{f : 5, c : 4, m : 4, a : 3, b : 3, p : 3\}$ in descending order of support value. The items in each transaction are sorted in descending order of support value and listed in the third column of Table 2.3.

The item sequence in the header table of the DB-tree before the dataset update is $i\_sequence = \{f : 4, c : 4, a : 3, b : 3, m : 3, p : 3, i : 1, j : 1, s : 1, e : 1\}$,

TID	List of Items	Ordered Items
100	a, f, c, m, b	f, c, m, a, b
200	b, j, f	f, b, j
300	m, f, p, e, a, c	f, c, m, a, p, e
400	f, p, c, m, i, a	f, c, m, a, p, i
500	b, s, c, p	c, b, p
600	f, m, j, d	f, m, j, d

Table 2.3: An updated transaction dataset with ordered items

as was shown in Figure 2.5. After updating the header table, *i_sequence* becomes $\{f:5, c:4, a:3, b:3, m:4, p:3, i:1, j:2, s:1, e:1, d:1\}$. It is clear that the item sequence in *i_sequence* is not in descending order of support any longer. All items in *i_sequence* are listed in descending order of support value in *i_list*. Thus, $i\_list = \{f:5, c:4, m:4, a:3, b:3, p:3, j:2, i:1, s:1, e:1, d:1\}$.

Reordering the DB-tree is required, since the positions of frequent items a, b and m are changed in *i_list* compared with *i_sequence*. The first and last frequent items with changed position are a and m respectively. The paths, which may need to be reordered, can be located by travelling along a, b and m in the header table. For example, the reordering process can identify the paths associated with a from the DB-tree shown in Figure 2.5. Figure 2.7 gives these identified paths. In these paths, the sub-paths, which only involve the nodes representing a, b and m, need to be checked. A checked sub-path and its sub-tree are shown in Figure 2.8.

Since the node sequence in the sub-path in Figure 2.8 does not follow the item sequence in *i_list*, the sub-path and its sub-tree are extracted from the paths in Figure 2.7. This step is shown in Figure 2.9. After the nodes in the extracted sub-path are sorted based on the item sequence in *i_list*, the reordered sub-path and its sub-tree are inserted back into the path they were extracted from. This step is shown in Figure 2.10. Now, the reordering process can move to the next path associated with a and check its sub-paths.

The result tree associated with the DB-tree shown in Figure 2.5 needs to be reordered, since the DB-tree has been reordered. The process for reordering the result tree shown in Figure 2.6 starts by locating the paths associated with the frequent items before m in the header table of the result tree. The paths associated with f are located as shown in Figure 2.11. After reordering the sub-path involving only the nodes representing a, b and m, the reordering process inserts the reordered sub-path and its corresponding sub-tree back into the path they were extracted

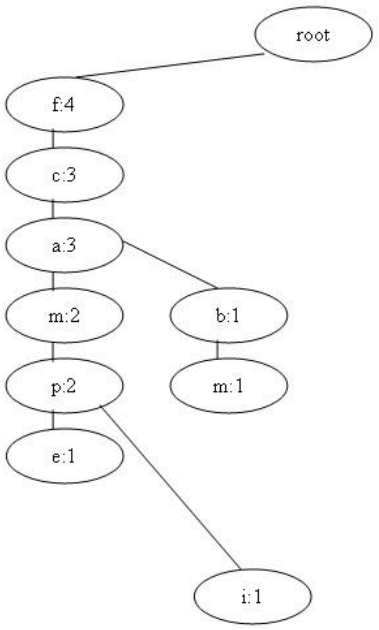


Figure 2.7: The paths associated with a in the DB-tree given in Figure 2.5

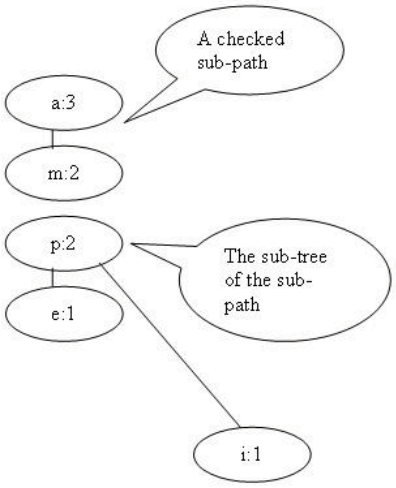


Figure 2.8: A checked sub-path and its sub-tree in Figure 2.7

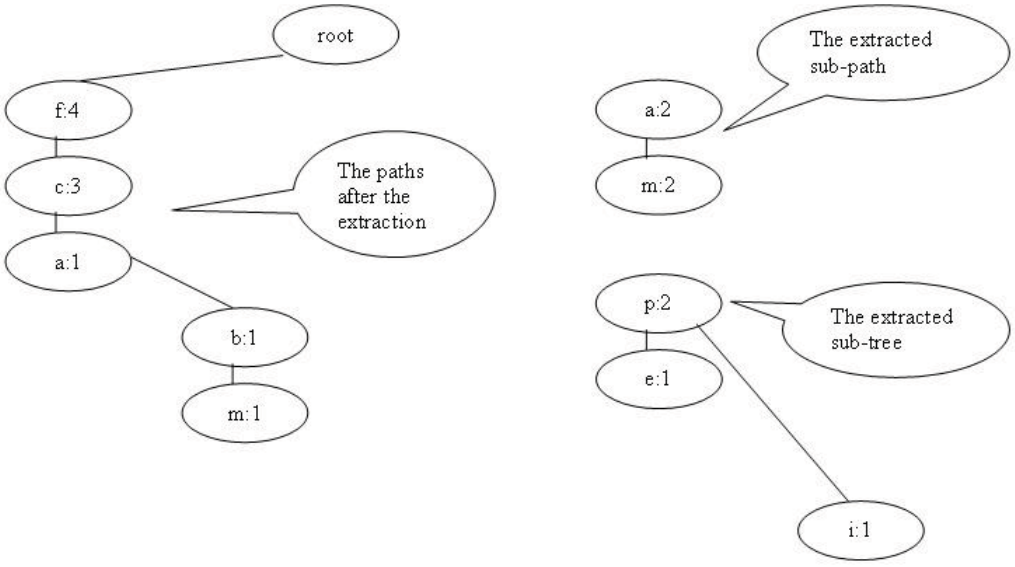


Figure 2.9: Extracting a sub-path and its sub-tree from the paths in Figure 2.7

from. The reordered paths associated with f in the result tree shown in Figure 2.6 are given in Figure 2.12.

After a dataset update, the incremental update procedure performs two main tasks. One is to mine frequent closed candidate patterns, and the other is to check their closure. However, in contrast to the pre-processing procedure, the incremental

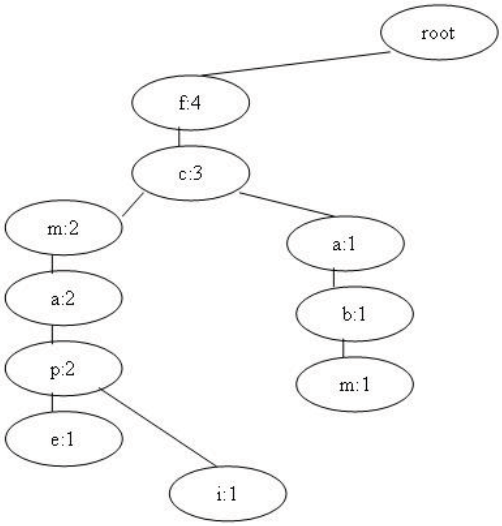


Figure 2.10: Inserting the reordered sub-path and its sub-tree back into the paths in Figure 2.9

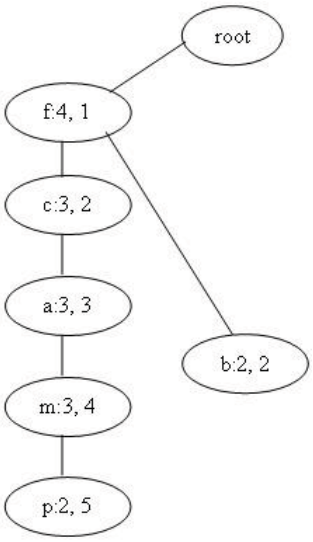


Figure 2.11: The paths associated with f in the result tree given in Figure 2.6

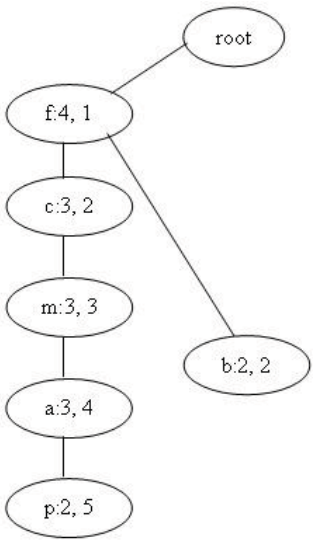


Figure 2.12: The reordered paths associated with f in the result tree given in Figure 2.6

update procedure needs to handle two different groups of frequent closed candidate patterns: the old ones mined before, and the new ones emerging after, the current dataset update.

Note that since the minimum frequency is applied to determine whether a pattern is frequent in a dynamic transaction dataset, a previously mined frequent pattern may not be frequent after a dataset update. Like the pre-processing procedure, only frequent patterns can be generated as frequent closed candidate patterns and the

infrequent patterns are abandoned.

It is hard to efficiently check the closure and prune the search space. Several useful lemmas can be obtained based on the influence of different dataset updates and the definition of frequent closed pattern. Suppose that there are two patterns z and z' in the original dataset D, $z \subset z'$, and D' is the updated dataset after a dataset update. Note that $sup(z, D) \geqslant sup(z', D)$, since $z \subset z'$.

Lemma 2.3.3. *When $sup(z, D) = sup(z', D)$, after an update adding new transactions into D:*

1. *if $sup(z, D') = sup(z, D)$, then $sup(z', D') = sup(z', D)$. Thus, $sup(z, D') = sup(z', D')$, so that if z is a non-closed pattern in D and its support value is the same after an update adding new transactions, z is still a non-closed pattern in D' and the closure of z does not need to be checked after the update.*

2. *if $sup(z, D') > sup(z, D)$, then $sup(z', D') \geqslant sup(z', D)$.*

   (a) *if $sup(z', D') = sup(z', D)$, then $sup(z, D') > sup(z', D')$. Thus, if z is a non-closed pattern in D, its support value increases after an update adding new transactions, and the support value of all its super-patterns is the same after the update, z becomes a closed pattern in D' and the closure of z does not need to be checked after the update.*

   (b) *if $sup(z', D') > sup(z', D)$, then $sup(z, D') \geqslant sup(z', D')$. Thus, if z is a non-closed pattern in D, its support value increases after an update adding new transactions, and the support value of any of its super-patterns is increased as well, the closure of z needs to be checked against its super-patterns with increased support value in D'.*

Lemma 2.3.4. *When $sup(z, D) > sup(z', D)$, after an update adding new transactions into D:*

1. *if $sup(z, D') = sup(z, D)$, then $sup(z', D') = sup(z', D)$. Thus, $sup(z, D') > sup(z', D')$, so that if z is a closed pattern in D and its support value is the same after an update adding new transactions, z is still a closed pattern in D' and the closure of z does not need to be checked after the update.*

2. *if $sup(z, D') > sup(z, D)$, then $sup(z', D') \geqslant sup(z', D)$.*

(a) *if* $sup(z', D') = sup(z', D)$, *then* $sup(z, D') > sup(z', D')$. *Thus, if* z *is a closed pattern in* D, *its support value is increased after an update adding new transactions, and the support value of all of its super-patterns are the same after the update,* z *is still a closed pattern in* D', *and the closure of* z *does not need to be checked after the update.*

(b) *if* $sup(z', D') > sup(z', D)$, *then* $sup(z, D') > sup(z', D')$. *Thus, if* z *is a closed pattern in* D, *its support value is increased after an update adding new transactions, and the support value of all of its super-patterns are increased as well,* z *is still a closed pattern in* D', *and the closure of* z *does not need to be checked after the update.*

Lemma 2.3.5. *When* $sup(z, D) = sup(z', D)$, *after an update deleting some existing transactions from* D:

1. *if* $sup(z, D') = sup(z, D)$, *then* $sup(z', D') = sup(z', D)$. *Thus,* $sup(z, D') = sup(z', D')$, *so that if* z *is a non-closed pattern in* D *and its support value is the same after an update deleting some existing transactions from* D, *then* z *is still a non-closed pattern in* D', *and the closure of* z *does not need to be checked after the update.*

2. *if* $sup(z, D') < sup(z, D)$, *then* $sup(z', D') < sup(z', D)$ *and* $sup(z, D') = sup(z', D')$. *Thus, if* z *is a non-closed pattern in* D *and its support value decreases after an update deleting some existing transactions from* D, z *is still a non-closed pattern in* D', *and the closure of* z *does not need to be checked after the update.*

Lemma 2.3.6. *When* $sup(z, D) > sup(z', D)$, *after an update deleting some existing transactions from* D:

1. *if* $sup(z, D') = sup(z, D)$, *then* $sup(z', D') = sup(z', D)$. *Thus,* $sup(z, D') > sup(z', D')$, *so that if* z *is a closed pattern in* D *and its support value is the same after an update deleting some existing transactions from* D, z *is still a closed pattern in* D', *and the closure of* z *does not need to be checked after the update.*

2. *if* $sup(z, D') < sup(z, D)$, *then* $sup(z', D') \leqslant sup(z', D)$.

(a) if $sup(z', D') = sup(z', D)$*, then* $sup(z, D') \geqslant sup(z', D')$*. Thus, if* z *is a closed pattern in* D*, its support value decreases after an update deleting some existing transactions from* D*, and the support value of any of its super-patterns is the same after the update, the closure of* z *needs to be checked against its super-patterns with unchanged support value after the update.*

(b) if $sup(z', D') < sup(z', D)$*, then* $sup(z, D') \geqslant sup(z', D')$*. Thus, if* z *is a closed pattern in* D*, its support value decreases after an update deleting some existing transactions from* D*, and the support value of any of its super-patterns is decreased as well, the closure of* z *needs to be checked against its super-patterns with decreased support value after the update.*

Since only subset-checking is required in our proposed algorithm, based on the lemmas introduced in the previous and current sections, the incremental update procedure can efficiently implement the subset-checking and search space pruning.

The incremental update procedure is outlined below:

Input: The added or deleted transactions ($not(D \cap D')$) affected by the update, the DB-tree and the two-level hash-indexed result tree produced by the previous procedure and a minimum frequency value min_freq. D denotes the original transaction dataset before the update and D' denotes the updated dataset after the update.

Output: The updated DB-tree and the updated two-level hash-indexed result tree based on the updated transaction dataset D'.

Method:

1. *Update the header table of the DB-tree. Scan the added or deleted transactions affected by the update once to update the support value of the items in the header table of the DB-tree by adding the support value of the items in the added transactions, or subtracting the support value of the items in the deleted transactions. Insert new items and their support values into the header table of the DB-tree if there are some new emerging items. If the support of an item is reduced to zero, remove the item from the header table, its linked nodes and the sub-trees of these nodes from the DB-tree.*

2. *Reorder the trees if necessary. Check whether the DB-tree needs to be reordered. Reorder the DB-tree and the result tree if required.*
3. *Update the nodes of the DB-tree based on the added or deleted transactions to let the DB-tree contain only the updated transaction dataset D'.*
4. *Use depth-first search to mine the updated DB-tree to generate frequent closed candidate patterns, like the mining process used in the pre-processing procedure, and check their closure with the result tree. However, if the support value of a frequent closed candidate pattern z is not changed by the dataset update, there is no need to mine z, nor the subtree that has z as its prefix path from the root of the DB-tree to the root of the subtree. The support value and the closure status of any pattern mined by growing z are not changed by the dataset update. The subtree, which has z as the prefix path from the root of the result tree to the root of the subtree, can be directly updated by just deleting the infrequent patterns from the subtree.*
5. *When no more frequent closed candidate patterns can be found in the DB-tree, prune the paths in the result tree that are not updated by the incremental update procedure. The result tree now contains only the frequent closed patterns in D'.*

Given the dataset in Table 2.3, the DB-tree updated by the incremental update procedure is given as Figure 2.13. With the same specified minimum frequency value of 40%, the updated two-level hash-indexed result tree is shown in Figure 2.14.

2.4 Experimental Evaluation

This section is devoted to a set of experiments conducted to compare our incremental mining version of CLOSET+ with the use of the original algorithm and rerunning it each time the dataset changes.

2.4.1 Experimental Environment and Performance Parameters

All the experiments were performed on a 3.20GHz Intel Pentium(R) 4 PC machine with 250MB of available main memory, running the Windows XP professional oper-

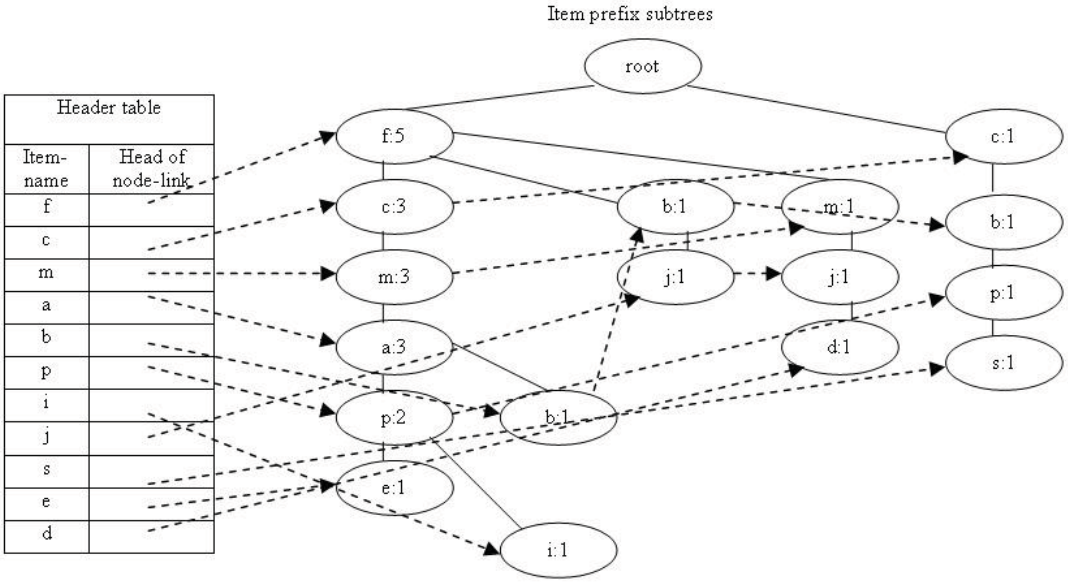


Figure 2.13: The updated DB-tree based on the updated dataset shown in Table 2.3

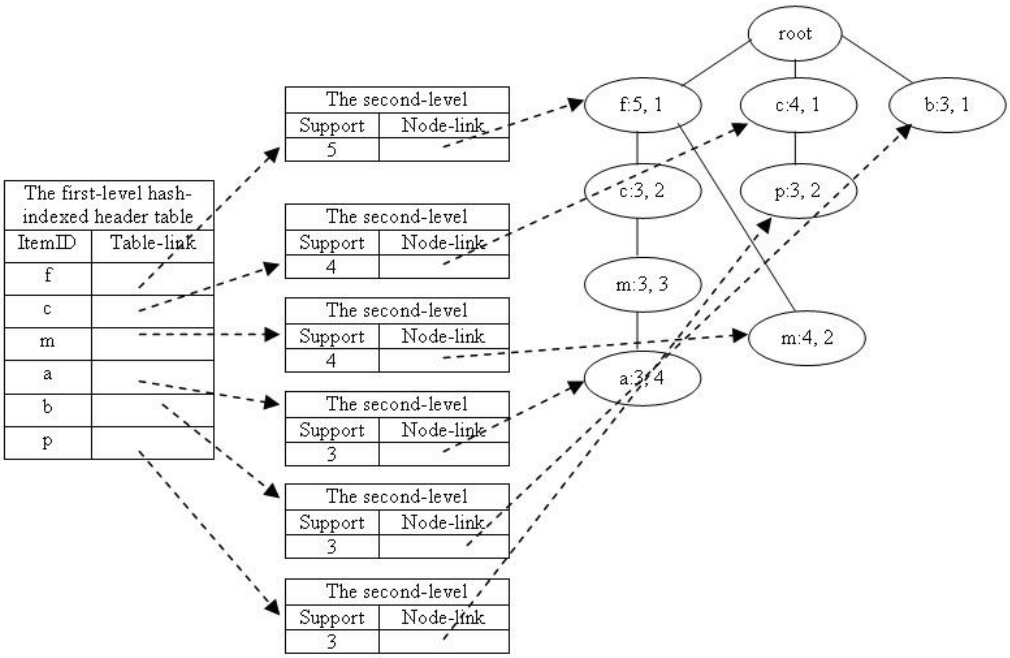


Figure 2.14: The updated result tree based on the updated dataset shown in Table 2.3 when $min\_freq = 40\%$

ating system. As mentioned previously, the study in this chapter is to demonstrate how to extend conventional mining methods to handle dynamic datasets and also gives an insight into the strengths and limitations of the incremental approach. Since our proposed algorithm is an extension of CLOSET+, the performance of our proposed algorithm is only compared with CLOSET+ in the same running environ-

ment. All the algorithms are coded in C++. We implement CLOSET+ to the best of our knowledge based on the published articles. CLOSET+ runs on the entire updated dataset each time the dataset is updated.

We were unable to find a real-world freely available dynamic transaction dataset, and so the experiments are based on synthetic transaction datasets generated based on the procedure proposed by Agrawal and Srikant [5]. The synthetic transaction datasets generated are assumed to be reasonably close to real retail transaction datasets.

The method takes several input parameters specified by users to generate a set of synthetic transactions. The size of a transaction is chosen from a Poisson distribution with mean equal to an input parameter that is the average size of the transactions. Items are assigned to the transaction by computing the union of a set of potentially frequent patterns. The size of each potentially frequent pattern is selected from a Poisson distribution with mean equal to an input parameter that is the average size of the maximal potentially frequent patterns. Each potentially frequent pattern is associated with a weight, which is selected from an exponential distribution with unit mean. Potentially frequent patterns are chosen one-by-one according to their weights to generate transactions. For each potentially frequent pattern, some items are generated randomly, and the others are inherited from the previous pattern. More details of this procedure can be found in [5].

We used an implementation of this procedure from the IBM Almaden Research Centre[4]. The generated synthetic transaction data is stored in ASCII format. Each item in a generated transaction is represented by an integer number. This data generation process has been widely used in the experimental evaluation of mining frequent closed patterns. The parameter settings used to generate the synthetic transaction datasets for our experiments are listed in Table 2.4.

Number of different items	1000
Average size of the transactions	10
Number of potentially frequent patterns	10,000
Average length of potentially frequent patterns	4
Correlation between patterns	0.25
Average confidence in an association rule	0.75

Table 2.4: Parameter settings

[4] `http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/datasets/syndata.html`

In common with [24, 105, 116] we simulated the dynamic dataset by randomly selecting transactions to be the original dataset and subsequent updates. The performance measure of the experiments is the run time, which refers to the total execution time, i.e., the period between the initial transactions being presented and the output appearing.

To reduce the effect of random factors and get relatively fair results, each experiment is repeated 20 times. The mean results and associated standard deviations are computed. These random factors can be caused by other tasks being run on the computer, the performance of disk accesses and implementation details. These random factors can create the uncertainty of the execution time of a run.

The experiments are conducted with respect to the specified value for the minimum frequency, the type of an update, and the size of an update. Note that the size of an update here refers to the total number of transactions in the update.

2.4.2 Measuring Performance with the Minimum Frequency Value

A set of experiments were performed to test the performance of our coded program with different minimum frequency values in static transaction datasets. The specified value for the minimum frequency value affects the run time of any implemented mining algorithm of frequent closed patterns. To effectively show the comparison on performance, the minimum frequency values used in the following experiments need to be identified first. The set of experiments conducted here serve this purpose.

Figure 2.15 shows the experimental results. The corresponding standard deviation values of the results in Figure 2.15 are of the order of 10^{-2}. Figure 2.15 roughly reveals the relationship between the run time of our coded program and the minimum frequency values from 0.3% to 0.1%. Note that the run time in those results actually refers to the performance of the pre-processing procedure of our proposed algorithm. The synthetic transactions used comply with the previous descriptions.

The run time is significantly longer when the minimum frequency values of 0.1% and 0.15% are used. The difference between the two sequential values is also notably bigger. Thus, we use these values for all remaining experiments. When the specified value of the minimum frequency gets greater, the total number of frequent closed patterns gets smaller and the proportion of frequent closed 1-patterns becomes greater. This is another reason why a minimum frequency value of 0.15% was

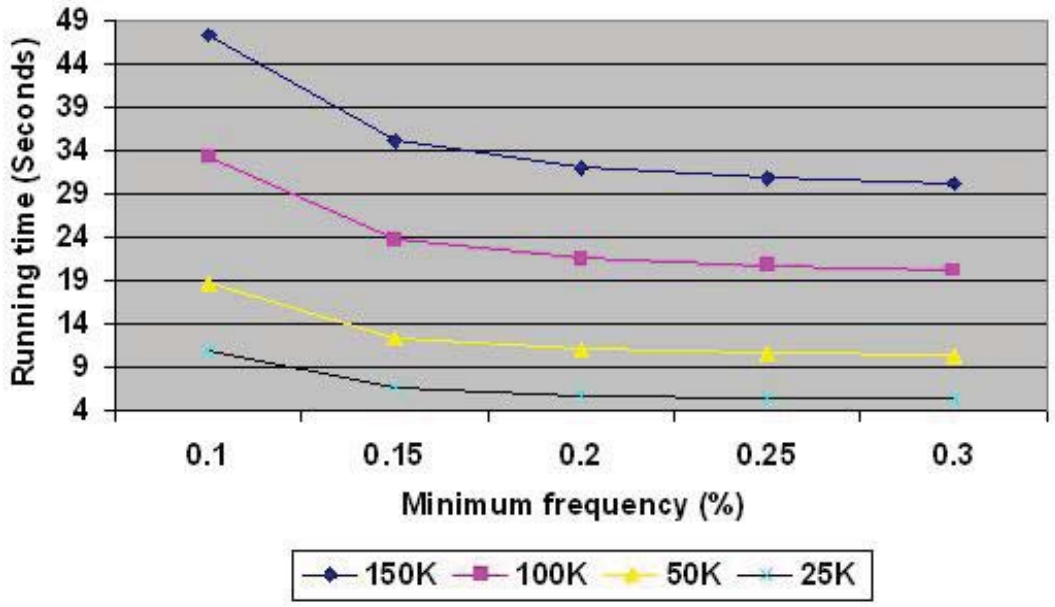


Figure 2.15: The relationship between the run time and the minimum frequency value

chosen, instead of 0.3%.

2.4.3 Measuring Performance with the Type and Size of an Update

We next test the scalability of our proposed algorithm with different types and sizes of an update.

Figure 2.16 shows the experimental results regarding the scalability with a different size of an update adding new transactions in the original dataset. The corresponding standard deviation values of the results in Figure 2.16 are shown in Table 2.5. The total number of transactions in the original dataset in these experiments is $100K$.

It is clear that the run time of our proposed algorithm is much less than that of rerunning CLOSET+ on each updated dataset when the size of the update is small compared with the number of transactions in the original dataset. In general, when the number of added transactions is less than 1% of the number of transactions in the original dataset, our proposed algorithm is faster than CLOSET+, mainly because that algorithm has to recreate and search the static tree-like data structures at each iteration.

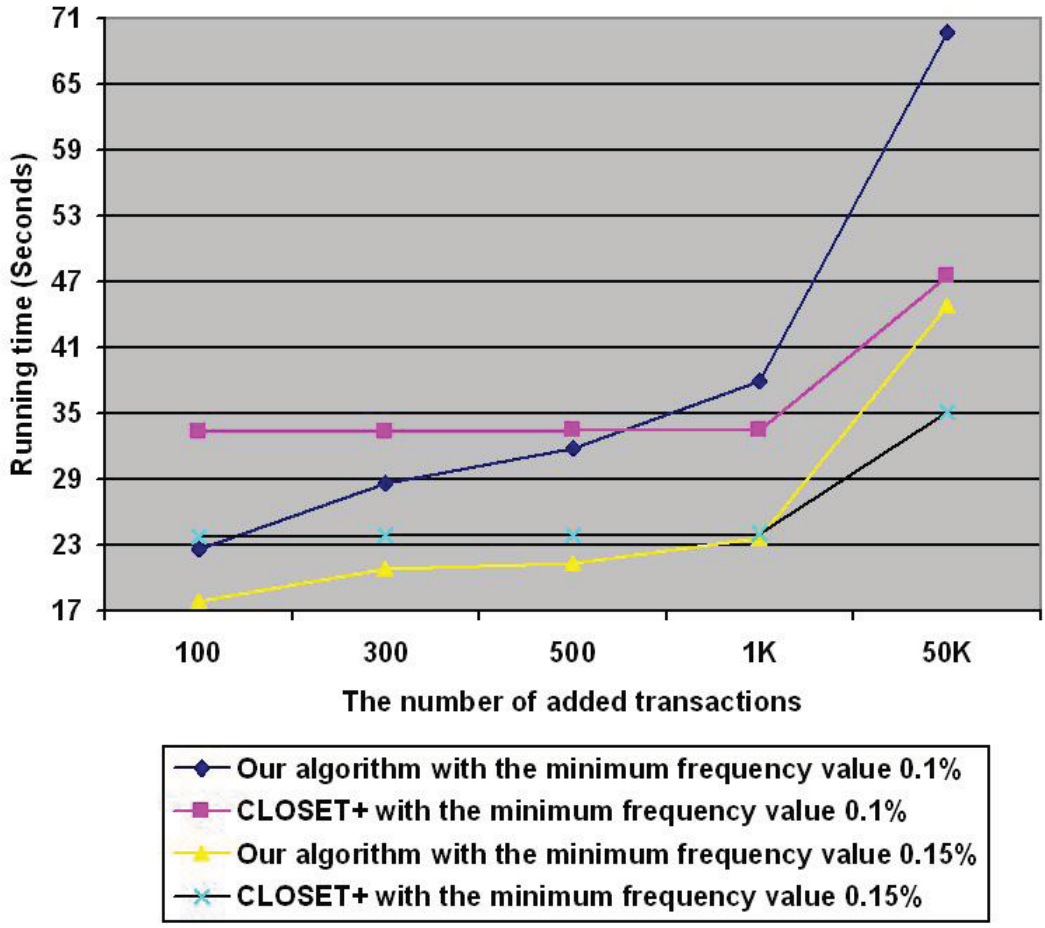


Figure 2.16: Scalability with the size of an update adding new transactions in the original dataset

Figure 2.17 shows the experimental results regarding the scalability with a different size of an update deleting existing transactions from the original dataset. The standard deviation values associated with the results in Figure 2.17 are given in Table 2.6. The total number of transactions in the original dataset in these experiments is $100K$.

It is clear that our proposed algorithm requires less run time than rerunning CLOSET+ on the transactions when the number of unchanged transactions is very small compared with the number of transactions in the original dataset. More specifically, our proposed algorithm is more efficient than rerunning CLOSET+ when less than 0.5% of transactions in the original dataset are deleted.

For all mining algorithms adopting the incremental method, handling the updates of deleting existing transactions is more computationally expensive. Algo-

	100	300	500	$1K$	$50K$
Our proposed algorithm with the minimum frequency value 0.1%	3.251442	2.217918	3.642159	3.747662	7.845326
CLOSET+ with the minimum frequency value 0.1%	0.04804	0.061553	0.037956	0.089401	0.060773
Our algorithm with the minimum frequency value 0.15%	2.057124	1.95186	2.386864	1.979323	4.301376
CLOSET+ with the minimum frequency value 0.15%	0.039723	0.028838	0.034003	0.053103	0.044856

Table 2.5: The corresponding standard deviation values of the results shown in Figure 2.16

	100	300	500	$1K$	$50K$
Our algorithm with the minimum frequency value 0.1%	1.538456	1.88692	4.21087	4.113333	7.894779
CLOSET+ with the minimum frequency value 0.1%	0.097017	0.052414	0.083462	0.029679	0.100493
Our algorithm with the minimum frequency value 0.15%	1.561854	2.329718	2.840679	3.018742	9.275538
CLOSET+ with the minimum frequency value 0.15%	0.064459	0.061222	0.018302	0.029597	0.047554

Table 2.6: The corresponding standard deviation values of the results shown in Figure 2.17

rithms adopting the incremental method aim to maintain the previously-discovered knowledge and update them after updates. If too many transactions are deleted, many previously mined results will expire. The operations of maintaining, updating and removing these useless results can cost lots of memory space and execution time. That can result in unnecessary, expensive and inefficient computation for mining the frequent closed patterns in the updated dataset.

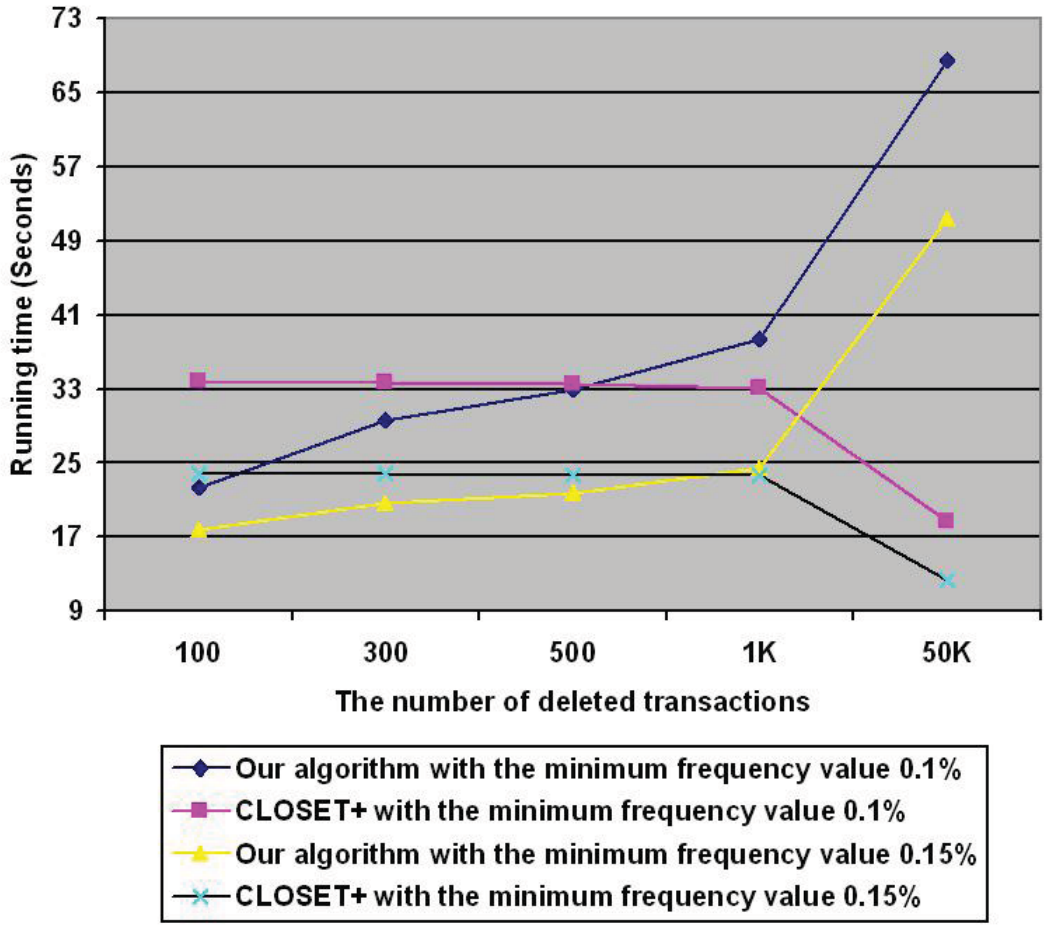


Figure 2.17: Scalability with the size of an update deleting existing transactions from the original dataset

2.4.4 Measuring Scalability with the Number of Transactions in the Original Dataset

Our next set of experiments tests the scalability of our proposed algorithm with different numbers of transactions in the original dataset.

Figure 2.18 shows the experimental results regarding the scalability with a different number of transactions in the original dataset, when the size of an update adding new transactions is fixed at 0.1% of the number of transactions in the original dataset. The standard deviation values associated with the results in Figure 2.18 are given in Table 2.7. As can be seen from Figure 2.18, the larger the original transaction dataset is, the shorter the run time of our proposed algorithm compared with rerunning CLOSET+.

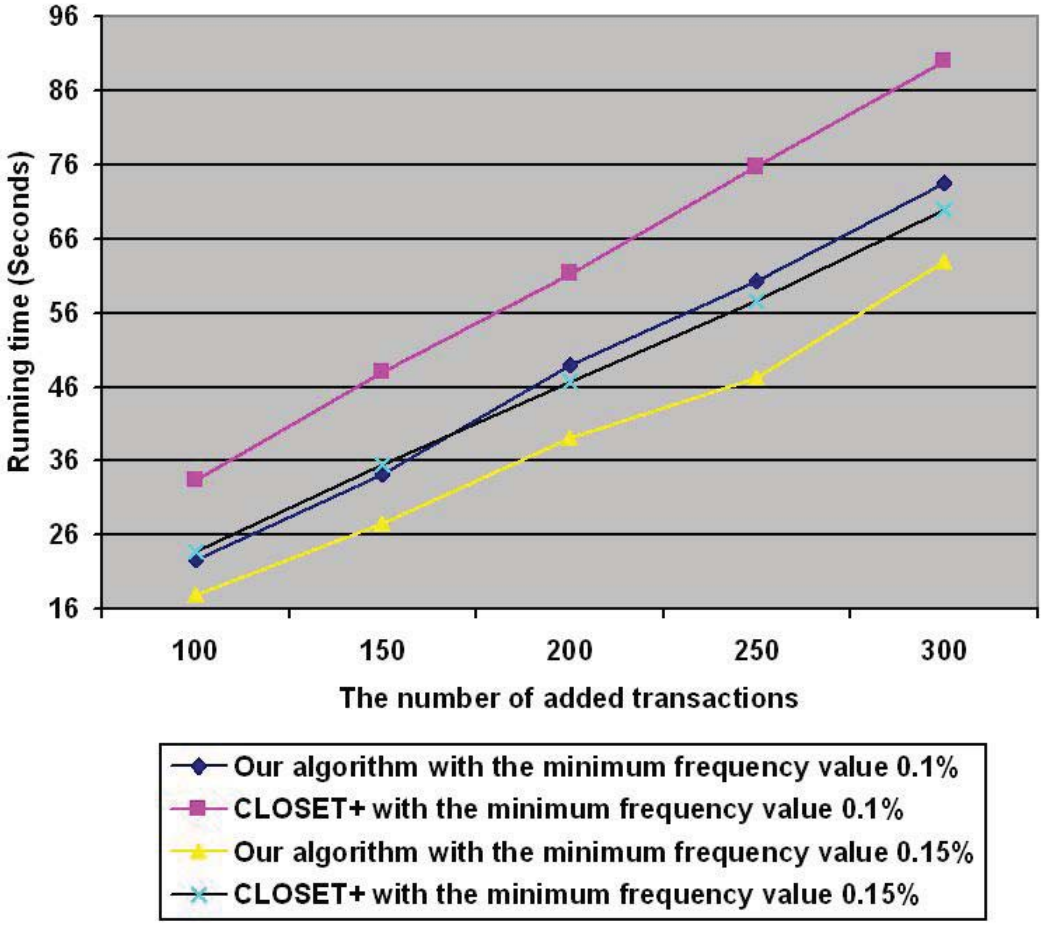


Figure 2.18: Scalability with the number of transactions in the original dataset, when the number of added transactions is fixed at 0.1% of the original dataset

The experimental results show that our proposed algorithm has a notable performance advantage over rerunning CLOSET+ for mining frequent closed patterns in dynamic transaction datasets when the number of transactions affected by the update is relatively small. Each dataset update may change some previously-discovered results. When the number of transactions affected by the update is relatively small, the previously-discovered results are affected very little, which maximizes the benefit of maintaining the previously-discovered results and improves the performance of mining process, though this kind of maintenance is normally memory-consuming. Thus, in this case the performance of algorithms adopting the incremental method to mine frequent closed patterns in dynamic transaction datasets is much better than adopting the rerunning method. This is also the key reason why in this case our proposed algorithm shows a much better performance than CLOSET+.

	100	150	200	250	300
Our algorithm with the minimum frequency value 0.1%	3.251442	2.080061	1.544821	1.248148	3.305489
CLOSET+ with the minimum frequency value 0.1%	0.04804	0.539493	0.593714	0.699251	0.74348
Our algorithm with the minimum frequency value 0.15%	2.057124	1.747591	2.933178	1.155505	6.414934
CLOSET+ with the minimum frequency value 0.15%	0.039723	0.274523	0.685666	0.637869	0.769947

Table 2.7: The corresponding standard deviation values of the results shown in Figure 2.18

As a final note, it should be mentioned that, based on the experimental results, by using different minimum frequency values (i.e., 0.1% and 0.15%), a trend can be observed. The performance of our proposed algorithm with lower minimum frequency values is more sensitive to the ratio of the size of an update and the number of transactions in the original dataset. With the same original transaction dataset, a lower minimum frequency value generally produces more frequent closed patterns and also increases the number of frequent closed patterns with longer length. Maintaining those frequent closed patterns requires more memory space and execution time. When a dataset update occurs, the more such frequent closed patterns are maintained, the higher the probability of those patterns being affected by the update. The more frequent closed patterns are affected by the update, the longer the run time required by mining algorithms adopting the incremental method. Thus, when the number of transactions affected by the update is relatively large compared with the one in the original dataset, the run time of our proposed algorithm with a lower minimum frequency increases faster. The increased cost mainly relies on the number of maintained frequent closed patterns, which is decided by the characteristics of the input dataset and the specified minimum frequency value. It means that similar trends are expected to be found by using any incremental mining algorithm with different input datasets and minimum frequency values.

2.5 Summary

In this chapter, the problem of mining frequent closed patterns in dynamic transaction datasets as a natural complement to the problem originally set in static transaction datasets is formally defined and studied. A new incremental mining algorithm is proposed to handle this problem by applying the incremental method on the CLOSET+ algorithm [120], which is one of the most efficient algorithms for mining frequent closed patterns in static transaction datasets. The purpose of the study in this chapter is to investigate how to use the incremental method to extend existing mining algorithms to efficiently handle dynamic datasets. With this purpose, only the principal data structures and techniques of CLOSET+ are implemented. Our proposed algorithm can serve as the core procedure and the platform for adopting other unimplemented techniques, although this is left to future work.

The proposed algorithm uses two tree-like data structures (the DB-tree and the two-level hash-indexed result tree) to compactly retain the dynamic transaction and the previously-discovered results. It then updates them by scanning the transactions affected by the update twice, after each dataset update of adding new transactions or deleting existing transactions from the original transaction dataset occurs. Hence, without requiring the entire updated dataset to be scanned again, the proposed algorithm can incrementally mine the current frequent closed patterns in the updated dataset by considering the changes caused by the updates only.

Several sets of experiments were conducted to explore the relative merits and limitations of the proposed algorithm compared with rerunning CLOSET+ on the entire updated dataset. The experimental results show that the proposed algorithm is more efficient than rerunning CLOSET+ in the updated dataset when the number of transactions changed by the dataset update is relatively small compared with the number of unchanged transactions in the dataset.

Enhancing the scalability of the proposed algorithm and reducing the cost of reordering the DB-tree and the two-level hash-indexed result tree during the mining process are in the future plan, so that the proposed algorithm can also be efficiently applied to other situations.

In general, the incremental method is about trade off. It intends to find a balance between the benefits of maintaining previously-discovered information and the associated costs in order to reduce the cost of discovering the information in the updated dataset. Thus, any algorithm adopting the incremental method actually attempts to find, use and maintain the balance to improve its performance. That is

also the golden rule for designing and modifying incremental algorithms.

The performance of mining frequent patterns can also be affected by the characteristics of target datasets. As an important characteristic of real datasets, the pattern support distribution is investigated in the following chapter in order to design efficient mining algorithms and turn the performance of mining algorithms with target datasets.

Chapter 3

Investigating Power-law Relationships in Pattern Support Distributions

It is common sense to take a method and try it.
If it fails, admit it frankly and try another. But above all, try something.
— Franklin D. Roosevelt

A Pattern Support Distribution (PSD) is an important characteristic of a dataset. The support of a pattern indicates the frequency of its occurrence in a dataset, and the distribution of the number of patterns against their corresponding supports is known as the pattern support distribution of the dataset. Identifying pattern support distributions of target datasets is very useful for many data mining tasks such as mining frequent patterns.

Power-law relationships appear very often in natural and man-made worlds such as the populations of cities [7]. Chuang et al. [28, 29] claim that power-law relationships and self-similarity phenomena also exist in the pattern support distributions of real datasets. In this chapter, we investigate the truth of this.

Identifying power-law relationships in the pattern support distributions of target datasets can provide better understanding of, and more successful applications of pattern support distributions. However, verifying whether power-law relationships really exist in pattern support distributions is difficult because of the large statistical fluctuations co-existing with power-law relationships in real world examples. Chuang et al. [28, 29] made their claim based on observing the visualization of some instances. Such a claim should be made based on statistical tests. Therefore, their claim is

unverified.

This chapter mainly focuses on the problem of how to use quantitative goodness-of-fit tests to examine whether power-law relationships exist in the pattern support distributions of real static transaction datasets. Some discussions, such as [30], have proposed some approaches to statistically verifying whether a power-law relationship really exists in a set of empirical data. These tests are limited to examining an instance of all possible datasets generated by an underlying process. That makes our problem more challenging since we intend to do the verification over the whole population generated by an underlying process. By extending the approach suggested by Clauset et al. [30], this chapter proposes a new method of utilizing the bootstrap method and the universality of power-law relationships to conduct quantitative verifications of whether power-law relationships exist in the pattern support distributions of real transaction datasets. Based on a large number of statistical goodness-of-fit test results and discussions in this chapter, eventually a new and more proper claim is given, which is that the hypothesis that power-law relationships exist in the pattern support distributions of real retail transaction datasets cannot be ruled out at the level of basic distributions. This is different to that given by Chuang et al. [28, 29].

This chapter is organized as follows. After introducing the background and formally defining the pattern support distribution of a transaction dataset in Section 3.1, we overview power-law relationships, especially the special properties of power-law relationships, fitting discrete power-law distributions to a set of empirical data and verifying the existence of power-law relationships in a set of empirical data in Section 3.2. By observing the visualization—a power-law relationship shows a roughly straight line in a log-log plot—of the partial (cumulative) pattern support distributions of five real transaction datasets, a qualitative appraisal is made and generates a hypothesis that power-law relationships exist in the pattern support distributions of real retail transaction datasets in Section 3.3. This hypothesis is further tested and verified based on a large number of experimental results and discussions. Moreover, the self-similarity phenomenon linked to power-law relationships in the pattern support distributions of real retail transaction datasets is also explored in this section. This chapter is summarized in Section 3.4.

3.1 Introduction

Some interesting work has been done along the line of discovering deeper knowledge about the characteristics of real datasets. For instance, Ramesh et al. [99] studied the length distributions of frequent and maximal frequent patterns, which show the relationship between the count of patterns and their lengths. Ramesh et al. [99] also attempted to apply their study to generate more realistic synthetic datasets for algorithm bench-marking. Lhote et al. [68] used probabilistic techniques to investigate the relationship between the average number of frequent (closed) patterns and their supports. Besson et al. [15] sought the relationship between the number of patterns and the conjunction of maximal frequency, minimal frequency and size constraints, which can guide users to choose initial parameter settings for substring pattern discovery.

One important characteristic of a dataset is the pattern support distribution. A pattern support distribution is a discrete distribution that comprises a set of points (x_i, y_i) with an absolute/relative support value x_i and a number y_i of the patterns with the support value x_i. This chapter concentrates on the pattern support distribution (denoted as $PSD(TD)$ or PSD) of a transaction dataset TD based on the absolute support of patterns. However, the ideas and methods applied to absolute support in this chapter can be applied to relative support too.

From the view of probability theory and statistics, a pattern support distribution can indicate the discrete probability distribution that expresses the probability that the support of an arbitrary pattern in a target dataset is equal to some particular value[1]. Pattern support distributions are of use in many data mining tasks, such as providing a method of determining an appropriate minimum support for mining frequent patterns, synthetic data generation and frequency approximation over data streams [28, 29].

A power-law relationship generally indicates a special inverse relationship between two quantities[2]. It indicates how one quantity reduces with the increase of the other quantity. Power-law relationships have been observed in many natural and man-made phenomena including city sizes, word frequencies, and sightings of bird species in the United States, to name a few [3, 7, 83, 85, 86]. Its ubiquity and

[1]Based on the definition in the Cambridge Dictionary of Statistics [35], in probability theory and statistics, a probability distribution identifies the probability of each value of a discrete random variable, such as the binomial distribution, or it gives the probability of the value of a continuous random variable, which falls within a particular interval.

[2]The work presented in this thesis is limited to two-dimensional space.

mathematical features make it of interest.

Chuang et al. [28, 29] observed the pattern support distributions drawn from several real retail datasets and claimed that power-law relationships also exist in pattern support distributions of real datasets. To the best of our knowledge, the work done by Chuang et al. [28, 29] is the first and only work on power-law-based pattern support distributions so far. Identifying power-law relationships in pattern support distributions can provide a better insight into real datasets and benefit mining applications. However, like most of the work regarding power-law relationships in the literature, [28, 29] do not provide any statistical tests to verify their claim.

In addition, the self-similarity phenomenon is often linked to power-law relationships. A self-similar object is exactly or approximately similar to a part of itself [78]. Chuang et al. [28, 29] claimed that power-law relationships also exist in the support distribution of patterns with a certain length and that it is a self-similarity phenomenon.

A power-law-based distribution is different to most other distributions, such as the normal distribution. Most of the other distributions show that quantities are distributed around their average values. That is to say, the probability of instances increases when the instances are closer to the mean. Such distributions can be well depicted by their mean and corresponding standard deviations. However, a power-law-based distribution cannot be depicted by these simple measurements. This abnormal characteristic of a power-law-based distribution often implies that some kind of complex underlying process is behind the distribution. It makes the power-law-based distributions more interesting to investigate, especially since the complex underlying processes are unknown in most cases. For the same reason, to the best of our knowledge, work such as [30] only discusses how to verify whether a set of empirical data (i.e., an instance of datasets) follows a power-law relationship.

The rest of this chapter mainly proposes a new method of using statistical tests and special properties of power-law relationships to verify whether power-law relationships exist in pattern support distributions. Our proposed method extends the verification of power-law hypothesis on an instance to a whole population of target datasets generated by an unknown process.

3.2 Overview of Power-Laws

A power-law relationship reveals the relationship between two variables Y and X which satisfy the equation

$$y = kx^{-\alpha} , \tag{3.1}$$

where y is the observed value of Y, x is the observed value of X, k is a proportionality constant determined empirically, and α is a positive constant parameter known as the exponent or scaling parameter.

As an empirical law, all specified functions describing power-law relationships are formulated based on observations. It simply says that the kind of relationship or phenomenon can be observed under the same conditions. Different from a theory, an empirical law does not include a formal explanation of the underlying cause of the relationship or phenomenon. Therefore, theoretically one can never be sure that the empirical data observed is generated by a specified function of the power-law relationship. The typical way to describe this should be that the observed empirical data is in agreement with the model depicting the power-law relationship in Equation 3.1. Based on this common understanding, for the sake of simplicity, in practice, people often say that some observed empirical data satisfy a particular function of power-law relationships, such as Zipf's law [141, 142].

3.2.1 Continuous and Discrete Power-law Distributions

Power-law relationships can be used to describe two kinds of distributions of values: continuous and discrete distributions. A continuous power-law distribution describes continuous real values, while a discrete power-law distribution is for discrete values, ordinarily positive integers.

Let X be an independent variable and a function $p(x)$ be a corresponding function that measures the quantity of the observed value x of X. A continuous power-law distribution uses $p(x)$ to indicate the sum of the corresponding values of $p(x)$ when x is within a particular interval $(a, b]$. When a continuous power-law distribution is used to describe a probability distribution, $p(x)$ is the probability density function (pdf) of the probability distribution, which can be expressed as:

$$Pr(a < X \leqslant b) = \int_a^b p(x)dx = \int_a^b Cx^{-\alpha}dx , \tag{3.2}$$

where

- x is the observed value of X;
- C is the normalizing constant;
- α is the exponent, or scaling, or scaling exponent parameter.

For example, the Pareto distribution is a Power-law continuous probability distribution whose probability density function is

$$p(x) = \frac{\alpha x_m^{\alpha}}{x^{(\alpha+1)}} \,, \tag{3.3}$$

where x_m is the minimum possible positive real value of X and α is a positive real value.

Supposing that $\alpha > 0$, $\alpha \neq 1$ and $x_{\min}$ is the lower bound where the power-law distribution holds, the normalizing constant can be easily computed:

$$C = \frac{1}{\int_{x_{\min}}^{+\infty} x^{-\alpha}\, dx} = \frac{\alpha - 1}{{x_{\min}}^{(1-\alpha)}}. \tag{3.4}$$

Thus, the continuous power-law distribution becomes:

$$p(x) = \frac{\alpha - 1}{x_{\min}} \left(\frac{x}{x_{\min}} \right)^{-\alpha}. \tag{3.5}$$

A discrete power-law distribution indicates the corresponding value of $p(x)$ when x represents the value of a discrete random variable. When a discrete power-law distribution is used to represent a probability distribution, $p(x)$ is the probability mass function (pmf) of the probability distribution, which has the form:

$$p(x) = Pr(X = x) = Cx^{-\alpha}. \tag{3.6}$$

For instance, the Zipf distribution is one of the family of discrete power-law probability distributions. The probability mass function of the Zipf distribution is

$$p(x) = \frac{x^{-s}}{H_{N,s}} \,, \tag{3.7}$$

where N is a natural number, s is a real value and $H_{N,s}$ is the Nth generalized harmonic number (i.e., $\sum_{n=1}^{N} n^{-s}$). The Zipf distribution may be considered as a discrete counterpart of the Pareto distribution.

Given a lower bound $x_{\min}$ on the discrete power-law distribution, after computing the normalizing constant

$$C = \frac{1}{\sum_{i=x_{\min}}^{+\infty} (i^{-\alpha})} = \frac{1}{\zeta(\alpha, x_{\min})}\ , \tag{3.8}$$

with $\alpha > 1$ and $x_{\min} > 0$, the discrete integer power-law distribution becomes:

$$p(x) = \frac{x^{-\alpha}}{\zeta(\alpha, x_{\min})}, \tag{3.9}$$

where $\zeta(\alpha, x_{\min})$ is a Hurwitz zeta function [55] or generalized Riemann zeta function [101], i.e.,

$$\zeta(\alpha, x_{\min}) = \sum_{i=0}^{+\infty} (i + x_{\min})^{-\alpha}. \tag{3.10}$$

In many cases, it is worth studying the complementary distributions of power-law distributions, which indicate the sum of the corresponding values of $p(x)$ when x is above a particular value. In probability theory and statistics, the cumulative distribution function (cdf) $P(x)$ is used to describe the kind of distribution and can be written as:

$$P(x) = Pr(X \geqslant x). \tag{3.11}$$

For the continuous random variable, a cumulative distribution function is

$$\begin{aligned} P(x) = Pr(X \geqslant x) &= C \int_x^{+\infty} p(t)\, dt \\ &= \frac{\alpha - 1}{{x_{\min}}^{-\alpha+1}} \int_x^{+\infty} t^{-\alpha}\, dt \\ &= \left(\frac{x}{x_{\min}}\right)^{(-\alpha+1)}. \end{aligned} \tag{3.12}$$

Note that Equation 3.12 indicates that the cumulative distribution of a continuous power-law distribution is also a power-law distribution with a smaller scaling exponent value $(\alpha - 1)$.

For discrete random variables, the cumulative distribution function is:

$$\begin{aligned} P(x) = Pr(X \geqslant x) &= \sum_{i=x}^{+\infty} p(i) \\ &= \sum_{i=x}^{+\infty} \left(\frac{i^{-\alpha}}{\zeta(\alpha, x_{\min})}\right) \end{aligned}$$

$$= \frac{\zeta(\alpha, x)}{\zeta(\alpha, x_{\min})}. \tag{3.13}$$

The form of Equation 3.13 is too complicated to directly show that the cumulative distribution of a discrete power-law distribution roughly displays a power-law relationship. However, since a discrete power-law distribution can be approximately computed by a continuous power-law distribution, the cumulative distribution of the discrete power-law distribution can be approximately computed by the cumulative distribution of the continuous power-law distribution as well. That indicates that the cumulative distribution of a discrete power-law distribution with a scaling exponent value α approximately follows a power-law distribution with the scaling exponent value $(\alpha - 1)$. This inference can also be revealed by using the qualitative appraisal based on visualizations without loss of generality.

For instance, let $\alpha = 2.5$. Figure 3.1(a) shows pairs $(x, \zeta(2.5, x))$ in the natural log-log plot, where $25,000 \leqslant x \leqslant 250,000$. Those pairs $(x, \zeta(2.5, x))$ in Figure 3.1(a) follow a straight line. Also, compare $\zeta(2.5, x)$ with the power-law distribution $x^{-1.5}$ in Figure 3.1(b) where $25,000 \leqslant x \leqslant 250,000$.

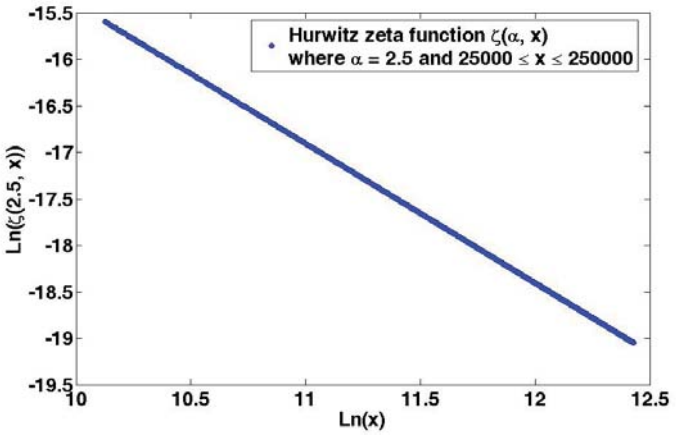
Note that

$$\begin{aligned}
&\lim_{x \to +\infty} \left[\zeta(2.5, x) - x^{-1.5}\right] \\
&\quad = \lim_{x \to +\infty} \left[\sum_{n=0}^{+\infty} (x+n)^{-2.5} - x^{-1.5}\right] \\
&\quad = \lim_{x \to +\infty} \left[-x^{-1.5} + x^{-2.5} + (x+1)^{-2.5} + (x+2)^{-2.5} + \cdots\right] \\
&\quad = 0
\end{aligned} \tag{3.14}$$

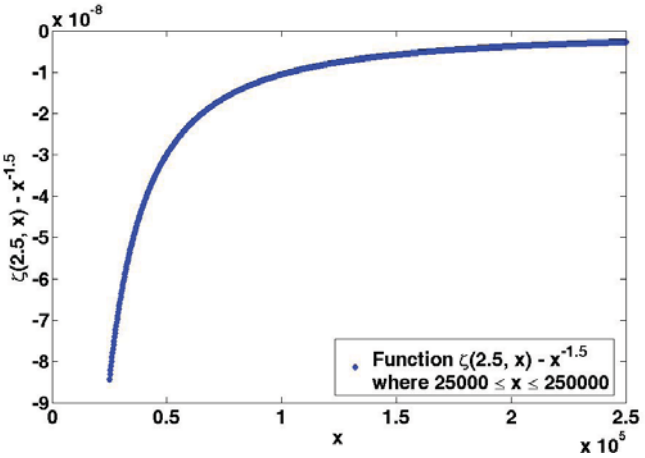
and

$$\lim_{x \to 1} \left[\zeta(2.5, x) - x^{-1.5}\right] = \zeta(2.5, 1) - 1 = \zeta(2.5) - 1 = 0.3415\ . \tag{3.15}$$

Since 0 and 0.3415 are very small values, $\zeta(2.5, x)$ is very close to the power-law distribution $x^{-1.5}$, when $x \in [1, +\infty)$. Moreover, $\zeta(2.5, x_{\min})$ is a constant with a certain value $x_{\min}$. Thus, $P(x) = \frac{\zeta(2.5,x)}{\zeta(2.5,x_{\min})}$, which is the cumulative distribution of the discrete integer power-law distribution $p(x) = \frac{x^{-2.5}}{\zeta(2.5,x_{\min})}$, is very close to the power-law distribution $\frac{x^{-1.5}}{\zeta(2.5,x_{\min})}$. This example demonstrates that the cumulative distribution of a discrete power-law distribution roughly follows a power-law distribution with a smaller scaling exponent value, which is approximately equal to



(a) Pairs $(x, \zeta(2.5, x))$ in the natural log-log plot



(b) Function $\zeta(2.5, x) - x^{-1.5}$ where $25,000 \leqslant x \leqslant 250,000$

Figure 3.1: The behaviour of Hurwitz zeta function $\zeta(\alpha, x)$, where $\alpha = 2.5$ and $25,000 \leqslant x \leqslant 250,000$

$(\alpha - 1)$. That is consistent with the inference based on that a discrete power-law distribution can be approximately computed by a continuous power-law distribution.

In practice, a discrete power-law distribution is often approximately computed by a corresponding continuous power-law distribution. From the formulas above, it is clear that the computation of the continuous power-law distribution is much easier than that of the discrete power-law distribution, since the computation of the discrete power-law distribution involves the calculation of a Hurwitz zeta function. In general, calculating the value of a Hurwitz zeta function requires complicated calculation. Consequently, for the sake of computational simplicity, in many applications, a discrete power-law distribution is approximated by a corresponding continuous power-law distribution [30].

There are many approximation methods and not all of them can generate accurate results in this case. For example, the approximations using rounding up or down normally generate poor results. A better approximation is to round the real values of x of a corresponding continuous power-law distribution to the nearest integer [30].

3.2.2 Special Properties of Power-Law Relationships

This section reviews two special properties of power-law relationships: universality and scale invariance. These two concepts are closely related and contribute to the development of a better understanding of complex processes and phenomena [80]. They have made power-law relationships attract particular interest over the years, and will be used later when analyzing pattern support distributions in Section 3.3.

Universality is the observation that there are some macroscopic properties for a class of systems that are independent of the particular microscopic mechanisms of the system [87]. The cause of the equivalence of the power-law relationships with a particular scaling exponent is the mechanism or process that generates the power-law relationship. A set of scaling exponents of the power-law relationships found in diverse systems or phenomena can be clustered into distinct classes called universality classes. It has been proposed that the systems or phenomena whose scaling exponents are in the same universality class share similar fundamental microscope mechanisms or generating processes [34]. Also, it might be the cause of the phenomena of scale invariance of these systems or phenomena. Understanding the underlying mechanism generating a power-law relationship in a universality class may lead to an explanation of the rest of the power-law relationships in the universality class.

For example, phase transitions happen at a critical temperature. The behaviour of a physical quantity shows a power-law relationship around the critical temperature. The scaling exponent of the power-law relationship is used to measure the behaviour of the phase transitions. Pure water and carbon dioxide (CO_2) are two different materials. However, they show the similar behaviour of phase transition at their boiling points and have identical scaling exponent values. Therefore, they are in the same universality class [111].

A power-law relationship demonstrates scale invariance, which means that scaling the argument in the function of a power-law relationship by a constant factor causes only a proportionate scaling of the function itself and the shape of the func-

tion is preserved. That is, with Equation 3.1 of a power-law relationship,

$$f(cx) = k(cx)^{-\alpha} = c^{-\alpha}kx^{-\alpha} \propto kx^{-\alpha} = f(x) \,, \tag{3.16}$$

where c and k denote constant factors.

The property of scale invariance indicates that power-law relationships with the same scaling exponent are equivalent up to constant factors. It guarantees that with appropriate factors of scaling transformation, the power-law relationships falling in the same universality class can show similar behaviour.

The scale invariance property produces a favoured behaviour of a power-law relationship, which is that a power-law relationship often appears as a roughly straight line in a log-log plot. Taking the natural logarithm of both sides of a power-law relationship, it can be expressed as:

$$\ln(f(x)) = -\alpha \ln(x) + b \,, \tag{3.17}$$

where b is the Y-intercept of the power-law relationship in the natural log-log plot. This expression shows the linear relationship between $\ln(f(x))$ and scaling exponent α. Scaling the argument x creates a linear shift of the function $\ln(f(x))$ up or down, while the form of the expression and the scaling exponent are not affected. This behaviour makes a power-law relationship easy to observe.

3.2.3 Fitting a Discrete Power-law Distribution to a Set of Empirical Data

This section reviews how to correctly identify the discrete power-law distribution best fitted to a target set of empirical data, which includes the estimation of the scaling exponent parameter and the lower bound parameter of the best-fit discrete power-law distribution.

Least-squares linear regression is the most common approach used in the literature with respect to computing a power-law distribution, due to its simplicity. As mentioned previously, a power-law distribution follows a straight line in a log-log plot. The distribution can be represented in the size-frequency form, such as Lotka's law [73], or the rank-frequency form, such as Zipf's law. The size-frequency form expresses the appearance frequency with a certain number of items, while the rank-frequency form expresses the appearance frequency of any item with its rank in the

frequency table [123]. Then, the scaling exponent value α can be interpreted as the absolute value of the slope of the straight line obtained by applying least-squares linear regression to these empirical data in a log-log plot. Algorithm PPL proposed by Chuang et al. [28, 29] for characterizing power-law-based pattern support distributions is based on least-squares linear regression.

However, as mentioned by Clauset et al. [30], there are some issues with this method and its variations based on the similar ideas, which includes using the least-squares linear regression on the cumulative distribution of a power-law distribution to characterize a power-law distribution[3]. The results obtained based on these methods are normally inaccurate, and may even be essentially wrong sometimes. As a result, this method and its variations should be avoided when characterizing a power-law distribution.

Estimating the scaling exponent parameter

Maximum likelihood estimation (MLE) is a relatively reliable statistical method used for fitting a power-law distribution to empirical data [30]. Maximum likelihood estimation finds the values for the distribution parameters that maximize the probability of the input empirical data. The maximum likelihood estimator is asymptotically unbiased, asymptotically efficient and distributed asymptotically normally [58]. These desired asymptotic properties give relatively accurate parameter estimations in the limit of large samples.

For any random value x that follows a power-law distribution in the interval $[x_{\min}, +\infty)$, one can compute the maximum likelihood estimator for the scaling exponent parameter of a power-law distribution.

Suppose x is defined as an integer variable, the function of a discrete power-law distribution is given in Equation 3.9. Based on the maximum likelihood estimation, the first step is to write the likelihood function of the scaling exponent parameter, i.e.,

$$\mathcal{L}(\alpha) = \prod_{i=1}^{n} \frac{x_i^{-\alpha}}{\zeta(\alpha, x_{\min})}, \tag{3.18}$$

where x_i $(i = 1, 2, \ldots, n)$ are the independently observed values of x and $x_i \geqslant x_{\min}$. Taking natural logarithms of both sides of the likelihood function, it is turned into

[3]The details of the issues, which are out of the scope of this chapter, are discussed in [30].

a sum:

$$\begin{aligned}\Lambda = \ln(\mathcal{L}) &= \ln\left(\prod_{i=1}^{n}\frac{x_i^{-\alpha}}{\zeta(\alpha, x_{\min})}\right) \\ &= -\alpha\sum_{i=1}^{n}\ln(x_i) - n\ln\left(\zeta(\alpha, x_{\min})\right). \end{aligned} \tag{3.19}$$

Maximizing Λ by setting $\frac{\partial\Lambda}{\partial\alpha} = 0$, gives:

$$\sum_{i=1}^{n}\ln(x_i) = -\frac{n}{\zeta(\alpha, x_{\min})}\frac{\partial}{\partial\alpha}\zeta(\alpha, x_{\min}) \ . \tag{3.20}$$

Thus, the estimated value $\hat{\alpha}$ for the scaling exponent parameter α by using MLE on the observed values x_i of x can be found by solving the equation

$$\frac{\zeta'(\hat{\alpha}, x_{\min})}{\zeta(\hat{\alpha}, x_{\min})} = -\frac{1}{n}\sum_{i=1}^{n}\ln(x_i) \ , \tag{3.21}$$

where $\zeta'(\hat{\alpha}, x_{\min})$ denotes the derivative function of $\zeta(\hat{\alpha}, x_{\min})$ with respect to $\hat{\alpha}$.

The form of the Hurwitz zeta function means that it is hard to calculate the exact theoretical solution for Equation 3.21. However, it can be approximately solved by using numerical analysis and other estimation methods. For instance, by using the idea that power-law distributed integers can be approximated as continuous power-law distributed real numbers rounded to their nearest integers, Clauset et al. [30] gave an approximate formula of Equation 3.21:

$$\frac{\zeta'(\hat{\alpha}, x_{\min})}{\zeta(\hat{\alpha}, x_{\min})} = \left[\frac{1}{\hat{\alpha}-1} - \ln\left(x_{\min} - \frac{1}{2}\right)\right]\left[1 + O(x_{\min}^{-2})\right] + O(x_{\min}^{-2}). \tag{3.22}$$

When $x_{\min}$ is large,

$$\hat{\alpha} \simeq 1 + n\left[\sum_{i=1}^{n}\ln\left(\frac{x_i}{x_{\min} - \frac{1}{2}}\right)\right]^{-1} \tag{3.23}$$

gives a relatively good approximate solution to Equation 3.21.

As the sample size n increases, the distribution of the maximum likelihood estimators to α for the same power-law distribution would be closer to the normal distribution with mean α. The corresponding standard error σ of any estimator $\hat{\alpha}$

of α, which is obtained by using MLE, is given by:

$$\begin{aligned}\sigma = \sqrt{\frac{1}{I(\alpha)}} &= \sqrt{\frac{1}{-E[\frac{\partial^2 \Lambda}{\partial \alpha^2}]}} \\ &= \frac{1}{\sqrt{n\left[\frac{\zeta''(\hat{\alpha},x_{\min})}{\zeta(\hat{\alpha},x_{\min})} - \left(\frac{\zeta'(\hat{\alpha},x_{\min})}{\zeta(\hat{\alpha},x_{\min})}\right)^2\right]}},\end{aligned} \tag{3.24}$$

where $I(\alpha)$ is the Fisher information. When n and $x_{\min}$ are reasonable large (i.e., when $n \gtrsim 50$ and $x_{\min} \gtrsim 6$, the average error is smaller than 1% of the typical value of α [30]), the standard error σ can be approximately computed by:

$$\sigma = \frac{\hat{\alpha} - 1}{\sqrt{n}} + O\left(\frac{1}{n}\right) , \tag{3.25}$$

which is the equation for calculating the standard error on $\hat{\alpha}$ of α for a continuous power-law distribution.

Estimating the lower bound parameter

It is important to select the correct value of the lower bound parameter $x_{\min}$ for reliably estimating the scaling exponent and characterizing the target power-law distribution. When the estimated value $\hat{x}_{\min}$ for $x_{\min}$ is smaller than the value of $x_{\min}$, some data in the empirical data, which does not follow the power-law distribution, is brought into the estimation of α. That can generate a biased $\hat{\alpha}$. When $\hat{x}_{\min}$ is larger than $x_{\min}$, some useful data will not be included, which reduces the valid sample size n and increases the standard error σ of $\hat{\alpha}$.

There are some methods for selecting the value of $x_{\min}$ in the discrete case. However, they have some disadvantages, such as being too subjective, sensitive to fluctuation, or having too many parameters involved. To overcome these disadvantages, Clauset et al. [30] suggested choosing the $\hat{x}_{\min}$ that maximizes the similarity between the best-fit power-law distribution and that of the empirical data where x is not less than the selected $\hat{x}_{\min}$.

One very simple and common measure for quantifying the distance between two statistical distributions is the Kolmogorov-Smirnov test or KS-test. A KS-test computes a D-statistic, which is the maximum absolute difference between the two cumulative distributions of the empirical distribution and the expected distribution. The KS-test does not make any assumption about the distribution that the empirical

data is drawn from (i.e., it is non-parametric or distribution free) [96].

To estimate power-law distributions, the D-statistic is:

$$D_m(x_{\min}) = \max_{x \geqslant x_{\min}} |S_m(x) - P(x)|, \tag{3.26}$$

where $S_m(x)$ denotes the cumulative distribution of the empirical distribution with m observed empirical data points in the interval $[x_{\min}, +\infty)$, and $P(x)$ denotes the hypothesized cumulative power-law distribution in the interval $[x_{\min}, +\infty)$. Clauset et al. [30] suggested choosing the value of $x_{\min}$ as the $\hat{x}_{\min}$ that minimizes $D(x_{\min})$.

To empirically calculate the uncertainty of the estimate of the lower bound parameter of the power-law distribution best fitted to the empirical data, Clauset et al. [30] suggested applying a nonparametric bootstrap method in this application:

1. Draw data uniformly at random from the empirical data set to compose a synthetic data set that has a similar distribution to the empirical one and the same dataset size to the empirical one

2. Use the combination of the MLE and KS-test to estimate $x_{\min}$ and α of the best-fit power-law distribution to the synthetic data set

3. Repeat Steps 1–2 a large number of times and collect the estimates of $x_{\min}$ and α respectively

4. The uncertainty of the estimate of $x_{\min}$ of the best-fit power-law distribution to the empirical data can be approximately estimated by the standard deviation of the estimates of $x_{\min}$ collected in the previous step. The uncertainty of the estimate of α can be empirically estimated in the same way.

Through a series of simulation tests and comparisons to other techniques, Clauset et al. [30] showed that using the combination of MLE and the KS-test can make relatively accurate and reliable estimation on the parameters of the best-fit power-law distribution to a target set of empirical data.

3.2.4 Verification of a Power-law Relationship in a Set of Empirical Data

The previous section reviews how to correctly find the discrete power-law relationship best fitted to a target set of empirical data. However, it does not tell whether

the best-fit power-law relationship is a good model for the target set of empirical data. This section reviews how to use quantitative goodness-of-fit tests based on D-statistic and likelihood ratio tests to verify whether a target set of empirical data follows a power-law distribution.

A straight line in a log-log plot is often seen as a sign of a power-law relationship. However, it does not justify the existence of the power-law relationship in practice, since finite amounts of data drawn from other relationships can display similar signs. Only their asymptotic limit can identify that these data does not follow a power-law relationship. Moreover, noise often exists in real empirical data and can affect observation. In practice, showing a roughly straight line usually replaces showing a precisely straight line as the sign. This increases the uncertainty that the data truly follows a power-law relationship. Therefore, the straight line shown by real empirical data is only a necessary condition, but not the sufficient condition of validating the real empirical data satisfying a power-law relationship.

A quantitative goodness-of-fit test is necessary and sufficient to verify whether a set of empirical data follows a power-law distribution. A lot of work in the literature, which alleges that they have found the power-law distributed empirical data based on qualitative appraisals of their empirical data instead of quantitative ones, can lead to erroneous judgments. Besides, qualitative appraisal cannot indicate how well the empirical data is fitted by a power-law distribution. A qualitative appraisal can only deliver a power-law hypothesis, which needs to be further verified by quantitative goodness-of-fit tests.

D-statistic tests

D-statistic tests were suggested by Clauset et al. [30] to quantitatively test how well empirical data is fitted by a power-law distribution and whether the empirical data follows a power-law distribution. The value of the D-statistic is computed by the KS-test and expresses the distance between the empirical distribution and the fitted power-law distribution. Also, the accompanying p-value quantitatively measures the probability that the empirical data is drawn from the fitted power-law distribution. The p-value indicates the statistical significance of the computed D-statistic. It can represent one of two results: either the empirical data is not drawn from the fitted power-law distribution, or the empirical data drawn from the fitted power-law distribution cannot be proven to be false at the significance level represented by the p-value. Therefore, strictly statistically speaking, for the task of proving that the empirical data is drawn from a fitted power-law distribution, the

best that can be stated by using D-statistic tests is to declare that it cannot be proven that the empirical data is not drawn from the fitted power-law distribution.

It should also be noticed that D-statistic tests examine how well the empirical data matches the fitted or tested power-law distribution. There is no guarantee that the best-fit power-law distribution is the true one that generates the empirical data observed. The reason is that all tests and statistics adopted here are computed based on the instances of the true distribution: a finite number of instances as a sample set drawn from the infinite population generated by the true distribution. Some factors, such as statistical fluctuations created during sampling, can influence the results of fitting tests. Therefore, the best-fit distribution here is often not equal to the true one.

There is no explicit expression of the p-value as a function of the D-statistic, when the $P(x)$ used in the KS-test (Equation 3.26) is the best-fit distribution of the empirical data used as the $S_m(x)$ in the KS-test. If the empirical data were truly drawn from the hypothesized distribution $P(x)$, $\sqrt{m}D_m$ converges to the Kolmogorov distribution [59]. Based on that, the explicit expression of the p-value as a function of the D-statistic can be obtained. However, as explained before, the $P(x)$ used here is the best-fit distribution of the empirical data observed before $x_{\min}$, which distinctly depends on the empirical data observed before $x_{\min}$. Therefore, the standard rule for calculating the p-value is not applicable in this case [30, 43].

In general, for cases like this where the theoretical values of the p-value cannot be calculated, the empirical p-value can be computed by using Monte Carlo procedures [76, 140]. Considering the situation where only the empirical data points in the interval $[x_{\min}, +\infty)$ follow its best-fit power-law distribution, the Monte Carlo procedure suggested by Clauset et al. [30] for numerically computing the empirical values of p-value works as follows:

1. Determine the best-fit power-law distribution $p(x)$ of the empirical data points observed, estimate the corresponding scaling exponent parameter α and the lower bound parameter $x_{\min}$ by using the MLE and KS-test.

2. Calculate the D-statistic as the goodness of fit of $p(x)$ to the empirical data points.

3. Generate a large number of synthetic datasets by using a semi-parametric approach. Suppose that the total number of the empirical data points observed over x is n, which includes m empirical data points observed in the interval

$[x_{\min}, +\infty)$. To generate a synthetic dataset, which has a similar non-power-law distribution to the empirical one observed below $x_{\min}$ and exactly follows $p(x)$ in $[x_{\min}, +\infty)$, the following steps are implemented.

(a) With probability m/n, a number x_i is randomly drawn from $p(x)$ in $[x_{\min}, +\infty)$. With probability $(1 - m/n)$, a data point is uniformly randomly selected from the set of empirical data observed below $x_{\min}$ and set x_i equal to it.

(b) Repeating Step (a) n times, a data set with n synthetic data points is generated, which satisfies the requirement mentioned above.

4. Fit the power-law distribution to each synthetic data set based on the same method that determines $p(x)$ of the empirical data, and calculate the D-statistic for each best-fit power-law distribution to its own synthetic data set.

5. Count the empirical value of the p-value as the fraction of the D-statistic values calculated for the synthetic data sets, which are larger than the value of the D-statistic calculated for the empirical data.

6. If the empirical value of the p-value is sufficiently small, it means that the empirical data is unlikely to be drawn from a power-law distribution.

The number of synthetic data sets generated to compute the empirical value of the p-value affects the accuracy of the computation. Generally, the larger the number of synthetic data sets, the more accurate the empirical value computed for the p-value can be. A rule of thumb provided by Clauset et al. [30] is:

$$N \geqslant \frac{1}{4}\epsilon^{-2}, \tag{3.27}$$

where $\epsilon = |\hat{p}_{\text{value}} - p_{\text{value}}|$, p_{value} denotes the true value of the p-value, $\hat{p}_{\text{value}}$ denotes the empirical value computed for the p-value, and N denotes the number of the synthetic data sets generated.

In addition, if the size n of the empirical data is small, a larger empirical data set (if possible) or a larger number of synthetic data sets should be used to improve the accuracy of the computation. As the size n becomes larger, the statistical variation of the computed D-statistics becomes smaller and the computed empirical value of the p-value becomes more reliable. When the size n is small, the chance of having high p-values is higher, even if the fitted distribution is the wrong one for

the empirical data. In this case, it simply reflects the uncertainty of the computed results based on small data sets.

Competing tests

It is not enough to verify that a set of empirical data follows a power-law distribution in the interval $[x_{\min}, +\infty)$ based only on the tests above. D-statistic tests only examine how well the best-fit power-law distribution fits to a target set of empirical data. Thus, even with a large empirical value of the p-value, one can only be sure that the power-law distribution has a good chance of being the one that actually generated the empirical data at a certain statistical significance level. However, it does not exclude the possibility that there is another distribution fitting to the target set of empirical data with a higher p-value or having a lower value of $x_{\min}$ when the p-values are equal. Consequently, further tests on other competing distributions need to be implemented.

The further tests can be implemented as described above [30], but replacing the power-law distribution with another distribution. The competing distribution is fitted to the empirical data in order to identify the best-fit values for the parameters and compute the D-statistic for the fit to the empirical data. Then, a large number of synthetic data sets are generated and the empirical value of the p-value for each synthetic data set is computed. If the empirical values of the p-value by using non-power-law distributions are smaller than those found by using the power-law distribution or with equal p-values the values of $x_{\min}$ estimated based on the non-power-law distribution is larger than the one based on the power-law distribution, one has more evidence to support the conjecture that the empirical data is likely to be drawn from a power-law distribution. Note that even if this situation happens, one still cannot be sure that the empirical data is truly drawn from the power-law distribution. That only means that it can still remain its status as a reasonable hypothesis.

To keep a fair comparison, one should compare apples to apples by using the same number of parameters for each fitted distribution, the same number of synthetic data sets, and the same fitting and testing methods.

In a statistical hypothesis test, a reasonable hypothesis needs to be decided in advance. In general, a distribution can show a better fit by using more parameters in the distribution. Also, a model can show a better fit by increasing the number of the distributions in the model. This does not mean that one should fit the distribution with more parameters to the empirical data to get a better match. One reason is that

if the number of the parameters in the fitted distribution is too large, the problem of over-fitting occurs. The fitted distribution attempts to fit the random error or noise as well as the model. This also applies when fitting a model to the empirical data. Note that there are infinitely many competing distributions. Obviously, it is impossible to implement further tests on every possible competing distribution. Hence, in practice, the question becomes which competing distributions should be tested, which depends on the applications.

To reduce the computational workload, Clauset et al. [30] suggested using likelihood ratio tests instead of D-statistic tests for examining competing distributions. D-statistic tests are more likely to measure the absolute goodness of fit, while likelihood ratio tests are more likely to measure the relative goodness of fit. Thus, likelihood ratio tests cannot tell how well the two competing distributions fit to the empirical data, unlike D-statistic tests. Likelihood ratio test can only tell that one of the two competing distributions is a relatively better fit to the empirical data. Therefore, to actually know how well a competing distribution fits to the empirical data, D-statistic tests are still required at least once. Since one of the purposes of this study is to verify whether a power-law relationship exists in the pattern support distribution drawn from a transaction dataset, it is necessary to implement D-statistic tests on the power-law distribution best fitted to a target pattern support distribution.

A likelihood ratio test computes the ratio of the maximum likelihood of the empirical data to one distribution and the empirical data to another competing distribution. The maximum likelihood of the empirical data to a tested distribution can be calculated by maximizing its corresponding likelihood function, which represents the maximum probability of the empirical data actually being drawn from the tested distribution. Hence, the better-fit distribution has a higher value of the maximum likelihood. Also, the likelihood ratio test indicates which of the two competing distributions is a better fit to the empirical data by checking whether the calculated likelihood ratio is greater or less than 1, or whether the calculated value is positive or negative if applying the logarithm on the likelihood ratio (i.e., the log likelihood ratio $\mathcal{R}$).

Note that the value of the likelihood suffers from statistical fluctuation. If the expected value of the maximum likelihood of one competing distribution to the empirical data is very close to that of the other competing distribution, the calculated result of the likelihood ratio test, which is close to 0, is not quite reliable, as the

statistical fluctuation can easily change the difference between the two calculated values of the maximum likelihood. Therefore, to make sure whether the returned result of the likelihood ratio test indicates the better fit, one needs to know whether there is a sufficient difference between the calculated values of the two maximum likelihoods. The size of the expected fluctuation, i.e., the standard deviation on the likelihood ratio, can quantitatively measure this.

Clauset et al. [30] suggested using the following method first proposed by Vuong [119] to estimate the standard deviation on $\mathcal{R}$. Suppose that there are two competing distributions $p_1(x)$ and $p_2(x)$ used in the likelihood ratio test. Given an empirical data set with n independent observations, the likelihood functions of $p_1(x)$ and $p_2(x)$ respectively are:

$$\mathcal{L}_1 = \prod_{i=1}^{n} p_1(x_i) \tag{3.28}$$

and

$$\mathcal{L}_2 = \prod_{i=1}^{n} p_2(x_i) \ . \tag{3.29}$$

The corresponding likelihood ratio is

$$\frac{\mathcal{L}_1}{\mathcal{L}_2} = \prod_{i=1}^{n} \frac{p_1(x_i)}{p_2(x_i)} \ . \tag{3.30}$$

Taking the natural logarithm, it becomes the log likelihood ratio, i.e.,

$$\mathcal{R} = \sum_{i=1}^{n} \left[\ln(p_1(x_i)) - \ln(p_2(x_i))\right] \ . \tag{3.31}$$

Since each x_i is independent observed, the value of each $[\ln(p_1(x_i)) - \ln(p_2(x_i))]$ is independent. By the central limit theorem, as n becomes sufficiently large, the sum $\mathcal{R}$ of the independent values $[\ln(p_1(x_i)) - \ln(p_2(x_i))]$ approximately approaches the normal distribution with expected variance $n\sigma^2 > 0$, where σ^2 is the expected variance of $[\ln(p_1(x_i)) - \ln(p_2(x_i))]$. Note that the value of σ^2 is unknown in this case. A common method is to approximate σ^2 by the variance s_n^2 of the n empirical data:

$$\sigma^2 \approx s_n^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\left(\ln\left(p_1(x_i)\right) - \ln\left(p_2(x_i)\right)\right) - \frac{1}{n}\sum_{j=1}^{n}\left(\ln\left(p_1(x_j)\right) - \ln\left(p_2(x_j)\right)\right)\right]^2 . \tag{3.32}$$

Therefore, the probability that the absolute value of the expected log likelihood ratio is not less than the absolute value $|\mathcal{R}|$ of the calculated empirical log likelihood ratio is

$$\begin{aligned} p &= \frac{1}{\sqrt{2\pi n\sigma^2}}\left[\int_{-\infty}^{-|\mathcal{R}|} e^{-t^2/(2n\sigma^2)}dt + \int_{|\mathcal{R}|}^{+\infty} e^{-t^2/(2n\sigma^2)}dt\right] \\ &= \left|\mathrm{erfc}\left(\frac{\mathcal{R}}{\sqrt{2n\sigma^2}}\right)\right|, \end{aligned} \tag{3.33}$$

where σ is calculated via Equation 3.32 and erfc denotes the complementary Gaussian error function, i.e.,

$$\mathrm{erfc}(z) = 1 - \mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_{z}^{+\infty} e^{-t^2}dt. \tag{3.34}$$

The calculated p value is an estimate of the probability that the measured value of $\mathcal{R}$ is calculated based on two likelihoods without a sufficient difference between them. Consequently, if the p value is too small (e.g., $p < 0.1$), it cannot reject the hypothesis that the calculated value of $\mathcal{R}$ can be trusted to tell which one of the two competing distributions used for calculating $\mathcal{R}$ is the better fit to the empirical data. However, if p is large, the calculated value of $\mathcal{R}$ cannot be trusted and the likelihood ratio test fails. When this happens, one could use more empirical data if possible.

Moreover, if the competing distributions are very close to the true one that generates the empirical data, the method introduced above does not work, even if more empirical data is used. In cases like this, the value of each $[\ln(p_1(x_i)) - \ln(p_2(x_i))]$ in Equation 3.31 and their variance σ^2 all tend to 0. The central limit theorem is not applicable in this case. Actually, in this case, $\mathcal{R}$ tends to follow a chi-squared distribution when n becomes larger. The correct p value can be calculated based on the chi-squared distribution [2].

3.3 Power-law-based Pattern Support Distributions

In this section, the partial pattern support distributions of five real transaction datasets have been computed and observed. After finding the best-fit discrete power-law distributions to the five partial pattern support distribution, a compari-

son between the (cumulative) pattern support distributions and their corresponding best-fit discrete power-law distributions is made in order to set the reasonable hypothesis that power-law distributions exist in the pattern support distributions of real retail transaction datasets. And then, by extending the approach suggested by Clauset et al. [30], this section proposes a new method of utilizing the bootstrap method and the universality of power-law relationships to quantitatively verify the hypothesis. At the end of this section, the self-similarity phenomenon linked to the power-law-based pattern support distributions is also explored.

3.3.1 Observations on Pattern Support Distributions

Several real datasets are investigated to observe the behaviour of pattern support distributions of real transaction datasets. All of them are widely used in the research field of data mining as benchmark datasets[4]. The parameters of these datasets are summarized in Table 3.1, where $|I|$ indicates the number of items in the dataset, $|TD|$ indicates the number of transactions, $|T|_{\max}$ indicates the maximum length of transactions, and $|T|_{\text{avg}}$ indicates the average length of transactions.

Dataset	$\|I\|$	$\|TD\|$	$\|T\|_{\max}$	$\|T\|_{\text{avg}}$
Retail	16,470	88,162	76	10.3
BMS-POS	1,657	515,597	164	6.5
Adult	29,035	48,842	15	15
Accidents	468	340,183	51	33.8
Pumsb	2,113	49,046	74	74

Table 3.1: Parameters of the five real transaction datasets used

The Retail dataset [18] contains a retail market basket dataset from an anonymous retail store. The BMS-POS dataset [61] contains several years of sales data from an anonymous electronics retailer. Each transaction in the dataset contains all the product categories instead of individual products purchased by a customer at one time. The Adult dataset [60] was extracted from the 1994 U.S. Census Bureau databases. Each transaction contains some properties of an adult, such as age, education and hours of work per week. The Accidents dataset [42] contains anonymous traffic accidents data obtained from the National Institute of Statistics (NIS)

[4]They are available from the Frequent Itemset Mining Implementations Repository (`http://fimi.cs.helsinki.fi/data/`) or the UCI Machine Learning Repository (`http://www.ics.uci.edu/mlearn/MLRepository.html`).

for the region of Flanders (Belgium) for the period 1991-2000. Each transaction in the dataset contains some information regarding to a traffic accident recorded by a police officer, such as cause of the accident and traffic conditions. The Pumsb dataset [13] is census data for population and housing extracted from PUMS (Public Use Microdata Sample)[5].

Note that there are at most $2^{|I|}-1$ ($= \sum_{i=1}^{|I|} \mathrm{C}_i^{|I|}$) possible patterns in a transaction dataset that has $|I|$ distinct items. Since, generally, the total number of distinct items in a real transaction dataset is large, mining all existing patterns and their corresponding supports in a reasonable time is often not practical. For the sake of efficiency, only the patterns whose supports are not less than a support threshold value (denoted as min_sup_1) for retrieving patterns, are mined. Ideally, this support threshold value should be as close to 1 as possible, since 1 is the minimum support value of all patterns existing in a dataset. Thus, considering the factor of execution time, the support threshold value for mining patterns was chosen for individual datasets, and is listed in Table 3.2. The program for retrieving the patterns and their corresponding supports in a dataset is derived from the FP-growth algorithm[6] [17].

Dataset	min_sup_1
Retail	2
BMS-POS	50
Adult	2
Accidents	5,000
Pumsb	21,000

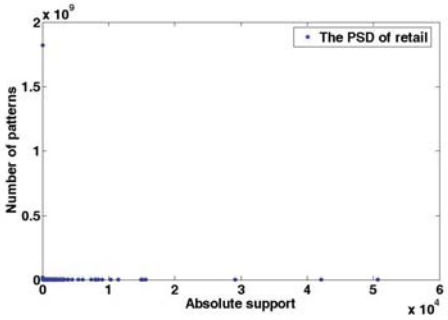
Table 3.2: The support threshold values for mining patterns in the five real transaction datasets

Based on these mined patterns and their supports, Figure 3.2 shows the partial pattern support distributions of the five real transaction datasets by plotting the number of patterns and their corresponding support discovered from each dataset. Figure 3.2 presents the original curve of each pattern support distribution in a normal 2D plot and the corresponding one in the natural log-log plot. As can be seen from log-log plots, each pattern support distribution appears as a roughly straight line. Also, explicit fluctuation exists in the higher support (right side) part of each pattern support distribution. For unknown reasons, the fluctuation in the pattern support distribution of the Pumsb dataset has a stronger appearance than

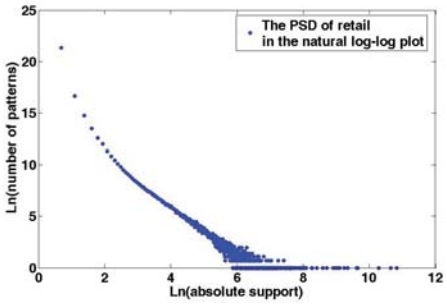
[5] `http://www.ciesin.org/datasets/pums/pums-home.html`

[6] It is available from `http://www.borgelt.net/fpgrowth.html`.

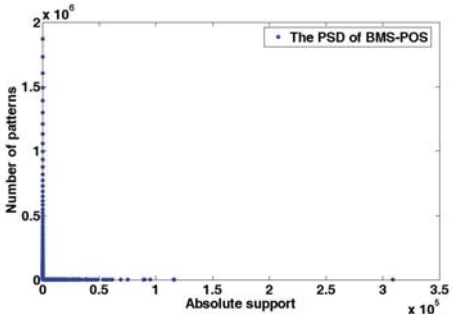
the ones in the pattern support distributions of other four datasets.



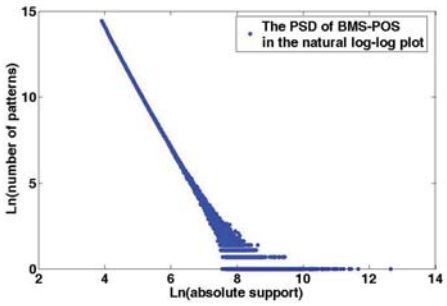
(a) The PSD of Retail



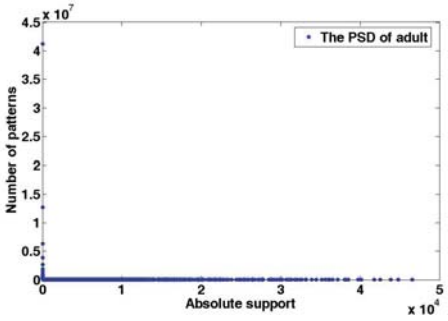
(b) The PSD of Retail in the natural log-log plot



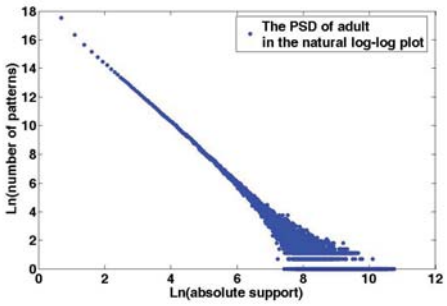
(c) The PSD of BMS-POS



(d) The PSD of BMS-POS in the natural log-log plot



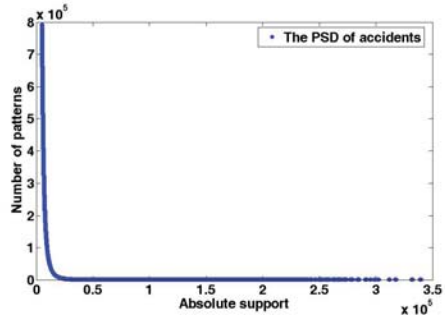
(e) The PSD of Adult



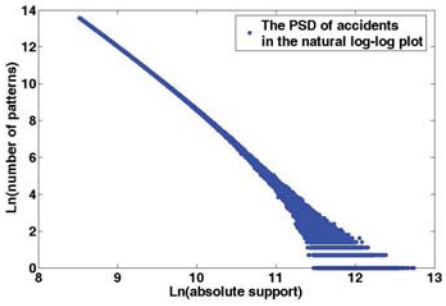
(f) The PSD of Adult in the natural log-log plot

Figure 3.2: The partial pattern support distributions of five real transaction datasets

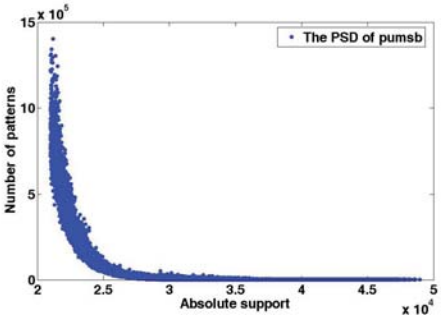
The observation conforms to the one detected by Chuang et al. [28, 29]. Based on the observation, Chuang et al. asserted the existence of the power-law relationship in pattern support distributions. However, as mentioned previously, the qualitative appraisal based on visualization—the power-law relationship shows a roughly straight line in a log-log plot—can only validate the necessary condition for the power-law relationship, but not the sufficient condition. Thus, differing from



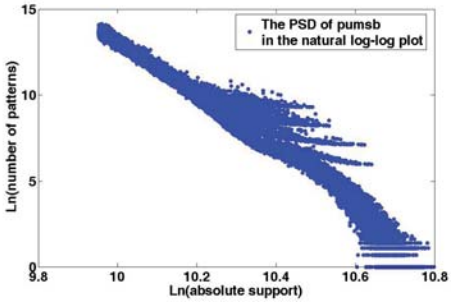
(b) The PSD of Accidents



(c) The PSD of Accidents in the natural log-log plot



(d) The PSD of Pumsb



(e) The PSD of Pumsb in the natural log-log plot

Figure 3.2: The partial pattern support distributions of five real transaction datasets (con't)

the work [28, 29], the existence of the power-law relationship in pattern support distributions is further investigated by using statistical techniques in the following sections.

3.3.2 Fitting Discrete Power-law Distributions to Pattern Support Distributions

This section aims to fit a discrete power-law distribution to the pattern support distributions drawn from the five real transaction datasets listed in Table 3.1. More specifically, a best-fit discrete power-law distribution is found for each partial pattern support distribution drawn from the five real transaction datasets by using the KS-test and maximum likelihood estimation, which are introduced in the fitting part of Section 3.2. This section applies these statistical techniques in the study of pattern support distributions and makes a hypothesis based on the experimental results.

The fitting procedure based on the overview and discussion in the fitting part of

Section 3.2 has three steps:

1. Each empirical pattern support distribution drawn from a target transaction dataset is transformed into a set of mined pattern support values as preprocessing for the following calculation.

2. For each possible choice of the lower bound parameter $x_{\min}$, the corresponding value of the scaling exponent parameter α of the best-fit discrete power-law distribution is estimated by using the MLE. The corresponding value of the D-statistic is also computed via a KS-test.

   Note that a KS-test requires working on the complementary cumulative distribution instead of its underlying discrete power-law distribution. Also, the value of α calculated by using the MLE is the one of the best-fit discrete power-law distribution, not its complementary cumulative one. However, based on the discussion in Section 3.2, the complementary cumulative distribution can be easily computed by using its underlying discrete power-law distribution.

3. The estimator $\hat{x}_{\min}$ of $x_{\min}$ chosen by the fitting procedure is the one that has the smallest D-statistic value compared with all other possible values of $x_{\min}$ tested in the previous step.

Note that even though the pattern support threshold value min_sup_1 was used for the mining procedure, the number of pattern support values mined from any one of the five real transaction datasets listed in Table 3.2 is still very huge. Also, the fitting procedure is very computationally costly too, since it is supposed to try each possible value of $x_{\min}$. Therefore, in practice, it is quite difficult to get (relatively) accurate results within a reasonable time by inputting all of the mined pattern support values. For the sake of execution time, it seems that a compromise needs to be made. There are at least four methods of making the compromise here:

1. Set a higher support threshold value (denoted as min_sup_2) and abandon the mined support values that are less than min_sup_2 in order to reduce the set size of pattern support value for the computation

2. Randomly draw a relatively large sample set of transactions from each one of the five real transaction datasets, and implement the computation on the pattern support distribution of the sample instead of that of the corresponding real dataset

3. Randomly draw a relatively large sample set of pattern support values from one of the five mined support values sets, and implement the computation on the sample set instead of the corresponding set of mined support values

4. Reduce the accuracy of the estimated values on the parameters of the best-fit power-law distribution: i) reduce the size of the possible value set of x_{min}; or ii) reduce the size of the possible value set of α used in testing

Note that the purpose of the fitting procedure here is to fit the discrete power-law distribution to the whole of a pattern support distribution drawn from each of the five real transaction datasets, so that later one can use the estimated results to verify whether the power-law relationship exists in the pattern support distribution of a real transaction dataset. Unfortunately, any one of the four methods can more or less reduce the accuracy of the estimated results, which may cause incorrect results on the verification tests later. Considering the consequences by implementing any one of the four methods and the corresponding computation involved, method 1 was implemented here with the utmost caution on the set value of min_sup_2.

Table 3.3 gives several generic statistics of the data sets of pattern support values, which are used by the fitting procedure.

Dataset	min_sup_2	n	mean(sup_n)	stdev(sup_n)	max(sup_n)
Retail	3	20,647,331	3.5345	18.9484	50,675
BMS-POS	63	18,878,499	105.3433	160.4050	308,656
Adult	6	16,390,449	29.9253	165.2179	46,560
Accidents	27,701	20,353,849	3.9173e+04	1.4105e+04	340,151
Pumsb	29,247	20,839,076	3.1783e+04	2.1759e+03	48,944

Table 3.3: Basic parameters of the pattern support value sets used by the fitting procedure, where n is the total number of pattern support values in a dataset, mean(sup_n) is the mean of the pattern support values in a dataset, stdev(sup_n) is the standard deviation of the pattern support value in a dataset, and max(sup_n) is the maximum value of the pattern support values in a dataset

Table 3.4 shows the best-fit discrete power-law distribution to each of the five datasets of pattern support values in Table 3.3 by using the combination of the MLE and KS-test. Note that, as introduced in Section 3.2, Equation 3.25 approximates to Equation 3.24 when $x_{\min}$ and m are sufficient large.

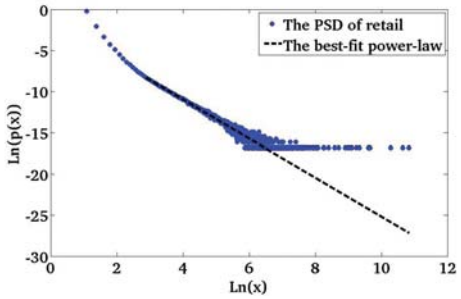
Figure 3.3 shows the normalized partial pattern support distributions drawn from the five real transactions datasets, and the best-fit discrete power-law probability

Dataset	$\hat{x}_{\min}$	$\hat{\alpha}$	$\hat{\sigma}_1$	$\hat{\sigma}_2$	m	Λ
Retail	17	2.37	0.0051	0.0051	73,098	-3.0813e+05
BMS-POS	329	3.20	0.0038	0.0038	334,076	-2.1594e+06
Adult	6	2.07	2.6544e-04	2.6429e-04	16,390,449	-5.8493e+07
Accidents	27,740	4.3161	9.5852e-04	7.3658e-04	20,267,959	-2.0944e+08
Pumsb	41,251	36.6511	0.2080	0.2080	29,374	-2.3739e+05

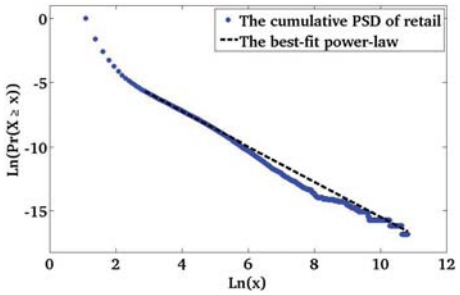
Table 3.4: The best-fit discrete power-law distributions to the pattern support value sets in Table 3.3, where $\hat{x}_{\min}$ is the estimate of the lower bound parameter $x_{\min}$ of the best-fit power-law distribution, $\hat{\alpha}$ is the estimate of the scaling exponent parameter α of the best-fit power-law distribution, $\hat{\sigma}_1$ and $\hat{\sigma}_2$ are the standard deviations of the estimate of α of the best-fit discrete power-law distribution based on Equations 3.24 and 3.25 respectively with an assumption that $\hat{x}_{\min}$ is equal to $x_{\min}$ required by Equations 3.24 and 3.25, m is the corresponding number of the mined pattern support values that are in the interval $[\hat{x}_{\min}, +\infty)$, and Λ is the natural logarithm of the corresponding maximum likelihood computed based on $\hat{x}_{\min}$ and $\hat{\alpha}$

distribution for each of them; their parameters are listed in Table 3.4. Figure 3.3 also gives the normalized cumulative distributions of these partial pattern support distributions and the best-fit discrete power-law probability distributions to the partial pattern support distributions. Generally, the visual form of the cumulative distribution is more robust than that of the discrete power-law distribution due to fluctuations cased by the limit of sample sizes, particularly in the lower tail of the distribution where fewer patterns appear, which can be clearly seen in Figure 3.3.

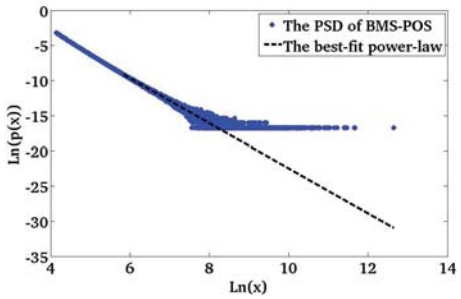
Although each pattern support distribution of the five datasets roughly follows a straight line in the interval $[\hat{x}_{\min}, +\infty)$ in a log-log plot, their corresponding cumulative distributions show different behaviours in log-log plots. The cumulative distributions of the pattern support distributions of the Retail and BMS-POS datasets more closely follow straight lines representing their corresponding maximum likelihood best-fit power-law distributions in the interval $[\hat{x}_{\min}, +\infty)$ in log-log plots, while the cumulative distributions of the pattern support distributions of the Adult, Accidents and Pumsb datasets do not closely follow their corresponding straight lines, and the divergence increases when the support value increases. It implies that the pattern support distributions of the Retail and BMS-POS datasets may follow power-law distributions, while the pattern support distributions of the Adult, Accidents and Pumsb datasets may not follow power-law distributions. Since the Retail and BMS-POS datasets are real retail transaction datasets, a reasonable hypothesis that power-law distributions exist in the pattern support distributions of real retail



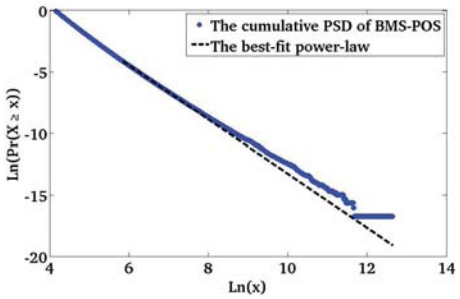
(a) The PSD of Retail and maximum likelihood power-law fit



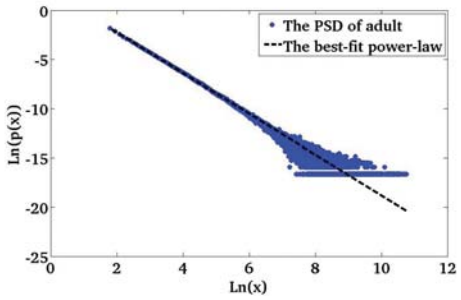
(b) The cumulative PSD of Retail and maximum likelihood power-law fit



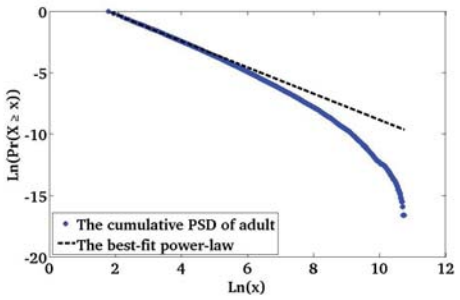
(c) The PSD of BMS-POS and maximum likelihood power-law fit



(d) The cumulative PSD of BMS-POS and maximum likelihood power-law fit



(e) The PSD of Adult and maximum likelihood power-law fit

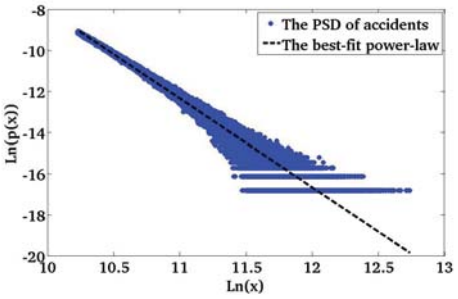


(f) The cumulative PSD of Adult and maximum likelihood power-law fit

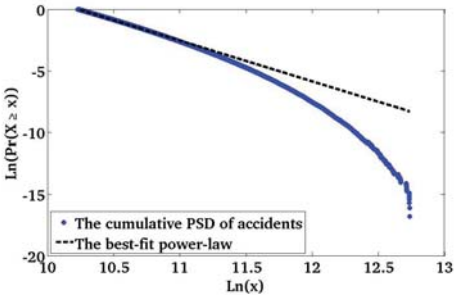
Figure 3.3: The normalized partial pattern support distributions of five real transaction datasets and their maximum likelihood discrete power-law fits in natural log-log plots

transaction datasets can be established, which will be further examined in the next section.

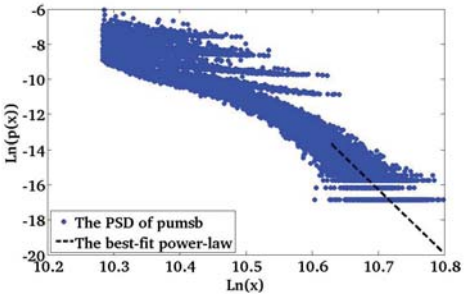
Table 3.5 shows the empirical values related to the uncertainty of the estimates of the lower bound and scaling exponent parameters of the best-fit discrete power-law distributions listed in Table 3.4 to the pattern support value sets in Table 3.3. mean($\hat{x}_{\min}$) and stdev($\hat{x}_{\min}$) are the mean and standard deviation of the estimates of



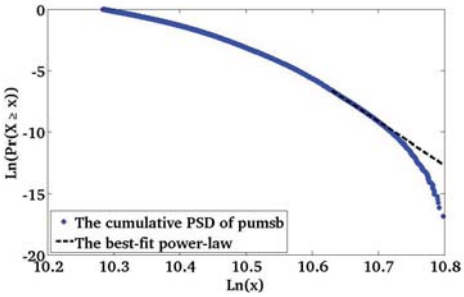
(b) The PSD of Accidents and maximum likelihood power-law fit



(c) The cumulative PSD of Accidents and maximum likelihood power-law fit



(d) The PSD of Pumsb and maximum likelihood power-law fit



(e) The cumulative PSD of Pumsb and maximum likelihood power-law fit

Figure 3.3: The normalized partial pattern support distributions of five real transaction datasets and their maximum likelihood discrete power-law fits in natural log-log plots (con't)

x_{min} of the best-fit power-law distribution to a pattern support value set. mean($\hat{\alpha}$) and stdev($\hat{\alpha}$) are the mean and standard deviation of the estimates of α of the best-fit power-law distribution to a pattern support value set. mean(m) and stdev(m) respectively are the corresponding mean and standard deviation of the number of the pattern support values that are not less than $\hat{x}_{\min}$ in a pattern support value set.

Each mean and standard deviation were computed based on the nonparametric bootstrap method introduced in Section 3.2 with $1,000$ repetitions, except the Accidents and Pumsb datasets. Note that the number of repetitions should be as large as possible, since mean($\hat{x}_{\min}$) and mean($\hat{\alpha}$) respectively tend to the true parameter values of the power-law distribution in the process of generating each pattern support value set if the power-law distribution truly exists. With the consideration of corresponding computational costs, $1,000$ is here considered as a sufficiently large number for the number of repetitions to show the uncertainty of $\hat{x}_{\min}$ and $\hat{\alpha}$ esti-

Dataset	mean($\hat{x}_{\min}$)	stdev($\hat{x}_{\min}$)	mean($\hat{\alpha}$)	stdev($\hat{\alpha}$)	mean(m)	stdev(m)
Retail	20.0	16.8	2.3746	0.0410	68,813	10,909
BMS-POS	495.0	272.8	3.1585	0.0719	289,580.9	269,993.0
Adult	6.7	0.9531	2.0735	0.0048	14,808,240.5	2,166,781.9
Accidents	32,251.9	25,240.4	4.4596	0.8032	19,466,606.2	3,493,384.9
Pumsb	41,207.8	95.2	36.6901	0.2463	30,629.8	2,610.1

Table 3.5: The computed uncertainty on the estimates of the parameters of the best-fit discrete power-law distributions to the pattern support value sets in Table 3.3 with $1,000$ repetitions, except the Accidents dataset with 100 repetitions and the Pumsb dataset with 500 repetitions due to the very expensive computational costs

mated and listed in Table 3.4. However, due to the very expensive computational costs, the number of repetitions for the Accidents dataset is reduced to 100 and the uncertainty computed over the 100 repetitions costs $40,318.35$ minutes on a computational server with two Quad-Core Intel 64-bit Xeons (2.83 GHz) processors and 32GB of RAM. For the same reason, the number of repetitions for the Pumsb dataset is reduced to 500 and the uncertainty computed over the 500 repetitions totally costs $44,119.06$ minutes on the same computational server.

The computed results clearly show that each stdev($\hat{x}_{\min}$) is greater than its corresponding stdev($\hat{\alpha}$), which means that the value of $\hat{\alpha}$ estimated in each repetition is relatively more stable and reliable than that estimated for $\hat{x}_{\min}$. Correspondingly, the number of pattern support values covered by each best-fit discrete power-law distribution in each repetition is the most unstable and unreliable one, which is very sensitive to the variation of $\hat{x}_{\min}$, since there are a large number of support values with each value in a pattern support distribution.

Comparing the computed results in Table 3.4 with the corresponding ones in Table 3.5 indicates that the difference between $\hat{\alpha}$ listed in Table 3.4 and its corresponding mean($\hat{\alpha}$) listed in Table 3.5 is smaller than the one between $\hat{x}_{\min}$ listed in Table 3.4 and its corresponding mean($\hat{x}_{\min}$). Also, the comparison clearly shows that the estimated parameter values listed in Table 3.4 can only be the best-fit parameter values of the best-fit discrete power-law distribution to each pattern support value set in Table 3.3 and may not be the true parameter values of the discrete power-law distribution in the process of generating each corresponding pattern support value set if such discrete power-law distributions truly exist in the generating processes. It means that the verification of a power-law distribution being the best-fit distribution to a pattern support distribution of a transaction dataset is not sufficient to verify

that a power-law distribution exists in the process of generating the transaction dataset. This issue will be further discussed and solved in the next section.

$\hat{\sigma}_1$, $\hat{\sigma}_2$ in Table 3.4 and stdev($\hat{\alpha}$) in Table 3.5 represent three methods to compute the standard deviation of the estimate $\hat{\alpha}$ on the scaling exponent parameter α in this section. $\hat{\sigma}_1$ and $\hat{\sigma}_2$ are given by Equations 3.24 and 3.25 respectively by assuming $\hat{x}_{\min} = x_{\min}$, while stdev($\hat{\alpha}$) is given via a large number of experiments.

Here, the methods represented by $\hat{\sigma}_1$ and $\hat{\sigma}_2$ are actually less reliable than the one represented by stdev($\hat{\alpha}$). The reason is that Equations 3.24 and 3.25 require $x_{\min}$ to give an accurate theoretical value of the standard deviation of the estimate of α calculated by using the MLE. However, $x_{\min}$ is unknown in the present case. Assuming that $x_{\min} = \hat{x}_{\min}$ can bring significant bias to the results calculated via Equations 3.24 and 3.25, especially when there is a significant difference between $x_{\min}$ and $\hat{x}_{\min}$. That could be the situation where the present case stands based on the calculated values of $\hat{x}_{\min}$ listed in Table 3.4, mean($\hat{x}_{\min}$) and stdev($\hat{x}_{\min}$) listed in Table 3.5. The method represented by stdev($\hat{\alpha}$) does not have this kind of pre-condition. When the number of repetitions in the nonparametric bootstrap method tends to positive infinity, the value of stdev($\hat{\alpha}$) to a target pattern support distribution tends to the true standard deviation of the estimates on α of the best-fit discrete power-law distribution by using the MLE.

3.3.3 Verification of Power-law Relationships in Pattern Support Distributions

This section aims to examine the existence of power-law relationships in the pattern support distributions of real transaction datasets. Based on the best-fit discrete power-law distributions identified in the previous section, the existence of a power-law relationship in the pattern support distributions of the five real transaction datasets listed in Table 3.1 is first examined by using D-statistic and likelihood ratio tests introduced in the verification part of Section 3.2.

Finding a phenomenon existing in some instances does not automatically prove that the phenomenon exists in each instance of the corresponding population. In the literature, lots of work shows the existence of power-law relationships in some instances in a target field, but claims the existence in the whole population of the field. The reason is that the power law itself is an empirical law, which generally means that there is not enough knowledge to theoretically truly and fully understand the

underlying process showing the power-law relationship. It is also impossible to examine the existence of power-law relationships in all instances of a target population, since normally this is infinite.

Therefore, by extending the approach suggested by Clauset et al. [30], this section also proposes a new method of utilizing the bootstrap method and the universality of power-law relationships to quantitatively verify the existence of power-law relationships in the pattern support distributions of the population of real transaction datasets.

Verification of power-law relationships in some instances

To examine the existence of the power-law relationship in the pattern support distributions of the five real transaction datasets, the verifying procedure based on the overview and discussion in the verification part of Section 3.2 has 3 steps:

1. The empirical value of the p-value is computed by using the Monte Carlo method introduced in the verification part of Section 3.2 to quantitatively measure the probability that the mined pattern support values comprising a tested pattern support distribution is drawn from its best-fit discrete power-law distribution listed in Table 3.4. The number of synthetic data sets used in the Monte Carlo method is $1,000$.

2. If the empirical value computed by using the Monte Carlo method is not greater than 0.1 (i.e., the conservative choice), it can reject the hypothesis that the power-law relationship exists in the tested pattern support distribution. Otherwise, the likelihood ratio test needs to be applied with the competing distributions of the power-law distribution to the tested pattern support distribution.

3. If the calculated value of the log maximum likelihood ratio is significantly different from 0 (i.e., the corresponding p-value is less than 0.1), the sign of the value can indicate which one of the two compared distributions is the better fit to the tested pattern support distribution.

The competing distributions used here, which were suggested by Clauset et al. [30], include:

1. the Poisson distribution, i.e.,

$$p(x) = \frac{1}{e^{\mu} - \sum_{k=0}^{(x_{\min}-1)} \frac{\mu^k}{k!}} \frac{\mu^x}{x!}, \tag{3.35}$$

2. the log-normal distribution, i.e.,

$$p(x) = \frac{1}{\sqrt{\frac{\pi\sigma^2}{2}}\,\mathrm{erfc}\left(\frac{\ln(x_{min})-\mu}{\sqrt{2}\sigma}\right)}\frac{1}{x}\exp\left(-\frac{(\ln(x)-\mu)^2}{2\sigma^2}\right), \tag{3.36}$$

3. the discrete exponential distribution, i.e.,

$$p(x) = (1-e^{-\lambda})e^{\lambda x_{\min}}e^{-\lambda x}, \tag{3.37}$$

4. the stretched exponential (Weibull) distribution, i.e.,

$$p(x) = \beta\lambda e^{\lambda x_{\min}^{\beta}}e^{-\lambda x^{\beta}}x^{\beta-1}, \tag{3.38}$$

5. the discrete power-law distribution with exponential cut-off. The discrete power-law distribution with exponential cut-off has a normalizing constant C that can only be obtained numerically from the target empirical data set.

$$p(x) = Cx^{-\alpha}e^{-\lambda x}\ . \tag{3.39}$$

Note that these competing distributions are normalized. Some of them are discrete distributions, i.e., the Poisson distribution, the discrete exponential distribution, and the discrete power-law distribution with exponential cut-off. The others are continuous distributions, i.e., the log-normal distribution and the stretched exponential (Weibull) distribution. Thus, these normalized discrete distributions obey $\sum_{x=x_{\min}}^{+\infty} p(x) = 1$, while these normalized continuous distributions obey $\int_{x_{\min}}^{+\infty} p(x) = 1$.

These normalized continuous distributions need to be discretized, when fitting them to the pattern support distributions, since the pattern support distributions are discrete. The log-normal continuous distribution is discretized by rounding, i.e., the probability of $X = k$ is the probability that the random continuous variable X would lie between $k-0.5$ and $k+0.5$. The stretched exponential (Weibull) distribution uses the Nakagawa-Osaki discretization [63], i.e., $p(x) = Pr(x) - Pr(x+1)$ where $p(x)$ is the probability mass function of the discretized Weibull distribution and $Pr(x)$ is the cumulative distribution function.

Table 3.6 shows the quantitative goodness-of-fit test results on the hypothesis that the power-law relationship exists in the pattern support distributions drawn

from the five real transaction datasets summarized in Table 3.1. The hypothesis is quantitatively measured by those results. The last row in Table 3.6 lists the results of the statistical support for the hypothesis that a power-law relationship exists in the pattern support distribution of each of the Retail, BMS-POS, Adult, Accidents, and Pumsb datasets. "Moderate" indicates that the power-law distribution is a good fit, but comparable competing distributions exist. "Poor" indicates that there are competing distributions showing a better fit than the best-fit power-law distribution.

		Dataset				
		Retail	BMS-POS	Adult	Accidents	Pumsb
power-law distribution	p_1	0	0	0	0	0
	D-statistic	0.0076	0.0062	0.0059	0.0193	0.0175
Poisson distribution	LR	7.2531	26.3793	207.8474	1,374.093	73.3836
	p_2	4.0723e-13	0	0	0	0
log-normal distribution	LR	1,662.045	10.5607	-94.4436	-224.017	-8.9972
	p_2	0	0	0	0	2.3161e-19
exponential distribution	LR	13.3998	37.8676	408.2333	219.7402	-16.6275
	p_2	0	0	0	0	4.4057e-62
stretched exponential (Weibull) distribution	LR	5.5188	28.9544	65.6457	-225.0558	-9.0710
	p_2	3.4138e-08	0	0	0	1.1797e-19
power-law distribution with exponential cut-off	LR	-14.9264	-0.0035	-12,801.60	-79,170.9	-110,785.7
	p_2	4.6614e-08	0.9329	0	0	0
support for power law		moderate (except that power-law with exponential cut-off is better)	moderate	poor	poor	poor

Table 3.6: Quantitative goodness-of-fit test results of power-law behaviour in the five studied pattern support distributions, where p_1 is the empirical p-value for the power-law distribution best fitted to a pattern support distribution, D-statistic is the maximum difference between the cumulative distribution of a pattern support distribution and the one of the best-fit power-law distribution to the pattern support distribution, LR is the log likelihood ratio of the best-fit power distribution to a corresponding competing distribution, p_2 is the statistically significant p-value for a log likelihood ratio test. Positive values of LR indicate that the power-law distribution is a better fit than a competing distribution when the corresponding p_2 is less than 0.1.

In general, the results in Table 3.6 do not provide strong evidence to support the hypothesis that the power-law relationship exists in the pattern support distribution of real transaction datasets in the interval $[x_{\min}, +\infty)$. The empirical p-values, p_1, which are computed via the Monte Carlo method, are less than 0.1 for all five real transaction datasets. It particularly shows that the pattern support distributions of 3 datasets (Adult, Accidents and Pumsb) do weakly follow the power-law distribution in the interval $[x_{\min}, +\infty)$, compared with the competing distributions.

However, the results also show that the pattern support distributions of 2 of the datasets (Retail and BMS-POS) are more likely to follow the power-law distribution than the competing distributions, except the power-law distribution with exponential cut-off.

In fact, the quantitative goodness-of-fit test results listed in Table 3.6 partially confirm the hypothesis set in the previous section. The hypothesis that the power-law relationship exists in the pattern support distributions of the Adult, Accidents and Pumsb datasets in the interval $[x_{\min}, +\infty)$ is ruled out, since their corresponding values of p_1 are less than 0.1 and there is at least one competing distribution having a better fit to each one of the 3 pattern support distributions than the power-law distribution.

Here, the hypothesis in the Retail and BMS-POS datasets is not considered to be ruled out, though the empirical values of p_1 are less than 0.1, which can rule out the hypothesis based on the suggestion given by Clauset et al. [30]. However, considering the circumstances, there are four reasons that could keep the hypothesis from being ruled out here.

First, the power-law distribution generally shows a much better fit to the pattern support distributions, compared with other competing distribution for the Retail and BMS-POS datasets. As mentioned previously, there are an unlimited number of competing distributions that can compete with the power-law distribution for testing the hypothesis. Obviously, it is impossible to test them all. Five of them, which are most comparable to the power-law distribution, are tested here. The power-law distribution shows a better fit than any one of the 4 basic distributions (i.e., the Poisson distribution, the log-normal distribution, the exponential distribution, and the stretched exponential distribution).

Although for the pattern support distribution of the Retail dataset, the power-law distribution with exponential cut-off gives a better fit than the power-law distribution, the hypothesis still cannot be ruled out. Since the power-law distribution with exponential cut-off is a nested distribution consisting of the families of the power-law and exponential distributions, in most cases it is expected to be a better fit than the pure power-law or exponential distribution. As long as there is no evidence to prove that there is another basic distribution that is a better fit to the pattern support distributions of the Retail or BMS-POS datasets, the power-law distribution can still be considered as the best-fit distribution at the level of basic distributions.

Second, possible noise and statistical fluctuation should be considered here, which can affect the results in Table 3.6. Most of the patterns mined from any one of the five real datasets are in the lower support (left side) part. Only relatively few patterns are in the higher support (right side) part. This means that the distribution in the higher support part can be more easily affected by possible noise and statistical fluctuation. It can be clearly seen in Figure 3.3 that the pattern support distributions show a straighter line in the lower support part than the higher support part. Although it cannot be sure whether there is noise affecting the result, this possibility should still be taken into account.

Third, the power-law distribution has the simplest expression form, compared with five competing distributions.

Fourth, based on the values of $\hat{x}_{\min}$ listed in Table 3.4, the best-fit power-law distribution to the pattern support distribution of the Retail/BMS-POS dataset covers the most of the possible support range in each dataset.

Based on the four reasons above, the hypothesis that power-law relationships exist in the pattern support distributions of the Retail and BMS-POS datasets cannot be ruled out.

Note that here, considering the very expensive computational cost (sometimes it is even impossible to compute), the whole pattern support distributions in the five real transaction datasets are not fully examined. Only the high support part (right side) of the pattern support distributions, which is in the interval $[x_{\min}, +\infty)$, are fully examined by using the statistical techniques. To more accurately show the expected results on the hypothesis for the pattern support distributions with the longer or even whole support value range, one can compare the D-statistic values for a pattern support distribution with different support value ranges.

Table 3.7 gives the best-fit power-law distribution and corresponding D-statistic value for each pattern support distribution mined from the five real transaction datasets with the threshold of min_sup_1. Comparing the values of D-statistic with respect to each dataset in Tables 3.6 and 3.7, the pattern support distributions of the Retail, BMS-POS and Adult datasets seem to be less likely to follow the power-law relationship when $\hat{x}_{min}$ is getting smaller, since the D-statistic value is larger. However, the pattern support distributions in the Accidents and Pumsb datasets seem to be more likely to follow the power-law relationship when $\hat{x}_{min}$ is getting smaller, since the D-statistic value is smaller. Therefore, there is a possibility that the hypothesis may hold for the pattern support distributions with a longer support

value range in the Accidents and Pumsb datasets.

Dataset	$\hat{x}_{min}(= min\_sup_1)$	$\hat{\alpha}$	Λ	D-statistic
Retail	2	5	-3.5854e+08	0.1425
BMS-POS	50	3.6	-1.5276e+08	0.0103
Adult	2	2.26	-1.6431e+08	0.0521
Accidents	5,000	3.4099	-1.5733e+10	0.0132
Pumsb	21,000	14.7963	-1.2159e+10	0.0087

Table 3.7: The best-fit discrete power-law distributions to the pattern support distributions found from the five real transaction datasets when $\hat{x}_{\min} = min\_sup_1$

Verification of power-law relationships in the whole population

Note that any one of the Retail and BMS-POS datasets tested above is just an instance of a possible dataset generated by their corresponding underlying data generating mechanism or process. Logically, proving that the power-law relationship exists in an instance is not sufficient to prove that the power-law relationship exists in the whole population (i.e., all possible datasets generated by the same underlying data generating mechanism), since it is possible purely by chance that the power-law distribution is the best fit for an instance. Therefore, it is necessary to examine the uncertainty of the hypothesis that the power-law relationship exists in the corresponding population of the Retail or BMS-POS datasets.

One method to do that is to make use of a bootstrap method. A bootstrapped transaction dataset with the same number of transactions as the original transaction dataset is generated by drawing transactions uniformly at random with replacement from the original dataset. In this way, the bootstrapped dataset has the same or similar data generating mechanism to the original one. The hypothesis that the power-law distribution is the best fit to the pattern support distribution of the bootstrapped dataset can then be examined. By taking the examined results over a large number of repetitions of this process, the uncertainty of the hypothesis in the population of the original dataset can be derived. Note that it is impossible to determine theoretically the uncertainty of the hypothesis in the population without precisely knowing the underlying data generating mechanism behind the population.

Table 3.8 shows the mean and standard deviation of the parameters and D-statistic of the best-fit power-law distribution for each pattern support distribution over thirty datasets generated by the bootstrap method. Note that thirty bootstrapped datasets are considered to be a large enough number of repetitions here,

as the standard deviation of D-statistic for each population is very small, as shown in Table 3.8.

	Dataset	
	Retail	BMS-POS
mean($\hat{x}_{\min}$)	21.4000	369.5000
stdev($\hat{x}_{\min}$)	0.9685	74.0003
mean($\hat{\alpha}$)	2.3983	3.1887
stdev($\hat{\alpha}$)	0.0070	0.0286
mean($D-$statistic)	0.0080	0.0061
stdev($D-$statistic)	7.1952e-04	3.6571e-04
max($D-$statistic)	0.0090	0.0069
min($D-$statistic)	0.0065	0.0055

Table 3.8: The computed uncertainty on the best-fit discrete power-law distributions to the pattern support distributions in the corresponding population

Table 3.9 gives the computed results of how well the power-law distribution fits to the pattern support distribution with the largest D-statistic value over the thirty bootstrapped datasets, compared with the five competing distributions. The computed results are similar to those of the original datasets (i.e., Retail and BMS-POS) shown in Table 3.6, which indicates that the power-law distribution is also the relatively better fit for each of the bootstrapped datasets. Since the value of D-statistic indicates that how well the power-law distribution fits to the pattern support distribution and in other bootstrapped datasets, the best-fit power-law distribution has a smaller D-statistic value, the power-law distribution is a relatively better fit to the pattern support distribution over the 2×30 bootstrapped datasets (i.e., the populations of the Retail and BMS-POS datasets).

As introduced in Section 3.2, the power-law relationship has a special property called universality. If the power-law relationships are found in each target dataset, and the data generating mechanisms behind each dataset are similar, the scaling exponents of the power-law relationships found in the datasets must have similar values. Since the bootstrapped datasets generated have very similar data generating mechanisms, the scaling exponents of the power-law distributions best fitted to the pattern support distributions of the bootstrapped datasets should have similar values. The values of relevant $\hat{\alpha}$ of the power-law relationships of the Retail dataset, the BMS-POS datasets and their bootstrapped datasets, which are shown in Tables 3.4 and 3.8, clearly demonstrate this inference.

		Dataset	
		Retail	BMS-POS
power-law distribution	$\hat{\alpha}$	2.4100	3.2100
	$\hat{x}_{\min}$	22	373
	D-statistic	0.0090	0.0069
Poisson distribution	LR	6.6866	24.0922
	p_2	2.2834e-11	0
log-normal distribution	LR	1,184.187	1,023.626
	p_2	0	0
exponential distribution	LR	11.7399	34.1848
	p_2	0	0
stretched exponential (Weibull) distribution	LR	6.2994	26.5496
	p_2	2.9880e-10	0
power-law distribution with exponential cut-off	LR	-8.0819	0.0336
	p_2	5.8096e-05	1
support for power law		moderate (except that power-law with exponential cut-off is better)	moderate

Table 3.9: Quantitative goodness-of-fit test results of power-law behaviours in the pattern support distributions with the maximum D-statistic value over thirty bootstrapped datasets of the Retail and BMS-POS datasets

As described early, the Retail and BMS-POS datasets are real retail transaction datasets, which (at a general level) should have similar data generating mechanisms. It is worth checking whether the power-law relationship exists in the pattern support distribution of any real retail transaction dataset. To do it, one can make use of the universality of the power-law relationship. The reliability of the results relies on the number of different real retail transaction datasets investigated.

Given the lack of real retail transaction datasets besides the Retail and BMS-POS datasets, the experimental results related to another two real retail transaction datasets (i.e., 3C_chain and Book), which are listed in the research paper [29], are retrieved for the following investigation. The transaction datasets of 3C_chain and Book are respectively from a 3C (Consumer appliances, Computer and Communication products) chain store and a large book store in Taiwan [29].

Table 3.10 gives the parameters of the 3C_chain and Book datasets and the scaling exponents of their best-fit power-law distributions based on the linear regression method. Although the scaling exponent value of the best-fit power-law distribution to a pattern support distribution is biased if the estimate is based on linear regres-

sion, the bias does not seem to be large by visually checking the difference between the estimated power-law distributions and the fitting pattern support distributions in the figures listed in [29].

Dataset	$\|I\|$	$\|TD\|$	$\|T\|_{\max}$	$\|T\|_{\text{avg}}$	$\hat{\alpha}$
3C_chain	130,108	8,000,000	87	5.4	1.9280
Book	12,082	100,000	13	2.3	2.5173

Table 3.10: Parameters of the 3C_chain and Book datasets and the scaling exponents of their best-fit power-law distributions based on linear regression

The values of the scaling exponents of the best-fit power-law distribution for the real retail transaction datasets (i.e., the Retail, BMS-POS, 3C_chain, and Book datasets), which are shown in Tables 3.4 and 3.10, are considered to be similar, based on two reasons:

First, the maximum difference between the values of the scaling exponents is 1.2720, which is very small compared to the range of possible values.

Second, the properties of each tested dataset are different, which can affect the estimated value of the scaling exponent of the best-fit power-law distribution to the pattern support distribution of the tested dataset. For instance, in general, with other properties being the same, the higher support part of the pattern support distribution of the transaction dataset with fewer existing items has a steeper curve than the one with more items, since the probability that more patterns have higher support value is higher. With other properties being the same, the higher support part of the pattern support distribution of the transaction dataset with a larger average number of items per transaction has a steeper curve than the one with smaller average number of items per transaction.

Table 3.11 lists the scaling exponent values of the best-fit power-law distributions for the four real retail transaction datasets and two important parameters that can affect the scaling exponent value. The value of $|I|/|T|_{\text{avg}}$ affects the value of the scaling exponent. The larger the value of $|I|/|T|_{\text{avg}}$ is, the fewer patterns with higher support values there are, when the number of transactions in the target dataset is fixed. Hence, in general, the higher support part of the pattern support distribution is flatter, in the dataset with a smaller value of $|I|/|T|_{\text{avg}}$, as can be seen via the value of $\hat{\alpha}$ in Table 3.11. Furthermore, the values of the scaling exponents are closer when the datasets have the same magnitude of values of $|I|/|T|_{\text{avg}}$.

At the same time, the number of transactions in the datasets also affects the

Dataset	$\|I\|/\|T\|_{\text{avg}}$	$\|TD\|$	$\hat{\alpha}$
BMS-POS	2.5492e+02	5.1560e+05	3.2
Retail	1.599e+03	8.8162e+04	2.37
Book	5.253e+03	1e+05	2.5173
3C_chain	2.4094e+04	8e+06	1.9280

Table 3.11: The similarity of the values of the scaling exponents of the best-fit power-law distributions for the 4 real retail transaction datasets

value of the scaling exponents. For instance, when using the bootstrap method to generate thirty datasets from the BMS-POS, and the number of transactions in each bootstrapped dataset is $88,162$, which is exactly how many transactions there are in the Retail dataset, the mean and standard deviation of the scaling exponent of the pattern support distributions of the thirty bootstrapped datasets is 3.0920 and 0.0609 respectively. This can explain why the values of $\hat{\alpha}$ for the Retail and Book datasets are not in the order of the values of their $|I|/|T|_{\text{avg}}$. In addition, it can also be observed from Table 3.11 that the number of transactions in a transaction dataset seems to have less power to affect the scaling exponent than the value of $|I|/|T|_{\text{avg}}$ of the dataset.

Ideally, any comparison between objects should be implemented based on the same conditions, so the results of the targeted properties that have been compared during the comparison are more reliable. Based on the discussion above, if the BMS-POS, Retail, Book, and 3C_chain datasets have the same (explicit) properties, the values of the scaling exponents for these four datasets can be closer and the maximum difference between the values of the scaling exponents is expected to be smaller than 1.2720. Hence, the scaling exponents of the best-fit power-law distributions to the pattern support distributions of the four real retail transaction datasets are considered to be similar, and the universality property of the power-law relationships of the four real retail transaction datasets is revealed.

It can be inferred that power-law relationships exist in the pattern support distributions of real retail transaction datasets, since power-law relationships are identified in the pattern support distributions of the instances and the population of the real retail transaction datasets generated from the same and diverse real retail data sources. Based on the experimental results listed in this section, a more proper claim is that the hypothesis that the power-law relationship exists in the pattern support distributions of real retail transaction datasets is not ruled out at the level of basic

distributions, which is clearly different to the one made by Chuang et al. [28, 29].

3.3.4 Observations on the Self-similarity Phenomena in Pattern Support Distributions

This section focuses of the observations on the self-similarity phenomena in the pattern support distributions of real retail transaction datasets. The statistical verification and theoretical model of the self-similarity phenomenon linked to the pattern support distributions of real retail transaction datasets are beyond the scope of this study. Note that as the cumulative distribution of any power-law distribution is also a power-law distribution, the self-similarity phenomena mentioned below are also expected to be observed from the cumulative pattern support distributions of real retail transaction datasets.

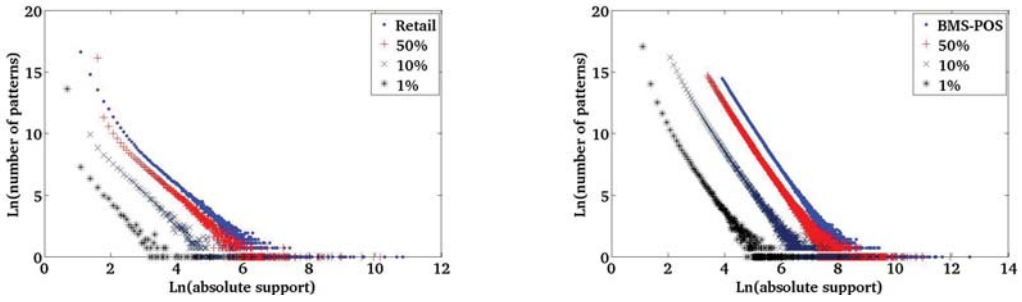
As introduced in Section 3.2, the power-law relationship has a special property called scale invariance. In reality, with the influence of noise and other external factors, self-similarity phenomena are more often found with power-law relationships than the phenomena of scale invariance, which depicts that in reality the appearance of a power-law relationship can be changed to some extent with difference scales.

An object is statistically self-similar if parts of it display the same statistical properties at many scales [78]. Note that self-similarity does not emphasize the magnification levels where the self-similarity holds. It is a loose constraint. Scale invariance is the strict kind of self-similarity, which requires that the self-similarity holds in any magnification level. It implies that the self-similarity phenomenon can be infinitely observed.

Depending upon the circumstances, a pattern support distribution can be said to be statistically self-similar if the appearance of the pattern support distribution viewed at many different scales is similar to the original one. Three groups of self-similarity phenomena are observed from the pattern support distributions of real retail transaction datasets.

First, as demonstrated above, a power-law relationship exists in the pattern support distributions of real retail transaction datasets. Consequently, parts of the pattern support distribution of a real retail transaction dataset are self-similar to the whole of the pattern support distribution in a log-log plot. As examples, the self-similarity phenomena can be seen from the pattern support distributions of the Retail and BMS-POS datasets shown in Figure 3.3.

Second, the pattern support distribution of a sample set randomly drawn from a real retail transaction dataset is self-similar to the one in the original dataset. In this case, the degree of the self-similarity is affected by the ratio of the number of transactions in the sample over the one in the original dataset. It is illustrated with Figure 3.4, where $i\%$ represents the pattern support distribution of the sample, in which the number of transactions is $i\%$ of the one in the original dataset. Three values (1, 10 and 50) are arbitrarily selected for i as examples shown in each subfigure of Figure 3.4. The labels of "Retail" and "BMS-POS" respectively represent the original pattern support distribution in the Retail and BMS-POS datasets.

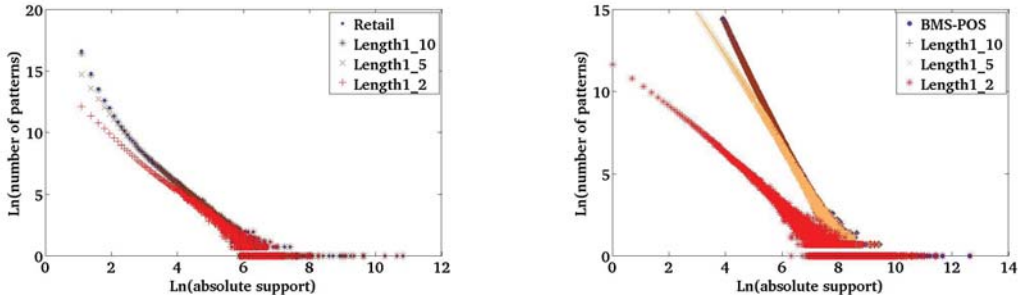


(a) Retail (b) BMS-POS

Figure 3.4: The self-similarity phenomena in the pattern support distributions between the Retail/BMS-POS dataset and its samples

Third, the self-similarity phenomena can also be observed between the pattern support distributions consisting of the existing patterns with all possible lengths and the one only consisting of the existing patterns whose length is not greater than a certain length. The examples are given in Figure 3.5, where "Length1_i" represents the pattern support distribution only consisting of the patterns whose length is between 1 and i. Three values (2, 5 and 10) are arbitrarily selected for i to illustrate the self-similarity phenomena in Figure 3.5.

Figure 3.5 clearly shows that, when the value of i is closer to the maximum length of patterns in the transaction datasets, the pattern support distribution represented by Length1_i is closer to the original one consisting of all patterns with diverse length. The degree of the self-similarity is reduced as the value of i gets smaller. The reason is that the longer the length of a pattern is, the smaller the probability that the pattern has a high support value in the dataset is. Therefore, patterns with longer length are more likely to be in the lower support part (left side) of the original pattern support distribution, in which a huge number of patterns exist. If the number of patterns with longer length is not large, the shape of the pattern



(a) Retail (b) BMS-POS

Figure 3.5: The self-similarity phenomena in the pattern support distributions of the Retail/BMS-POS dataset with different scales on pattern length

support distribution is not affected too much by the factor of whether or not these patterns are included in the pattern support distribution.

Note that the similarity phenomenon is not equal to the self-similarity phenomenon, which can be quite confused in some situations. The similarity phenomena can be observed between the pattern support distributions of real retail transaction datasets from different real retail data sources. Moreover, the similarity phenomena exists between the pattern support distributions only consisting of the patterns with a certain different length in a dataset, which is claimed as self-similarity by Chuang et al. [28, 29]. For example, in the Retail dataset, the pattern support distribution only consisting of the patterns whose length is equal to 10, represented by "Length10" in Figure 3.6, is similar to the one only consisting of the patterns whose length is equal to 11, represented by "Length11". These two similarity phenomena are actually not the self-similarity phenomena. For self-similarity, the similarity has to exist between the targeted object and its parts. The two similarity phenomena do not satisfy this requirement.

3.4 Summary

Pattern support distributions are an important and implicit characteristic of datasets. They are useful to lots of data mining applications, such as determining the appropriate minimum support for frequent pattern mining. The power-law relationship and self-similarity are very interesting natural phenomena that have been claimed to exist in pattern support distributions of real transaction datasets by Chuang et al. [28, 29]. Inspired by this claim, this chapter has investigated power-law relationships in the pattern support distributions of real transaction datasets.

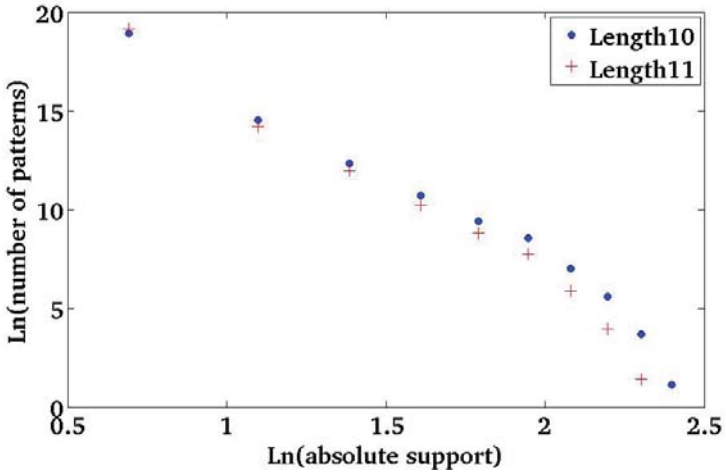


Figure 3.6: The similarity phenomena between the pattern support distributions with a certain different length in the Retail dataset

Besides using qualitative appraisal based on visualizations of the partial pattern support distributions of five real transaction datasets, we also utilize statistical goodness-of-fit tests to quantitatively examine the existence of power-law relationships in the pattern support distributions of real transaction datasets.

Based on the approach suggested by Clauset et al. [30], we propose a new method of utilizing the bootstrap method and the universality of power-law relationships to extend the quantitative goodness-of-fit verification for a single target dataset to all datasets from the same or similar real data sources such as retail transaction data.

More accurately, this chapter aims to give a new method to test the hypothesis that power-law relationships exist in the pattern support distributions of real retail transaction datasets, instead of proving the hypothesis. The reason is that the new method based on the bootstrap method and the universality of power-law relationships cannot be tested with every existing real retail transaction dataset.

Based on a large number of experimental results listed in this chapter, strictly speaking, only the hypothesis that the power-law relationship exists in pattern support distributions of real retail transaction datasets cannot be rejected at the level of basic distributions. Note that our finding is different from the claim made by Chuang et al. [28, 29]. Moreover, three groups of self-similarity phenomena linked to power-law relationships in pattern support distributions of real retail transaction datasets have been revealed.

There are some other forms of power-law relationships such as $f(x) \propto L(x)x^{-\alpha}$, where $L(x)$ is a slowly varying function. When one applies other forms of power-law

relationships in the study regarding the power-law-based pattern support distributions of real transaction datasets, the ideas and methods introduced and proposed in this chapter can still be very useful.

Chapter 4

Efficiently Estimating Power-law-based Pattern Support Distributions

Your true value depends entirely on what you are compared with.

— Bob Wells

This chapter studies how to efficiently estimate full power-law-based pattern support distributions of large transaction datasets. Accurately characterizing the power-law relationship in the full pattern support distribution of a given dataset based on MLE and the KS-test, which was introduced in Chapter 3, is often not practical. In order to perform accurate characterization, the complete set of distinct patterns and their supports in the dataset is required. Generally, the number of distinct patterns in a transaction dataset is exponential in the number of items and polynomial in the number of transactions, as demonstrated in the experiments in Chapter 3. Counting all distinct patterns in a given transaction dataset is virtually impossible in most cases. In this chapter, we will propose cost-effective approaches to solving this problem.

This chapter is organized as follows. The background of the problem is formally introduced in Section 4.1. After that, the influence of sampling on the estimation of the full power-law-based pattern support distributions of transaction datasets is investigated in Section 4.2, since it seems reasonable that using sampled datasets instead of the complete dataset may reduce the computational cost without significant detriment to the estimation accuracy. Then, in Section 4.3 novel approaches are proposed to effectively and efficiently estimate the full power-law-based pattern

support distribution in a given transaction dataset by using two statistics: the mean support of patterns and the mean number of patterns per transaction. Theoretical discussion and empirical studies show that the proposed approaches can relatively efficiently estimate the characteristics of the full power-law-based pattern support distribution of static transaction datasets with relatively high estimation accuracy. Finally, this chapter is summarized in Section 4.4.

4.1 Introduction

As was demonstrated in Chapter 3, the hypothesis that power-law relationships exist in the pattern support distributions of real retail transaction datasets cannot be ruled out. For the sake of analytical simplicity, in this chapter, a power-law relationship is assumed to exist in the full pattern support distribution of a transaction dataset. Thus, the pattern support distribution of a transaction dataset TD can be represented by a straight line in a natural log-log plot, i.e.,

$$\ln(PSD(x, TD)) = -\alpha \ln(x) + b \,, \tag{4.1}$$

where $PSD(x, TD)$ denotes the number of patterns whose support values are equal to x in TD. $\alpha(> 0)$ and $b(> 0)$ respectively denote the scaling exponent and the Y-intercept of the power-law-based pattern support distribution in a natural log-log plot, which can be computed by using Maximum Likelihood Estimation as introduced in Chapter 3.

The estimation method suggested in Section 3.2 is necessary to examine whether a pattern support distribution actually follows a particular distribution, since the verification requires the extremely accurate estimation on the pattern support distribution.

In general, for large datasets, to precisely estimate the pattern support distribution over an entire dataset is computationally expensive, since it requires counting the number of times each of the $(2^{|I|}-1)$ possible patterns occurs in the target transaction dataset, where $|I|$ denotes the number of items in the dataset. A naïve implementation of this would therefore have the computational cost $O\big(\big(2^{|I|}-1\big)|TD||T|_{\text{avg}}\big)$, where $|TD|$ denotes the number of transactions and $|T|_{\text{avg}}$ denotes the average length of transactions in the target dataset. While more advanced algorithms are possible using data structures such as trees, the problem of computational cost does not go

away: it is still an NP-hard problem [124].

Without considering the execution time, the estimation method suggested in Section 3.2 is perfect for estimating a power-law-based pattern support distribution. However, in practice, the execution time often needs to be considered, especially when high accuracy of the estimated pattern support distribution is not requested. Therefore, methods that can efficiently estimate the pattern support distribution without significant detriment to the estimation accuracy are needed.

4.2 The Influence of Sampling on Pattern Support Distributions

This section investigates the influence of sampling on the power-law-based pattern support distribution of a large transaction dataset. More specifically, this section focuses on the relationship between the shape deviation of the power-law-based pattern support distribution of a transaction dataset and a sample size when using stratified sampling with replacement.

Sampling has been successfully used in many applications where large datasets make computations expensive and approximate results are acceptable. Therefore, an obvious method to alleviate the computational cost of estimating the power-law-based pattern support distribution of a large transaction dataset is to use sampling.

Sampling here refers to drawing a relatively small number of transactions as a sample dataset from the original large dataset so that the power-law-based pattern support distribution of the original large dataset can then be inexpensively estimated based on support values in the sample dataset(s). This can drastically reduce the number of transactions to be considered, and avoid the need for retrieving and handling lots of patterns. However, the computed results are not as accurate as those directly based on the original entire datasets [29, 135]. Patterns may be miscounted, which may also cause changes in the estimated distribution. Consequently, the pattern support distribution estimated based on samples can differ from the one based on the original entire dataset.

The distribution of support values of a pattern in a sample typically deviates from the underlying distribution in the original dataset. This is sometimes known as the 'support deviation phenomenon'. It can be minimized by determining an appropriate sample size in order to control the bias of the estimated support for the patterns in the dataset. For example, progressive sampling [91] aims to keep

on increasing the sample size and testing the corresponding model accuracy until the model accuracy stops increasing notably. Statistical approaches include using the Central Limit Theorem [135] and Chernoff bound [134]. Another approach [29] used a relatively large sample size and quantized the support distribution in order to reduce the support deviation.

For pattern support distributions, another phenomenon caused by sampling is the 'shape deviation phenomenon', which refers to the fact that the shape of the pattern support distribution of a large dataset differs from that of sample datasets drawn from it. Unlike the support deviation phenomenon, the shape deviation phenomenon has not been deeply investigated. We will theoretically and experimentally investigate this phenomenon in this section.

There are many different basic types of sampling methods described in the statistical literature (see [72] for a review). It is impossible to discuss the influence of every sampling method on a power-law-based pattern support distribution. Stratified sampling with replacement, which is one of the basic sampling methods, is used in this section. The average number of items per transaction in a transaction dataset has a big impact on the pattern support distribution of the dataset, as was indicated in Section 3.3.4.

Stratified sampling with replacement refers to sampling according to the length of a transaction. All transactions in a given transaction dataset are categorized into different length groups and then a number of transactions are randomly selected from each length group, based on the size of each group and the sampling ratio used, in order to set up a sample dataset for estimating the pattern support distribution of the given dataset. In this way, the number of patterns with different lengths matches the proportions in the given dataset. Thus, the difference between the average number of items per transaction of the given dataset and its sample dataset does not affect the estimation of the power-law-based pattern support distribution of the given dataset too much.

The following discussion about the influence of sampling on the power-law-based pattern support distribution of a real retail transaction dataset is under two assumptions: 1) uniform distribution of pattern supports, and 2) independently and identically distributed (i.i.d.) distributions of pattern supports. The difference between the pattern support distributions and their corresponding samples is computed and analyzed in order to hopefully lead to methods of identifying the pattern support distribution of a dataset based on those found in its sample dataset(s).

Firstly, the influence of sampling on the pattern support distribution is analyzed based on the assumption that the support of each pattern in the target transaction dataset is uniformly distributed. In other words, each transaction in the target dataset has a uniform probability of containing each item in the dataset. Even though this assumption is apparently far from reality when the value of possible support in a real transaction dataset is discrete, the discussion can still highlight some interesting phenomena and provide preliminary understanding of the pattern support distributions of real transaction datasets where each item is treated independently.

Suppose that a sample dataset SD is randomly drawn from a large transaction dataset TD. Treating the pattern support distribution as a continuous distribution enables the computation of absolute support for a pattern z in SD as:

$$\frac{sup(z, SD)}{|SD|} \approx \frac{sup(z, TD)}{|TD|} , \tag{4.2}$$

where $sup(z, SD)$ is the absolute support of a pattern z in transaction dataset SD.

Under the assumption of uniformly distributed pattern supports, the distribution of support values in SD will approximate those in TD, with the match getting stronger as the number of transactions in SD increases. Therefore, the following approximation can be made:

$$PSD(sup(z, TD), TD) \approx PSD(sup(z, SD), SD) . \tag{4.3}$$

Accordingly, the pattern support distribution of SD can be described as:

$$\ln(PSD(x, SD)) \approx -\alpha \left(\ln(x) + \ln\left(\frac{|TD|}{|SD|} \right) \right) + b . \tag{4.4}$$

Figure 4.1 shows the effect of this based on samples drawn by using the sampling ratios (i.e., $\frac{|SD|}{|TD|}$) of 30% and 5% respectively from an artificially created transaction dataset where the support of each pattern is independently and uniformly distributed. It clearly demonstrates that the influence of sampling on the pattern support distribution can be precisely predicted by using Equation 4.4, namely an affine shift along the X-axis by $\ln\left(\frac{|TD|}{|SD|}\right)$.

However, in reality the pattern support distribution of a real transaction dataset is discrete and the absolute support of a pattern in a real transaction dataset has to be a positive integer. As a result, each point of the pattern support distribution

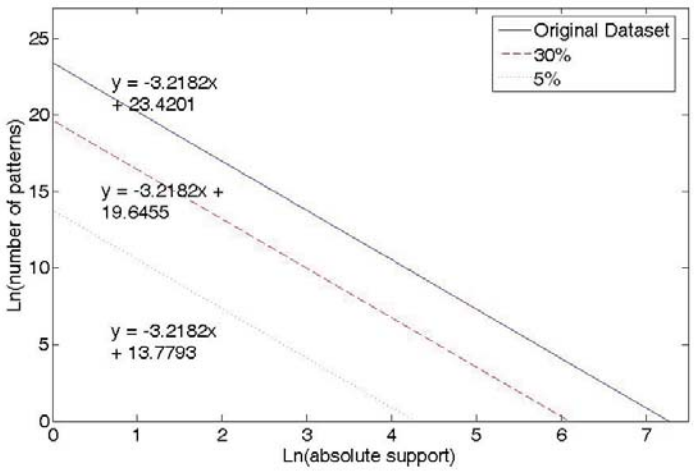


Figure 4.1: The influence of sampling on the PSD in a dataset where the support of each pattern is continually uniformly distributed

of SD is aggregated by (on average) $\frac{|TD|}{|SD|}$ corresponding sequential points in the pattern support distribution of TD. Therefore, the number of the patterns whose absolute support is equal to a positive integer x' in SD approximately satisfies:

$$PSD(x', SD) \approx \sum_{i=1}^{\frac{|TD|}{|SD|}} PSD(x_i, TD) = e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} (x_i)^{-\alpha} \,, \tag{4.5}$$

where x_i is the support value of a pattern in TD, whose support value is close to x', i.e.,

$$x' - 0.5 \leqslant \frac{|SD|}{|TD|} x_i < x' + 0.5 \,. \tag{4.6}$$

By using $\frac{x'}{|SD|} \approx \frac{x_i}{|TD|}$ and taking the natural logarithm on the leftmost and rightmost sides of Expression 4.5, it becomes:

$$\begin{aligned} \ln(PSD(x', SD)) &\approx \ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} (x_i)^{-\alpha} \right) \\ &\approx \ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} \left(\frac{|TD|}{|SD|} x' \right)^{-\alpha} \right) \\ &= -\alpha \ln(x') + (-\alpha + 1) \ln\left(\frac{|TD|}{|SD|} \right) + b \,. \end{aligned} \tag{4.7}$$

The bounds on the second approximation caused by using $\frac{x'}{|SD|} \approx \frac{x_i}{|TD|}$ can be

calculated straightforwardly. Since $x' - 0.5 \leqslant \frac{|SD|}{|TD|}x_i < x' + 0.5$,

$$\ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} (x_i)^{-\alpha}\right) - \ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} \left(\frac{|TD|}{|SD|}x'\right)^{-\alpha}\right) \leqslant -\alpha \ln\left(1 - \frac{1}{2x'}\right) \tag{4.8}$$

and

$$\ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} (x_i)^{-\alpha}\right) - \ln\left(e^b \sum_{i=1}^{\frac{|TD|}{|SD|}} \left(\frac{|TD|}{|SD|}x'\right)^{-\alpha}\right) > -\alpha \ln\left(1 + \frac{1}{2x'}\right). \tag{4.9}$$

There are three sample datasets with randomly selected sampling ratios (2%, 0.5875% and 0.2%) respectively shown in Figure 4.2. For the sake of simplicity, the best-fit power-law distributions to the pattern support distributions of the original and sample datasets, which are represented by "RL" in Figure 4.2, are computed by using linear regression, which does not affect the following observation and discussion related to Figure 4.2. As can be seen in the figure, when the pattern support distribution is treated as a discrete distribution, the influence of sampling approximately produces an affine transformation in log-log space that shifts the pattern support distribution $PSD(TD)$ by $\ln\left(\frac{|TD|}{|SD|}\right)$ along the X-axis, then shifts it by $\ln\left(\frac{|TD|}{|SD|}\right)$ along the Y-axis. As shown by Equations 4.8 and 4.9, the real behaviour of the influence of sampling is slightly different to that shown by Expression 4.7. With a certain sampling ratio, the closer the value of x' is to 1, the steeper the slope at Point $(x', PSD(x', SD))$ of $PSD(SD)$ is.

In reality, the support of each pattern is rarely independently and uniformly distributed in a real transaction dataset. While there is very likely to be some correlations between patterns within a dataset, and therefore possible correlations within support distributions, these are not usually known to the data miner. Indeed, this may be one of the features that they want to discover. Hence, the case that each pattern support is treated as if it follows an independent, unknown distribution in a transaction dataset is considered in the following discussion.

Suppose that a pattern z in TD is randomly distributed with an unknown distribution. For a given transaction T, the pattern z is either a subset of T, or it is not. When randomly sampling with replacement, the probability that a pattern z is included in any given transaction in TD is equal to its relative support (i.e., frequency) in TD, denoted as $freq(z, TD)$, and the probability that z is not included

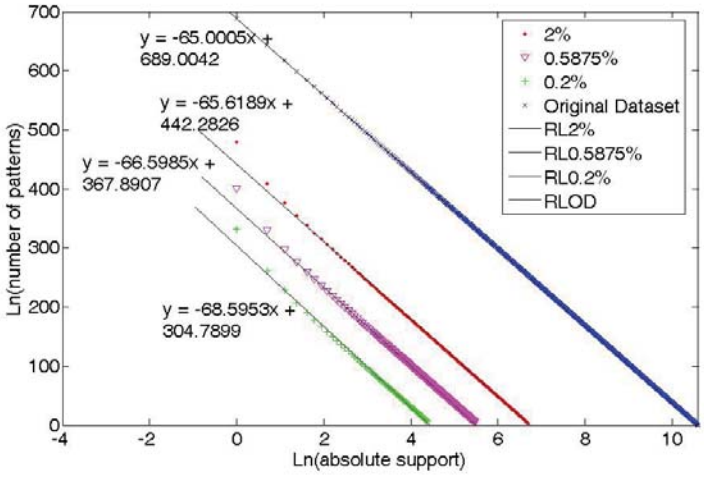


Figure 4.2: The influence of sampling on the PSD in a dataset where the support of each pattern is discretely uniformly distributed

is $1 - freq(z, TD)$.

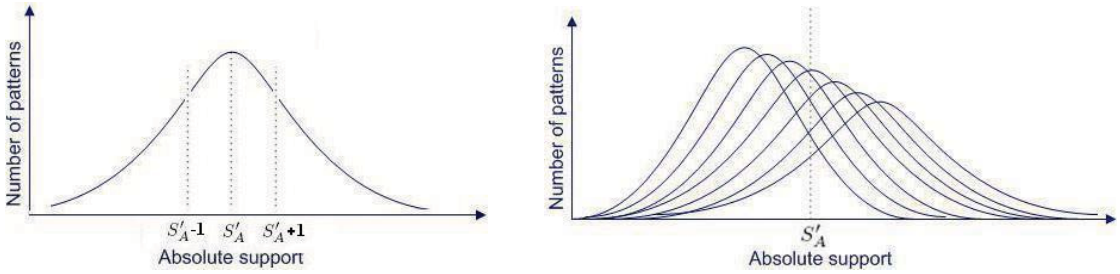
The absolute support of a pattern z in a random sample SD will follow a binomial distribution with the probability mass function:

$$f(sup(z, SD); |SD|, freq(z, TD)) = \\ \mathrm{C}^{|SD|}_{sup(z,SD)} \, (freq(z, TD))^{sup(z,SD)} \, (1 - freq(z, TD))^{|SD| - sup(z,SD)}, \quad (4.10)$$

where $\mathrm{C}^{|SD|}_{sup(z,SD)}$ is the usual combinatoric function ($|SD|$ choose $sup(z, SD)$), and the absolute support $sup(z, SD)$ runs over all possible absolute support values from 1 to $|SD|$.

Figure 4.3 shows the influence of randomly sampling: the absolute support value $sup(z, SD)$ of a pattern z in SD may not be equal to the expected value $freq(z, TD)$ $|SD|$. In particular, the binomial distribution is a discrete distribution. As the matter of fact, these issues can significantly affect the values of the parameters of the power-law-based pattern support distribution in a sample dataset.

Given a sample dataset SD randomly drawn with replacement from TD, the patterns with support x in TD will have the same probability that their support value in SD is equal to a certain support value x'. Let $(x, PSD(x, TD))$ be a point in the pattern support distribution of TD, where x represents an absolute support value and $PSD(x, TD)$ is the number of the patterns whose support values are equal to x in TD. As the number of the patterns with the same support value increases, the distribution will tend to be binomial. In this case, the number of the patterns,



(a) The distribution of the support of a single pattern drawn from an original transaction dataset TD

(b) The distributions of the supports of multiple patterns with different support values in TD when drawing them from TD into a sample dataset SD

Figure 4.3: The influence of sampling on the number of the patterns with a certain absolute support value in a sample dataset

denoted by n, whose absolute support value is respectively equal to x in TD and x' in SD, satisfies:

$$\begin{aligned} n &\approx PSD(x,TD)\ f\left(x';|SD|,\frac{x}{|TD|}\right) \\ &= PSD(x,TD)\ \mathrm{C}_{x'}^{|SD|}\left(\frac{x}{|TD|}\right)^{x'}\left(1-\frac{x}{|TD|}\right)^{|SD|-x'} . \end{aligned} \tag{4.11}$$

Let $(x', PSD(x', SD))$ be a point in the pattern support distribution of a sample set SD drawn from a transaction dataset TD. x' represents an absolute support value in SD. When the absolute supports of the patterns in SD ideally follow their corresponding binomial distributions, or $PSD(x_i, TD)$ tends to be a very large positive number,

$$\begin{aligned} &PSD(x', SD) \\ &= \sum_{i=1}^{k} n_i \\ &\approx \sum_{i=1}^{k}\left(PSD(x_i,TD)\ f\left(x';|SD|,\frac{x_i}{|TD|}\right)\right) \\ &= \sum_{i=1}^{k}\left(PSD(x_i,TD)\ \mathrm{C}_{x'}^{|SD|}\left(\frac{x_i}{|TD|}\right)^{x'}\left(1-\frac{x_i}{|TD|}\right)^{|SD|-x'}\right) , \end{aligned} \tag{4.12}$$

where x_i is a particular absolute support value in the original transaction dataset TD, and k is the maximum support value in $PSD\,(x, TD)$.

Since $PSD\,(x, TD)$ is a positive integer and $\alpha > 0$, based on Equation 4.1, one can obtain:

$$\ln(1) = -\alpha \ln(k) + b\,, \tag{4.13}$$

where k is the maximum support value covered by $PSD\,(x, TD)$.

As k is a positive integer, by rounding off, one can get

$$k \approx \left[e^{\frac{b}{\alpha}}\right]\,. \tag{4.14}$$

Since only the patterns whose absolute support is not less than x' in TD have the chance that their absolute support value is equal to x' in SD, and here $x_i = i$,

$$\begin{aligned} &PSD(x', SD) \\ &\quad \approx \sum_{i=x'}^{k} \left(PSD(i, TD)\ \mathrm{C}_{x'}^{|SD|} \left(\frac{i}{|TD|}\right)^{x'} \left(1 - \frac{i}{|TD|}\right)^{|SD|-x'} \right). \end{aligned} \tag{4.15}$$

Based on Equation 4.1,

$$\begin{aligned} PSD(x, TD) &= e^{\ln(PSD(x,TD))} \\ &= e^{-\alpha \ln(x)+b} \\ &= x^{-\alpha} e^{b}\,. \end{aligned} \tag{4.16}$$

Taking the natural logarithm on both sides of Equation 4.15,

$$\begin{aligned} &\ln(PSD(x', SD)) \\ &\approx \ln\left(\sum_{i=x'}^{k} \left(i^{-\alpha} e^{b}\ \mathrm{C}_{x'}^{|SD|} \left(\frac{i}{|TD|}\right)^{x'} \left(1 - \frac{i}{|TD|}\right)^{|SD|-x'} \right) \right) \\ &= \ln\left(|TD|^{-\alpha} e^{b}\ \mathrm{C}_{x'}^{|SD|} \sum_{i=x'}^{k} \left(\left(\frac{i}{|TD|}\right)^{x'-\alpha} \left(1 - \frac{i}{|TD|}\right)^{|SD|-x'} \right) \right) \\ &= \ln\left(\sum_{i=x'}^{k} \left(\left(\frac{i}{|TD|}\right)^{x'-\alpha} \left(1 - \frac{i}{|TD|}\right)^{|SD|-x'} \right) \right) \\ &\quad + \ln\left(\mathrm{C}_{x'}^{|SD|}\right) - \alpha \ln(|TD|) + b \\ &= \ln\left(\sum_{i=x'}^{k} \left(\left(\frac{i}{|TD|}\right)^{x'-\alpha} \left(1 - \frac{i}{|TD|}\right)^{|SD|-x'} \right) \right) \\ &\quad + \ln(|SD|!) - \ln((|SD| - x')!) - \ln(x'!) - \alpha \ln(|TD|) + b\,. \end{aligned} \tag{4.17}$$

In general the number of transactions in TD, denoted by $|TD|$, is very large. Based on the Riemann integral [100],

$$\begin{aligned}
&\lim_{1/|TD|\to 0}\left(\sum_{i=x'}^{k}\left(\left(\frac{i}{|TD|}\right)^{x'-\alpha}\left(1-\frac{i}{|TD|}\right)^{|SD|-x'}\right)\right)\\
&\rightarrow \int_{x'/|TD|}^{k/|TD|}\left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right)\,dt\\
&= \frac{\left(t^{x'-\alpha+1}\,{}_2F_1\left(x'-|SD|,x'-\alpha+1;x'-\alpha+2;t\right)\right)\Big|_{\frac{x'}{|TD|}}^{\frac{k}{|TD|}}}{x'-\alpha+1},
\end{aligned} \tag{4.18}$$

where ${}_2F_1$ denotes a Gaussian or ordinary *hypergeometric function* [12].

$${}_2F_1\left(a,b;c;x\right)=\frac{\Gamma\left(c\right)}{\Gamma\left(b\right)\Gamma\left(c-b\right)}\int_0^1 v^{b-1}(1-v)^{c-b-1}(1-vx)^{-a}dv\ , \tag{4.19}$$

where Γ represents a gamma function [12]. If a or b is 0 or a negative integer, ${}_2F_1\left(a,b;c;x\right)$ reduces to a polynomial, but if c is 0 or a negative integer, ${}_2F_1\left(a,b;c;x\right)$ is not defined [110].

This leads to an intractable problem. Lots of researchers such as John Wallis [33], Leonhard Euler [98], Carl Friedrich Gauss [16], Ernst Eduard Kummer [64], Srinivasa Ramanujan [6], and B. C. Berndt [31] have made great contributions to hypergeometric functions. Nevertheless, to the best of our knowledge, so far there is still no accurately simply theoretical solution to a Gaussian hypergeometric function with general arguments. An approximate solution can be calculated by using numerical analysis for each individual hypergeometric function. However, this is not practical in this case, since the involved calculation can be very expensive. Moreover, the numerical result cannot accurately express the relationship between the sample size chosen and the shape deviation on the pattern support distribution. Therefore, advanced mathematical development is required in the future in order to accurately and explicitly express the relationship between the sample size and the shape deviation on the pattern support distribution.

Approximate bounds such as the following ones on $\ln(PSD(x',SD))$ can be calculated, although currently there is no simple form to express $PSD(x',SD)$ or $\ln(PSD(x',SD))$ due to the involved hypergeometric functions.

Note that when $t=\frac{x'-\alpha}{|SD|-\alpha}$, $f=t^{x'-\alpha}(1-t)^{|SD|-x'}$ reaches its maximum value.

Hence, when $\frac{k}{|TD|} \leqslant \frac{x'-\alpha}{|SD|-\alpha}$,

$$\begin{aligned}
&\int_{x'/|TD|}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt \\
&\geqslant \int_{x'/|TD|}^{k/|TD|} t^{|SD|-\alpha} dt \\
&= \frac{1}{|SD|-\alpha+1}\left(\left(\frac{k}{|TD|}\right)^{|SD|-\alpha+1} - \left(\frac{x'}{|TD|}\right)^{|SD|-\alpha+1}\right) \qquad (4.20)
\end{aligned}$$

and

$$\begin{aligned}
&\int_{x'/|TD|}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt \\
&\leqslant \int_{x'/|TD|}^{k/|TD|} (1-t)^{|SD|-\alpha} dt \\
&= \frac{1}{|SD|-\alpha+1}\left(\left(1-\frac{x'}{|TD|}\right)^{|SD|-\alpha+1} - \left(1-\frac{k}{|TD|}\right)^{|SD|-\alpha+1}\right). \qquad (4.21)
\end{aligned}$$

When $\frac{k}{|TD|} > \frac{x'-\alpha}{|SD|-\alpha}$,

$$\begin{aligned}
&\int_{x'/|TD|}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt \\
&= \int_{x'/|TD|}^{(x'-\alpha)/(|SD|-\alpha)} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt \\
&\quad + \int_{(x'-\alpha)/(|SD|-\alpha)}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt \\
&\geqslant \int_{x'/|TD|}^{(x'-\alpha)/(|SD|-\alpha)} t^{|SD|-\alpha} dt + \int_{(x'-\alpha)/(|SD|-\alpha)}^{k/|TD|} (1-t)^{|SD|-\alpha} dt \\
&= \frac{1}{|SD|-\alpha+1}\left(\left(\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1} - \left(\frac{x'}{|TD|}\right)^{|SD|-\alpha+1}\right. \\
&\quad \left. + \left(1-\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1} - \left(1-\frac{k}{|TD|}\right)^{|SD|-\alpha+1}\right) \qquad (4.22)
\end{aligned}$$

and

$$\int_{x'/|TD|}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right) dt$$

$$
\begin{aligned}
&= \int_{x'/|TD|}^{(x'-\alpha)/(|SD|-\alpha)} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right)dt \\
&\quad + \int_{(x'-\alpha)/|SD|-\alpha}^{k/|TD|} \left(t^{x'-\alpha}(1-t)^{|SD|-x'}\right)dt \\
&\leqslant \int_{x'/|TD|}^{(x'-\alpha)/(|SD|-\alpha)} (1-t)^{|SD|-\alpha}dt + \int_{(x'-\alpha)/(|SD|-\alpha)}^{k/|TD|} t^{|SD|-\alpha}dt \\
&= \frac{1}{|SD|-\alpha+1}\left(\left(1-\frac{x'}{|TD|}\right)^{|SD|-\alpha+1} - \left(1-\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1}\right. \\
&\quad \left. + \left(\frac{k}{|TD|}\right)^{|SD|-\alpha+1} - \left(\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1}\right). \qquad (4.23)
\end{aligned}
$$

As a result, when $\frac{k}{|TD|} \leqslant \frac{x'-\alpha}{|SD|-\alpha}$,

$$
\begin{aligned}
&\ln(PSD(x', SD)) \\
&\approx \ln\left(\sum_{i=x'}^{k}\left(\left(\frac{i}{|TD|}\right)^{x'-\alpha}\left(1-\frac{i}{|TD|}\right)^{|SD|-x'}\right)\right) \\
&\quad + \ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b \\
&\gtrsim \ln\left(\left(\frac{k}{|TD|}\right)^{|SD|-\alpha+1} - \left(\frac{x'}{|TD|}\right)^{|SD|-\alpha+1}\right) - \ln(|SD|-\alpha+1) \\
&\quad + \ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b \qquad (4.24)
\end{aligned}
$$

and

$$
\begin{aligned}
&\ln(PSD(x', SD)) \\
&\lesssim \ln\left(\left(1-\frac{x'}{|TD|}\right)^{|SD|-\alpha+1} - \left(1-\frac{k}{|TD|}\right)^{|SD|-\alpha+1}\right) - \ln(|SD|-\alpha+1) \\
&\quad + \ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b\,. \qquad (4.25)
\end{aligned}
$$

When $\frac{k}{|TD|} > \frac{x'-\alpha}{|SD|-\alpha}$,

$$
\begin{aligned}
&\ln(PSD(x', SD)) \\
&\approx \ln\left(\sum_{i=x'}^{k}\left(\left(\frac{i}{|TD|}\right)^{x'-\alpha}\left(1-\frac{i}{|TD|}\right)^{|SD|-x'}\right)\right) \\
&\quad + \ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b \\
&\gtrsim \ln\left(\left(\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1} + \left(1-\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1}\right.
\end{aligned}
$$

$$-\left(\frac{x'}{|TD|}\right)^{|SD|-\alpha+1} - \left(1-\frac{k}{|TD|}\right)^{|SD|-\alpha+1}\Bigg) - \ln(|SD|-\alpha+1)$$
$$+\ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b \qquad (4.26)$$

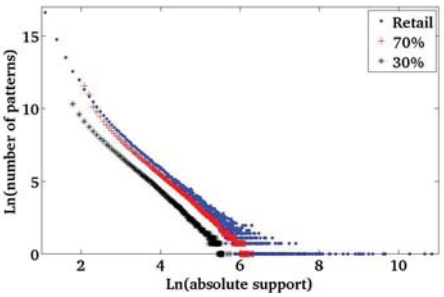
and

$$\begin{aligned}
&\ln(PSD(x', SD)) \\
&\lesssim \ln\Bigg(\left(1-\frac{x'}{|TD|}\right)^{|SD|-\alpha+1} - \left(1-\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1} \\
&\quad +\left(\frac{k}{|TD|}\right)^{|SD|-\alpha+1} - \left(\frac{x'-\alpha}{|SD|-\alpha}\right)^{|SD|-\alpha+1}\Bigg) - \ln(|SD|-\alpha+1) \\
&\quad +\ln(|SD|!) - \ln((|SD|-x')!) - \ln(x'!) - \alpha\ln(|TD|) + b\,. \qquad (4.27)
\end{aligned}$$

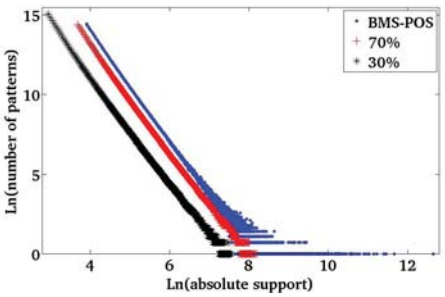
As can be seen from the approximate bounds calculated above, although the form of the approximate bounds is simpler than those of $\ln(PSD(x', SD))$, the relationship between the sample size and the shape deviation on the pattern support distribution still cannot be explicitly seen from the complicated form of the approximate bounds. Thus, experimental studies are used to indicate the general trend of the influence of sampling, especially the chosen sample size, on the power-law-based pattern support distribution of a transaction dataset.

Two real retail transaction datasets—the Retail and BMS-POS datasets—are used in the following experimental studies. Figure 3.4 shows the influence of sampling on the pattern support distributions between the real retail transaction dataset of Retail/BMS-POS and one of its randomly selected samples with a certain sampling ratio (i.e., 50%, 10%, or 1%). Using stratified sampling with replacement and the sampling ratios 30% and 70%, Figure 4.4 shows the mean and corresponding standard deviation of each point in the pattern support distribution for these sampling ratios over 30 trials in a natural log-log plot. Note that the points where the mean result is less than 1 are not shown in Figure 4.4, since it means that the points have a very small probability of existing in the pattern support distributions over the 30 samples.

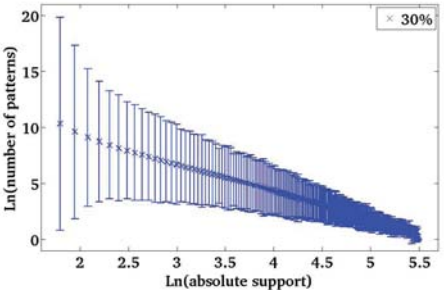
The best-fit discrete power-law distributions to the pattern support distributions shown in Figure 4.4 are calculated. Some related parameter values and statistics are listed in Table 4.1. Since there are a different number of transactions in the original dataset and the samples, the value of $x_{\min}$ is set to be certain values for each original dataset and its sample datasets, so that the best-fit power-law distributions to each



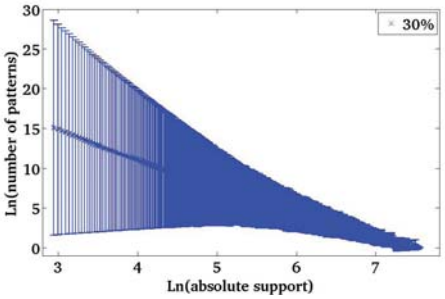
(a) The original PSD of the Retail dataset and the mean PSDs over the 30 samples when the sampling ratio is equal to 30% and 70% respectively



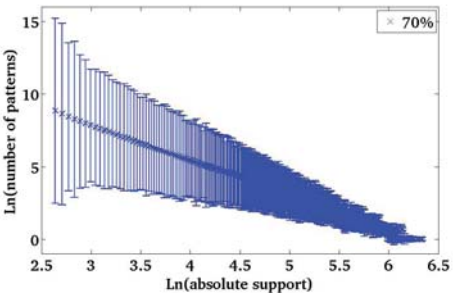
(b) The original PSD of the BMS-POS dataset and the mean PSDs over the 30 samples when the sampling ratio is equal to 30% and 70% respectively



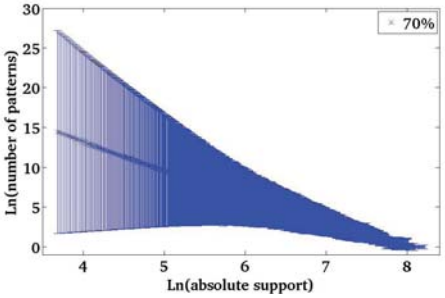
(c) The mean and corresponding standard deviation of each point in the PSD over 30 samples of the Retail dataset when the sampling ratio is equal to 30%



(d) The mean and corresponding standard deviation of each point in the PSD over 30 samples of the BMS-POS dataset when the sampling ratio is equal to 30%



(e) The mean and corresponding standard deviation of each point in the PSD over 30 samples of the Retail dataset when the sampling ratio is equal to 70%



(f) The mean and corresponding standard deviation of each point in the PSD over 30 samples of the BMS-POS dataset when the sampling ratio is equal to 70%

Figure 4.4: Observations on the pattern support distributions of the samples of the Retail and BMS-POS datasets. Note that the axes in these figures are different lengths.

original dataset and its sample datasets are calculated in the same relative support range.

	Retail			BMS-POS		
	The original	70%	30%	The original	70%	30%
$x_{\min}$	20	14	6	63	44	19
mean($\hat{\alpha}$)	2.3600	2.4413	2.7397	3.5300	3.5967	3.7350
stdev($\hat{\alpha}$)		0.0266	0.2255		0.0787	0.1029
mean($D-$statistic)	0.0093	0.0153	0.0649	0.0097	0.0118	0.0160
stdev($D-$statistic)		0.0087	0.0349		0.0014	0.0018

Table 4.1: The best-fit discrete power-law distributions to the pattern support distributions of the corresponding thirty samples drawn from the Retail/BMS-POS dataset with the sampling ratios 70% and 30% respectively

As can be seen in Figure 4.4 and Table 4.1, the shape of the mean pattern support distribution over each 30 samples is different to that of the corresponding original dataset. This clearly shows the existence of the influence of sampling and the effect of the sample size on the pattern support distribution.

As shown in Figure 4.4 and Table 4.1, the scaling exponent values of the power-law-based pattern support distributions of sample datasets are generally larger than that of the power-law-based pattern support distribution of the original dataset, especially when $\frac{|TD|}{|SD|}$ is large. With the decrease of the sample size, the scaling exponent of the power-law-based pattern support distribution becomes larger. One extreme example is that the scaling exponent of the power-law-based pattern support distribution of a sample dataset, which only contains one transaction, is positive infinity no matter what the slope of the pattern support distribution of the original dataset is.

The calculated standard deviations shown in Figure 4.4 and Table 4.1 indicate that with the decrease of the sample size, the scaling exponent of the power-law-based pattern support distribution becomes more unstable. Moreover, Figure 4.4 shows that, with a certain sampling ratio, the high-support part of the pattern support distribution has a lower standard deviation than the low-support part. With the decrease in the sampling ratio, the corresponding standard deviation grows.

As mentioned previously, each point in $PSD\,(x', SD)$ is the sum of the contribution of each pattern in the original dataset. Ideally, such a contribution perfectly follows a binomial distribution. However, the reality is not perfect. Such a contribution does not precisely follow a binomial distribution. Hence, the contribution of patterns in the original dataset is not stable for a point in $PSD\,(x', SD)$. With the decrease of the number of transactions in the sample dataset, in general the

contribution of a pattern in the original dataset to a certain point in $PSD\,(x', SD)$ increases. Since such a contribution is not stable over a single sample dataset of the original dataset, the uncertainty of each point in $PSD\,(x', SD)$ increases with the decrease of the number of transactions over a single sample dataset.

Moreover, a point with a lower absolute support value in $PSD\,(x', SD)$ involves more such contributions compared with a point with a higher absolute support value in $PSD\,(x', SD)$. Therefore, in a single sample dataset, $PSD\,(x', SD)$ becomes more unstable with the decrease of the value of x', although such a contribution may still show a strong behaviour of a binomial distribution to each point in the mean $PSD\,(x', SD)$ over a large number of sample datasets.

As can be seen in Figure 4.4, the maximum support values covered by the pattern support distributions of sample datasets are smaller than those covered by the pattern support distributions of their original datasets. The maximum support value covered by the pattern support distribution of a sample dataset reduces with the decease of the number of transactions in the sample dataset.

However, the maximum relative support value covered by the pattern support distribution of a sample dataset generally increase with the decrease of the number of transactions in the sample dataset. As mentioned previously, each point in $PSD\,(x', SD)$ is the sum of the contribution of each pattern in the original dataset. With the decrease of the number of transactions in the sample dataset, such a contribution on each point in $PSD\,(x', SD)$ increases. Thus, with the decrease of the number of transactions in the sample dataset, the maximum support value covered by $PSD\,(x', SD)$ is closer to $|SD|$ and the maximum relative support value covered by $PSD\,(x', SD)$ increases. One extreme example is that the maximum support value covered by $PSD\,(x', SD)$ of a sample dataset which only has one transaction is one. The maximum relative support value covered by $PSD\,(x', SD)$ of the sample dataset is one, which is the maximum value that a relative support can take.

In addition, the calculated D-statistics listed in Table 4.1 show that with the decrease of the sample size, the distance between the best-fit power-law distribution and the corresponding pattern support distribution becomes larger and more unstable. That means that the pattern support distribution of a sample dataset drawn from a large transaction dataset is unlikely to follow a discrete power-law relationship as closely as the pattern support distribution of the large transaction dataset.

The reason may be that the values of $x_{\min}$ used to estimate the best-fit power-law

distribution to the pattern support distributions of samples are smaller than those of $x_{\min}$ for the pattern support distributions of the corresponding original datasets. Although in this chapter the pattern support distributions of large transaction datasets are assumed to fully follow discrete power-law distributions, in reality with noise and statistical fluctuation a pattern support distribution of a large transaction dataset does not. This difference increases with the decrease of $x_{\min}$. Since, as shown in Table 4.1, the chosen values of $x_{\min}$ for the pattern support distributions of the samples are smaller than these of the original datasets, the values of the D-statistic related to the pattern support distributions of the samples are more affected by their corresponding small $x_{\min}$, and so become larger.

Note that the noise or statistical fluctuation in the high-support part of the pattern support distribution of an original dataset shrinks in the mean pattern support distribution over a number of sample datasets, as shown in Figures 4.4(a) and 4.4(b). Based on the experimental results shown in Figures 4.4(a) and 4.4(b), it seems that the degree of such shrinking increases with the decrease of the number of transactions in the sample datasets. Thus, using the sampling method to estimate the power-law-based pattern support distribution may effectively reduce the effect of the statistical fluctuation and even noise in the pattern support distribution.

This section takes stratified sampling with replacement as an example to theoretically and experimentally discuss the influence of sampling (particularly emphasizing the sample size) on a power-law-based pattern support distribution. The influence of sampling increases with the decrease of the number of transactions in the sample dataset, which is normal in most applications where a basic sampling method (such as stratified sampling) is applied. With the decrease of the number of transactions in a sample dataset, the scaling exponent of the power-law-based pattern support distribution of the sample dataset increases and the maximum support value covered by the pattern support distribution decreases.

However, as can be seen from the experimental results shown in Figure 4.4 and Table 4.1, the scaling exponent of the power-law-based pattern support distribution of a relatively small sample dataset does not have a large difference to the one of the pattern support distribution of the corresponding original dataset. That suggests that although the degree of accuracy of the estimation by using stratified sampling with replacement mostly relies on the sample size applied, a very large sample size may not be a necessary condition for estimating the power-law-based pattern support distribution of the original dataset with a small bias, which gives an

insight for designing more effective and efficient estimation methods in the following section.

Note that the influence of sampling can also be discussed with other sampling methods. The influence of sampling can vary with different sampling methods. In general, a more complex and specially designed sampling method may reduce its influence. However, the execution cost of the sampling may increase at the same time.

4.3 Efficient Estimation of Power-law-based Pattern Support Distributions

This section investigates how to efficiently estimate the power-law-based pattern support distribution of a transaction dataset with relatively high accuracy. Based on two statistics (the mean number of patterns per transaction and the mean absolute support value of patterns), two new estimation methods (an approximate estimation method and a numerical estimation method) are proposed to deal with the problem in different conditions.

Simply using the basic sampling methods discussed in Section 4.2 is not suitable for estimating the power-law-based pattern support distribution of a large transaction dataset. As demonstrated in Section 4.2, the influence of sampling on the pattern support distribution exists, so that the shape of the pattern support distribution of a sample dataset drawn from a large transaction dataset differs from that of the original transaction dataset. The pattern support distribution of a single sample dataset with a certain number of transactions is unstable and unpredictable. Consequently, it is unreliable to use the pattern support distribution of a single sample dataset to estimate that of the large dataset, particularly when the number of transactions in the sample set is small.

Note that with a certain number of transactions in each sample set, the mean pattern support distribution is more stable and predictable, especially when the number of sample datasets increases. However, due to the involved hypergeometric functions, the complicated equations, which can express the influence of sampling discussed in Section 4.2, currently do not have general theoretical solutions. Although a solution to each individual case can still be numerically or approximately calculated, obviously this is not convenient. Additionally, the estimation based on the mean pattern support distribution over a number of sample datasets costs more

compared with a single sample dataset. Accordingly, considering the total cost that includes drawing sample datasets, mining patterns, computing the mean pattern support distribution and estimating the pattern support distribution of the target transaction dataset, simply using basic sampling methods may not be good enough for efficiently estimating the pattern support distribution of a target transaction dataset with relatively high accuracy.

It is normally incorrect to estimate a power-law-based pattern support distribution based only on the high support part, although it costs much less. The patterns in the high support part of the pattern support distribution can be enumerated and counted quickly and inexpensively, since the number of such patterns is small. However, there is generally a large statistical fluctuation in the high support part of the pattern support distribution.

In addition, the number of patterns in the high support part of the pattern support distribution is normally quite small compared with the number of patterns in the target dataset. Since each pattern is equally important in the estimation based on MLE, the estimated result that only relies on the patterns in the high support part cannot represent the majority of the patterns comprising the pattern support distribution. Moreover, there is no guarantee that the shape of the low support part remains consistent with that of the high support part. Hence, the result of estimating a full pattern support distribution based only on the patterns in the high support part is wrong in most cases.

Another issue with this estimating method is how to determine the threshold to distinguish the patterns with the low and high supports. Basically, the smaller the threshold is, the more accurate the estimated power-law-based pattern support distribution is. However, at the same time, the computational cost grows. Therefore, only relying on the patterns in the high support part to estimate the full pattern support distribution may not be the best choice.

An effective and efficient method is proposed here to solve the problem of efficiently estimating the characteristics (i.e., α and b) of the power-law-based pattern support distribution in a real retail transaction dataset. If one knows the mean number of patterns per transaction (denoted by mean(n_{PT}))[1] and the mean absolute support value of patterns (denoted by mean(sup_P)) in the target transaction dataset, based on the assumption made early (i.e., $\ln(PSD\,(x, TD)) = -\alpha \ln(x) + b$),

[1]There are in total $\left(\mathrm{C}_1^{|T|} + \mathrm{C}_2^{|T|} + \ldots + \mathrm{C}_{|T|-1}^{|T|} + \mathrm{C}_{|T|}^{|T|} =\right) 2^{|T|} - 1$ number of patterns in a transaction T with length $|T|$.

the value of α and b can be approximately obtained by solving the simultaneous equations:

$$\text{mean}(n_{PT}) = \frac{\sum_{i=1}^{k} (PSD(x_i, TD)\ x_i)}{|TD|} \tag{4.28}$$

$$\text{mean}(sup_P) = \frac{\sum_{i=1}^{k} (PSD(x_i, TD)\ x_i)}{\sum_{i=1}^{k} (PSD(x_i, TD))}\ , \tag{4.29}$$

where $k \approx \left[e^{\frac{b}{\alpha}}\right]$, which is the maximum absolute support value in $PSD(x, TD)$.

Since $x_i = i$ here,

$$\text{mean}(n_{PT}) = \frac{\sum_{i=1}^{k} \left(i^{-\alpha} e^b i\right)}{|TD|} = \frac{e^b \sum_{i=1}^{k} (i^{1-\alpha})}{|TD|} \tag{4.30}$$

$$\text{mean}(sup_P) = \frac{\sum_{i=1}^{k} \left(i^{-\alpha} e^b i\right)}{\sum_{i=1}^{k} (i^{-\alpha} e^b)} = \frac{\sum_{i=1}^{k} (i^{1-\alpha})}{\sum_{i=1}^{k} (i^{-\alpha})}\ , \tag{4.31}$$

where $\sum_{i=1}^{k} (i^{1-\alpha}) = \sum_{i=1}^{k} \left(\frac{1}{i^{\alpha-1}}\right)$ is the *generalized harmonic number* of order k of $(\alpha - 1)$. Since this form of a generalized harmonic number cannot be simplified, there is no accurate, explicit, theoretical solution to the simultaneous nonlinear equations above. An approximate solution can be calculated by approximate or numerical calculation.

In the limit of $k \to +\infty$, with $\alpha > 1$, the generalized harmonic number $H_{k,\alpha}$ converges to the Riemann zeta function $\zeta(\alpha)$, i.e.,

$$\lim_{k \to +\infty} (H_{k,\alpha}) = \lim_{k \to +\infty} \left(\sum_{i=1}^{k} \left(\frac{1}{i^{\alpha}} \right) \right) = \zeta(\alpha)\ . \tag{4.32}$$

Based on the generalized harmonic number or the Riemann zeta function, if k is very large, the value of k has a very limited impact on the value of the generalized harmonic number and the Riemann zeta function, especially when α is relatively large. For example, $H_{10000,1.5} = 2.5491$ and $H_{50000,1.5} = 2.6034$. Therefore, if k is very large and $(\alpha - 1)$ is relatively large,

$$\text{mean}(sup_P) = \frac{\sum_{i=1}^{k} \left(\frac{1}{i^{\alpha-1}}\right)}{\sum_{i=1}^{k} \left(\frac{1}{i^{\alpha}}\right)} \approx \frac{\zeta(\alpha - 1)}{\zeta(\alpha)}\ . \tag{4.33}$$

When $\alpha > 2$, $\frac{\zeta(\alpha-1)}{\zeta(\alpha)}$ is a monotonically decreasing function of α, as demonstrated in Figure 4.5.

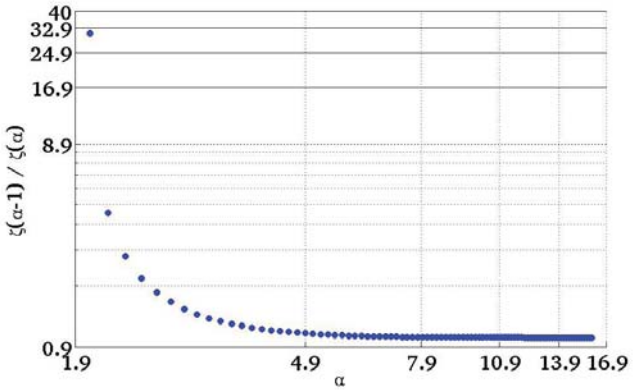


Figure 4.5: $\frac{\zeta(\alpha-1)}{\zeta(\alpha)}$ when $\alpha > 2$

Also, under the same condition, since $k \approx \left\lceil e^{\frac{b}{\alpha}} \right\rceil$,

$$\text{mean}(n_{PT}) = \frac{e^b \sum_{i=1}^{k} \left(\frac{1}{i^{\alpha-1}}\right)}{|TD|} \approx \frac{k^\alpha \zeta(\alpha-1)}{|TD|} \ . \tag{4.34}$$

Therefore, given the value of mean(sup_P) and mean(n_{PT}), the estimators to α and b can be inexpensively computed from Equations 4.33 and 4.34 with relatively high estimation accuracy.

When k is not very large, $(\alpha - 1)$ is not relatively large, or the values of k and $(\alpha - 1)$ are unknown, the following numerical estimation can be applied.

As observed and discussed in Sections 3.3.4 and 4.2, in general, the value of the scaling exponent α_P of the pattern support distribution of a transaction dataset is not less than that of the scaling exponent α_I of the item pattern support distribution (i.e., the special pattern support distribution only including items instead of patterns) in the same dataset, since the support value of a pattern cannot be larger than that of any item in the pattern. Also, the value of α_P in a transaction dataset TD is not larger than that of the scaling exponent $\alpha_P{}'$ in a sample dataset SD randomly drawn from TD. Thus,

$$\alpha_I \leqslant \alpha_P \leqslant \alpha_P{}' \ . \tag{4.35}$$

Also, since $-\alpha \ln(k) + b = 0$, respectively replacing α in Equation 4.30 with α_I and $\alpha_P{}'$, a bound $[k_{\min}, k_{\max}]$ of k_P can be obtained.

Note that in Section 3.3.4 the self-similarity phenomenon is observed between the pattern support distribution and item support distribution of a real transaction

dataset. Moreover, the scaling exponent values of the pattern support distribution and item support distribution of a sample dataset SD randomly drawn from a transaction dataset TD are all affected by the sampling process. Therefore, an inaccurate estimator $\hat{\alpha}_P$ to α_P in TD can be easily computed,

$$\frac{\alpha_P{}'}{\alpha_P} \approx \frac{\alpha_I{}'}{\alpha_I} \,. \tag{4.36}$$

However, since the approximate equation does not express the difference between the pattern support distribution and item support distribution of a transaction dataset, the estimated result by using the approximate equation cannot be accurate.

Actually, based on Equation 4.36, the bounds of α, which are shown by Inequality 4.35, can be approximately reduced to $[\frac{\alpha_P{}' \times \alpha_I}{\alpha_I{}'}, \alpha_P{}']$, since compared with the items, fewer patterns exist in SD. Thus, the computational cost of numerical estimation can be reduced. However, in order to obtain a more reliable result, the bounds of α are still used, as shown by Inequality 4.35.

A more accurate result can be obtained by numerically solving the simultaneous Equations 4.30 and 4.31, with certain values of mean(sup_P), mean(n_{PT}) and $|TD|$. Accurate values of mean(n_{PT}) and $|TD|$ can be inexpensively obtained via scanning TD once. However, an accurate value of mean(sup_P) is too expensive to obtain, since it requires knowing all patterns and their accurate support values in TD. Note that the empirical value mean($sup_P{}'$) of mean absolute support of patterns in SD costs relatively little to compute, since there are a smaller number of patterns in SD. Hence, if the value of mean(sup_P) is unknown, one can use mean($sup_P{}'$) instead of mean(sup_P) to estimate the full power-law-based pattern support distribution in TD.

The relationship between mean($sup_P{}'$) and the full power-law-based pattern support distribution in TD is complicated. Note that the expected number of patterns in SD randomly drawn from TD is equal to the number of patterns in TD minus the expected number of the patterns that are in TD, but not in SD. Assuming that the support of each pattern is independently and identically distributed (i.i.d.) over TD, based on Equation 4.11 and the discussion in Section 4.2, the expected value of mean absolute support of patterns in SD randomly drawn from TD is:

$$\mathrm{E}(\mathrm{mean}(sup_P{}')) = \frac{\frac{|SD|}{|TD|}\sum_{i=1}^{k}\left(PSD\left(x_i, TD\right)\ x_i\right)}{\sum_{i=1}^{k}\left(PSD\left(x_i, TD\right)\left(1 - \mathrm{C}_0^{|SD|}\left(\frac{x_i}{|TD|}\right)^0\left(1 - \frac{x_i}{|TD|}\right)^{|SD|-0}\right)\right)}$$

$$
\begin{aligned}
&= \frac{\frac{|SD|}{|TD|}\sum_{i=1}^{k}\left(PSD\left(x_i, TD\right)\ x_i\right)}{\sum_{i=1}^{k}\left(PSD\left(x_i, TD\right)\left(1-\left(1-\frac{x_i}{|TD|}\right)^{|SD|}\right)\right)} \\
&= \frac{\frac{|SD|}{|TD|}\sum_{i=1}^{k}\left(i^{-\alpha}e^{b}i\right)}{\sum_{i=1}^{k}\left(i^{-\alpha}e^{b}\left(1-\left(1-\frac{i}{|TD|}\right)^{|SD|}\right)\right)} \\
&= \frac{|SD|}{|TD|}\frac{\sum_{i=1}^{k}\left(i^{1-\alpha}\right)}{\sum_{i=1}^{k}\left(i^{-\alpha}\left(1-\left(1-\frac{i}{|TD|}\right)^{|SD|}\right)\right)}\ . \qquad (4.37)
\end{aligned}
$$

Note that the value of E(mean($sup_P{}'$)) in SD, which is used in Equation 4.37, is usually not equal to that of mean($sup_P{}'$) in SD. The value of E(mean($sup_P{}'$)) in SD is equal to the mean value of mean($sup_P{}'$) of w sample datasets randomly drawn from TD with the sampling ratio $\frac{|SD|}{|TD|}$ when $w \to +\infty$. Normally, the value of mean($sup_P{}'$) in SD is closer to the one of E(mean($sup_P{}'$)) in SD when $|SD|$ is closer to $|TD|$.

For the sake of computational simplicity, the value of E(mean($sup_P{}'$)) in Equation 4.37 can be replaced with the one of mean($sup_P{}'$) in SD. That is computationally cheaper, since SD is a relatively small sample dataset. However, in this way the accuracy of the estimated results may not be very high, as the probability that the difference between E(mean($sup_P{}'$)) and mean($sup_P{}'$) is large increases. The more accurate value of E(mean($sup_P{}'$)) can be computed based on a sample dataset with a large number of transactions or a number of such small sample datasets. However, the corresponding cost of computation increases at the same time. Balancing the accuracy and the computational cost varies with different applications.

In addition, the value of k for a pattern support distribution in a transaction dataset must be a positive integer, while the value of α and b must be positive real numbers. Consequently, suitable estimators of α and b of the pattern support distribution of TD may not be obtained by simultaneously solving Equations 4.37 and 4.30.

Actually, it becomes an optimization problem. With given values of $|TD|$, $|SD|$, α_I, $\alpha_P{}'$, mean($sup_P{}'$), and mean(n_{PT}), suitable estimators for α and k are the ones that minimize the sum of the relative errors with respect to Equations 4.37 and 4.30 of two constrained variables α and b, i.e.,

$$
\min\left\{\frac{|f_1 - \text{mean}(sup_P{}')|}{\text{mean}(sup_P{}')} + \frac{|f_2 - \text{mean}(n_{PT})|}{\text{mean}(n_{PT})}\right.
$$

$$\text{such that } \alpha \in [\alpha_I, \alpha_P{}'],\ k \in [k_{\min}, k_{\max}],\ k \in \mathrm{N} \Bigg\} , \tag{4.38}$$

where

$$f_1 = \frac{|SD|}{|TD|} \frac{\sum_{i=1}^{k} (i^{1-\alpha})}{\sum_{i=1}^{k} \left(i^{-\alpha} \left(1 - \left(1 - \frac{i}{|TD|} \right)^{|SD|} \right) \right)} , \tag{4.39}$$

$$f_2 = \frac{k^{\alpha} \sum_{i=1}^{k} (i^{1-\alpha})}{|TD|} . \tag{4.40}$$

Then, the corresponding estimator to b can be obtained by using the equation

$$b = \alpha \ln(k) . \tag{4.41}$$

Since the expression form of the function, i.e.,

$$f = \frac{|f_1 - \text{mean}(sup_P{}')|}{\text{mean}(sup_P{}')} + \frac{|f_2 - \text{mean}(n_{PT})|}{\text{mean}(n_{PT})} \tag{4.42}$$

is too complicated, it is not easy to solve the optimization problem, although there are lots of different methods available.

Here, an approach is used to solve the complicated optimization problem. Note that k must be a positive integer in f. One can reduce the complexity of the optimization problem by turning f of two bounded variables to a function of one bounded variable. For instance, one can compute the local minimum value of f for each possible value of k in $[k_{\min}, k_{\max}]$, and then determine the global minimum value of f in order to attain the corresponding $\hat{\alpha}$ and $\hat{k}$. Since f is continuous in a closed and bounded area (i.e., $\alpha \in [\alpha_I, \alpha_P{}'],\ k \in [k_{\min}, k_{\max}]$), based on the extreme value theorem, f must attain its local minimum values with each certain $\hat{\alpha}/\hat{k}$ at least once. Also, f must attain its global minimum value in the area at least once. That guarantees that the adopted general approach can find the global minimum value of f in the bounded and closed area and the corresponding $\hat{\alpha}$ and $\hat{k}$.

A synthetic transaction dataset instead of a real transaction dataset is used to demonstrate the effectiveness and efficiency of the numerical estimation proposed above. As mentioned previously, it is very computationally expensive to count all patterns and their accurate supports in a large transaction dataset. The full pattern support distribution in a real retail transaction dataset (e.g., the Retail or BMS-POS dataset), where the power-law relationship in a pattern support distribution cannot

be ruled out, is not able to be computed in a reasonable time. However, in this section, the efficient estimation method is designed to identify the full power-law-based pattern support distribution. Therefore, a synthetic transaction dataset is created based on the Retail and BMS-POS datasets and is adopted for the following demonstration.

The synthetic transaction dataset mainly satisfies 4 requirements: 1) the full pattern support distribution in the dataset can be obtained; 2) a power-law relationship appears in the full pattern support distribution; 3) the appearance of the full pattern support distribution in the dataset is similar to the one in a real transaction dataset. For example, the statistical fluctuation can be observed in the pattern support distribution of the synthetic dataset; and 4) the synthetic dataset cannot be too small.

Tables 4.2, 4.3 and 4.4 respectively list the relevant parameters of the synthetic transaction dataset, the best-fit power-law-based pattern support distribution and the best-fit power-law-based item support distribution in the synthetic dataset. Figure 4.6 shows the full pattern support distribution and the full item support distribution of the synthetic transaction dataset, and their corresponding best-fit discrete power-law distributions in the natural log-log plot.

$\|I\|$	$\|TD\|$	$\|T\|_{\max}$	$\|T\|_{\text{avg}}$	$\max(sup_n)$	Number of patterns	mean(n_{PT})	mean(sup_P)
1,562	5,516	13	2.829	328	103,127	26.884	1.438

Table 4.2: Parameters of the synthetic transaction dataset

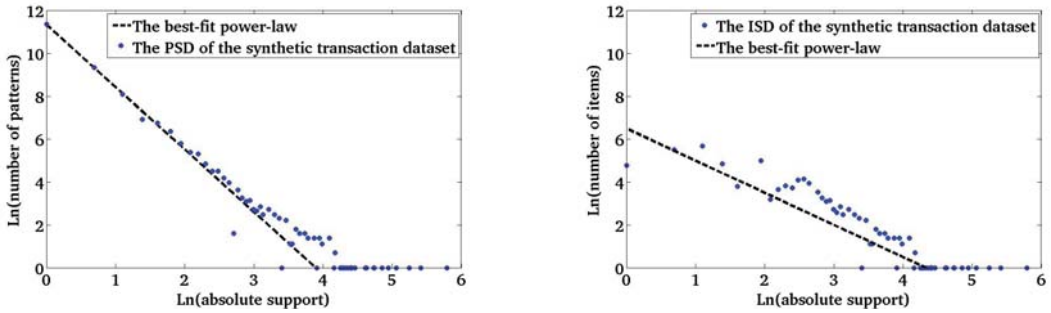
$\hat{\alpha}_P$	$\hat{b}_P$	$\hat{k}_P$	D-statistic
2.91	11.3547	50	0.0040

Table 4.3: Parameters of the best-fit power-law distribution to the full pattern support distribution in the synthetic transaction dataset based on MLE

$\hat{\alpha}_I$	$\hat{b}_I$	$\hat{k}_I$	D-statistic
1.5	6.4862	76	0.3066

Table 4.4: Parameters of the best-fit power-law distribution to the full item support distribution in the synthetic transaction dataset based on MLE

100 transactions as a sample dataset SD are drawn from the synthetic dataset TD by using stratified sampling. Tables 4.5, 4.6 and 4.7 respectively list the relevant



(a) The PSD of the synthetic transaction dataset and its maximum likelihood power-law fit

(b) The ISD of the synthetic transaction dataset and its maximum likelihood power-law fit

Figure 4.6: The pattern support distribution and item support distribution (ISD) of the synthetic transaction dataset and their corresponding maximum likelihood discrete best power-law fits in the natural log-log plot

parameters of SD, the discrete power-law distribution best fitted to the full pattern support distribution, and the discrete power-law distribution best fitted to the full item support distribution in SD. Figure 4.7 shows the full pattern support distribution and the full item support distribution of the sample dataset drawn from the synthetic transaction dataset, and their corresponding best-fit discrete power-law distributions in the natural log-log plot.

$\|I'\|$	$\|SD\|$	$\|T'\|_{\max}$	$\|T'\|_{\text{avg}}$	$\max(sup_n{}')$	Number of patterns	$\text{mean}(n_{PT}{}')$	$\text{mean}(sup_P{}')$
220	100	10	2.88	10	2,931	31.24	1.0658

Table 4.5: Parameters of the sample dataset drawn from the synthetic transaction dataset

$\hat{\alpha_P}'$	$\hat{b_P}'$	$\hat{k_P}'$
4.47	7.928	6

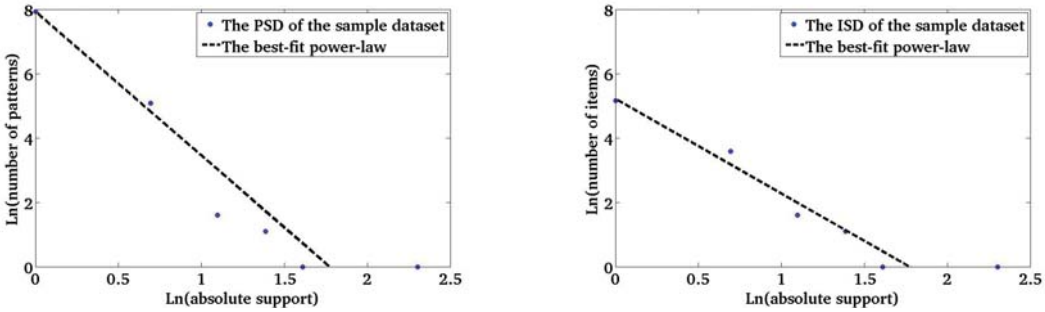
Table 4.6: Parameters of the best-fit power-law distribution to the full pattern support distribution in the sample dataset of the synthetic transaction dataset based on MLE

Based on the approximate equation 4.36, an inaccurate value of the scaling exponent of PSD in TD can be attained, i.e.,

$$\frac{\hat{\alpha_P}' \times \hat{\alpha}_I}{\hat{\alpha_I}'} = \frac{4.47 \times 1.5}{2.94} = 2.2806 < 2.91 = \hat{\alpha}_P \ . \tag{4.43}$$

$\hat{\alpha_I}'$	$\hat{b_I}'$	$\hat{k_I}'$
2.94	5.2107	6

Table 4.7: Parameters of the best-fit power-law distribution to the full item support distribution in the sample dataset of the synthetic transaction dataset based on MLE



(a) The PSD of the sample dataset and its maximum likelihood power-law fit

(b) The ISD of the sample dataset and its maximum likelihood power-law fit

Figure 4.7: The pattern support distribution and item support distribution (ISD) of the sample dataset drawn from the synthetic transaction dataset and their corresponding maximum likelihood discrete best power-law fits in the natural log-log plot

According to Inequality 4.35 and Equation 4.30,

$$1.5 \leqslant \alpha \leqslant 4.47 \,, \tag{4.44}$$

$$13 \leqslant k \leqslant 277 \,. \tag{4.45}$$

Given $|TD| = 5,516$, $|SD| = 100$, mean$(sup_P{}') = 31.24$, and mean$(n_{PT}) = 26.884$, with the certain precision of α (i.e., 10^{-4}) and the certain precision of k (i.e., 1), the global minimum value of f in the closed and bounded area (i.e., $\alpha \in [1.5, 4.47], k \in [13, 277]$) can be determined from the local minimum values of f for each integer between 13 and 277. The determined global minimum value of f is $2.087E-4$. The corresponding $\hat{\alpha}$, $\hat{b}$ and $\hat{k}$ are 2.435, 10.9570 and 90 respectively.

Based on the empirically computed values of $\hat{\alpha}_P$, $\hat{b}_P$ and $\hat{k}_P$ listed in Table 4.3, the relative errors of $\hat{\alpha}$, $\hat{b}$ and $\hat{k}$ respectively are

$$\eta(\hat{\alpha}) = \frac{|2.435 - 2.91|}{2.91} = 0.1632 \,, \tag{4.46}$$

$$\eta(\hat{b}) = \frac{|10.9570 - 11.3547|}{11.3547} = 0.0350 \ , \tag{4.47}$$

$$\eta(\hat{k}) = \frac{|90 - 50|}{50} = 0.8 \ . \tag{4.48}$$

Considering that the cost of mining $100,196 (= 103,127 - 2,931)$ patterns in TD has been saved by using the numerical estimation proposed and these relative errors of $\hat{\alpha}$, $\hat{b}$ and $\hat{k}$, the proposed estimation has achieved relatively high estimation accuracy with relatively high efficiency, although the proposed estimation is not very computationally inexpensive if very high estimation accuracy is required. The efficiency of the numerical estimation can be improved with relatively lower accuracy.

Note that the numerical estimation proposed above is based on the assumption that the full pattern support distribution follows a power-law relationship. As shown in Table 4.3, the D-statistic indicates that the full pattern support distribution in the synthetic transaction dataset does not perfectly satisfy a power-law relationship, which can also affect the accuracy of the numerical estimation.

In addition, for each individual application, there may be some special properties of the application can be used. For instance, in the demonstrating example above, one can observe that the local minimum values show a special curve in Figure 4.8. Based on this special property, the speed of the numerical estimation can be significantly increased.

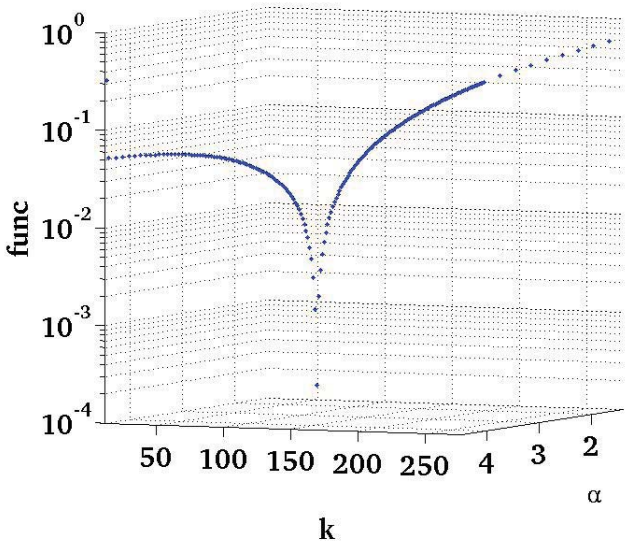


Figure 4.8: The curve of the local minimum values of f in the demonstrating example

When the condition that k is very large and $\alpha - 1$ is relatively large is satisfied, if the value of mean(sup_P) is known, the approximation in Equations 4.33 and 4.34 can be applied, especially since the approximation is generally inexpensive compared with the numerical estimation proposed. When the condition is satisfied, but the value of mean(sup_P) is unknown, one can consider simplifying f_1 and f_2 in order to reduce the computational cost of the numerical estimation. Only when the condition is not satisfied and the value of mean(sup_P) is unknown, the numerical estimation and the general optimization approach are considered to be straightforwardly applied.

In this section, efficient estimation on the characteristics (i.e., α, b and k) of a full power-law-based pattern support distribution in a transaction dataset with relatively high accuracy is discussed. Accurately discovering the characteristics of such a distribution are generally very computationally expensive, since it requires the process of accurately finding all patterns and their supports in the target transaction dataset. To avoid this costly process, two efficient and effective estimations (i.e., an approximate estimation and a numerical estimation) designed for different conditions are proposed in this section. The proposed methods are based on two statistics, i.e., the mean number of patterns per transaction and the mean absolute support value of patterns. As shown in the discussion and the demonstrating example, the proposed methods are able to efficiently estimate the characteristics of a full power-law-based pattern support distribution in a target transaction dataset with relatively high estimation accuracy.

Note that although all discussions in this section are based on the assumption that the full pattern support distribution in a target transaction dataset follows a discrete power-law relationship, the way to analyze and solve this problem can apply with other underlying relationships when there is another relationship showing a better fit to the full pattern support distribution than the power-law one.

4.4 Summary

This chapter investigates how to estimate full power-law-based pattern support distributions of large transaction datasets efficiently.

Accurately estimating the full power-law-based pattern support distribution of a large transaction dataset requires finding all patterns and their supports in the dataset. Usually, this is computationally expensive. Using sample datasets ran-

domly drawn from the original dataset can reduce the cost of the estimation. In Section 4.2 taking stratified sampling as an example, the influence of sampling on the characteristics of the power-law-based pattern support distribution of a large transaction dataset was analyzed theoretically and experimentally. The analysis indicated that, with the decrease in the number of transactions in a sample dataset, the scaling exponent of the power-law-based pattern support distribution of the sample dataset increases and the maximum support value covered by the pattern support distribution decreases.

Due to the uncertainty brought by sampling, the analysis also showed that by simply using sampling methods one cannot reliably and efficiently estimate the power-law-based pattern support distribution of a large transaction dataset from sample datasets, although the scaling exponent of the power-law-based pattern support distribution of a relatively small sample dataset is not massively different to that of the pattern support distribution of the corresponding original dataset.

Based on two statistics (the mean number of patterns per transaction and the mean absolute support value of patterns), two new estimation methods (an approximate estimation method and a numerical estimation method) were proposed to efficiently estimate the characteristics of the full power-law-based pattern support distribution of a large transaction dataset. The theoretical discussion and the demonstrating example showed that the proposed methods can attain relatively high accurate estimators of α, b and k of the full power-law-based pattern support distribution without mining all patterns in the target dataset. Thus, the required computational cost can be reduced.

Note that in this chapter, the pattern support distribution was investigated only on the absolute support of patterns. The power-law-based pattern support distribution can be based on the relative support of patterns too. The most important point is that the estimated value of the scaling exponent parameter α of the power-law relationship in a pattern support distribution is regardless of whether absolute or relative support is adopted in the estimation, since the relative support value of a pattern is equal to the absolute support value of the pattern over the number of transactions in the target dataset.

Chapter 5

Fuzzy Frequent Pattern Mining

Predicting the future is easy. Getting it right is the hard part.

— Howard Frank

In frequent pattern mining algorithms, users are generally required to specify a precise data-dependent value for the minimum support in order to discover frequent patterns in a target dataset. However, normally, users cannot specify a suitable value without detailed knowledge of the features of their target dataset. In most cases, this is unknown to users before they specify the minimum support value and mine the target dataset.

In this chapter, based on the power-law-based pattern support distribution (discussed in Chapters 3 and 4) and fuzzy logic control techniques, we propose a new fuzzy approach to automatically determining a suitable value for the minimum support in order to mine frequent patterns in a target dataset. By using our proposed approach, users are not required to have detailed knowledge of the features of their target dataset. The effectiveness and efficiency of our proposed approach is demonstrated experimentally on different real transaction datasets.

This chapter is organized as follows. First, the existing problems related to the user-specified minimum support value in frequent pattern mining are introduced in Section 5.1. The general ideas about how to solve these problems are presented in Section 5.2. After an overview of fuzzy logic control in Section 5.3 and related work in Section 5.4, our proposed fuzzy approach to automatically determining a suitable minimum support value is presented in Section 5.5. The effectiveness and efficiency of the proposed approach are experimentally evaluated in Section 5.6. Finally, conclusions are given in Section 5.7.

5.1 Introduction

It has been well recognized that mining frequent patterns can be very computationally expensive, since the search space for mining frequent patterns is exponentially large in the number of items. As a result, lots of algorithms have been designed for efficiently mining frequent patterns, such as Apriori [5] and FP-growth [47]. The assumption behind these algorithms is that the patterns that appear frequently (i.e., have high support values) are the interesting ones. People are usually more interested in patterns whose supports are above a pre-specified minimum threshold since the patterns that appear more often in the dataset are more important, e.g. in sales setting they are responsible for a high sales volume [4]. If the support of a pattern is not large enough, it may not be worth being considered, or simply be less favoured in the first place, although it may be found useful later.

Using the minimum threshold has other benefits, which include that: 1) the corresponding computation is simple; 2) no assumptions are made about target datasets; 3) the anti-monotone (downward closure) property [5] of frequent patterns can be easily used to increase the efficiency of searching frequent patterns; 4) the mined results are easily understood and used in target applications [45].

Most of these algorithms need a user-specified threshold value called the minimum support as an input parameter in order to distinguish frequent and infrequent patterns in a target dataset. However, an unsuitable minimum support value can cause problems because the performance of an algorithm generally depends on the specified minimum support. If the value is too large, few or no patterns may be mined from the target dataset, which can provide little useful information to users. Meanwhile, lots of valuable patterns can be missed out. Conversely, if the specified value is too small, a large number of frequent patterns may be mined and the correspondingly computational cost is large. Additionally, users may have to spend lots of time analyzing the mined frequent patterns in order to identify real useful information hidden in the patterns.

Users need to have enough information about the target dataset to be able to specify a suitable and data-dependent minimum support value for a target dataset. Since datasets have different properties, it is quite likely that users may need to specify different values for the minimum support for different datasets, especially datasets from different domains. Thus, the value of the minimum support has to be dependent on the target dataset. Even for the same dataset, a user may specify different values for different investigations. Therefore, without enough information

about their target datasets, users often have to try several values, which can cost lots of computational time and increase the workload of users. As a result, it becomes a tricky task, even for experienced data miners, to specify a suitable value for the minimum support without detailed knowledge of the target dataset [48, 136, 137].

Moreover, the user-specified minimum support value used for mining frequent patterns in a target dataset is a precise value. However, this precise threshold value challenges users because it deviates from the qualitative thinking pattern of human beings [69]. For example, if a user specifies the minimum relative support value to be 80%, it is hard for the user to judge how meaningless are patterns whose relative support is equal to 79%. By nature, people prefer to use qualitative terms to describe what they want, such as 'very frequent' instead of '80%'. That is particularly true when users do not know the accurate features of what they deal with. Thus, it is more convenient for users to give this kind of description of expected results instead of precise threshold values.

Motivated by these problems regarding the user-specified minimum support value, this chapter presents an approach to generating a precise, suitable and data-dependent minimum support value to mine frequent patterns in a target transaction dataset, based on two user-specified approximate data-independent threshold values. By using the proposed approach, users are able to more conveniently and effectively express their requirements and control the expected results.

5.2 Fundamental Ideas

Our proposed approach is to determine a suitable, data-dependent and precise minimum support value based on some user-specified approximate data-independent threshold values. To achieve this goal, some features of the target dataset have to be discovered in advance as a pre-processing step. The pattern support distribution as a feature of a dataset is fitted to this requirement since it depicts how the patterns in a dataset are distributed over the range of the support values in the dataset.

In this chapter, it can be assumed that a power-law relationship exists in the pattern support distribution of a static transaction dataset, as demonstrated in Chapter 4. In case there is another relationship showing a better fit to the pattern support distribution, one can simply replace the power-law relationship with the better-fitted relationship; this does not affect the fundamental idea behind the proposed approach.

Whether something is frequent, and how frequent it actually is, depends on its reference. There are at least two ways to define frequent patterns. When the smallest support value in a target dataset is treated as the reference, frequent patterns should be those patterns whose support values are much greater than the smallest support value. When the mean support value of all patterns in a target dataset is treated as the reference, frequent patterns should be those patterns whose support values are much greater than the mean support value. Most algorithms for mining frequent patterns consider the smallest support value in a target dataset as the reference. As a result, the mined results may not reflect the information about the majority of patterns in a target dataset, which could potentially mislead users trying to understand and use their mined results.

In order to overcome this shortcoming and provide more comprehensive information about frequent patterns in a target dataset, our proposed approach adopts the mean support value of all patterns as the reference, to determine a suitable value for the minimum support. The mean support value of all patterns in a target dataset can be estimated according to the pattern support distribution of the dataset.

Another benefit of using the pattern support distribution of a dataset is that the relationship between the minimum support and the number of patterns whose supports are not less than the minimum support can be revealed by the corresponding cumulative distribution of the pattern support distribution. By using this, users can easily determine a value for the minimum support according to their requirement of the number of expected patterns in a mined result.

The notable disadvantage of using the mean support value of all patterns and the pattern support distribution of a target dataset is the correspondingly prohibitive computational costs. Calculating the mean support value of all patterns and the pattern support distribution of a dataset requires knowing the support values of all patterns in the dataset, and counting all distinct patterns in a large dataset is virtually impossible in most cases. Although the correspondingly computational costs can be reduced by using approximate estimation instead of accurate estimation of the support values of all patterns in a target dataset, the accuracy of the obtained results can be compromised.

Our proposed approach requires users to specify their approximate expectations instead of a precise threshold value. As discussed previously, people prefer to use qualitative terms such as linguistic terms to approximately express the expected result, especially when they do not know the accurate features of their target dataset.

Unfortunately, computers can only directly process precise values and do not understand qualitative terms. To deal with this problem, our proposed approach asks users to specify two numerical reference values, which are the users' qualitative requirements for the frequency and the number of expected patterns mined from the target dataset, instead of a precise minimum support value for the target dataset.

Although users are still required to specify certain numerical values, unlike the requirement of directly specifying a precise minimum support value, the two values required by our proposed approach do not need to be extremely precise, since they are only used to approximately express users' expectations. Based on the two user-specified values, the results mined by using our proposed approach can be close to the results expected by users. Thus, our proposed approach can be closer to the nature of human beings, and allows users to express their requirements more flexibly and control the mined results better.

Based on the pattern support distribution of a target dataset, our proposed approach uses fuzzy logic control to merge and balance the two user-specified values, then generates a suitable, data-dependent and precise value for the minimum support, which is used for mining frequent patterns in the target dataset.

There are three reasons to use fuzzy logic control in our proposed approach. First, the user requirements for the frequency and the number of expected patterns mined from a target dataset can be inconsistent with each other. In order to get mined results that best match the user requirements, the ambiguity reflected by inconsistent user requirements needs to be processed.

Second, the two user-specified values required by our proposed approach qualitatively indicate user requirements. Qualitative information generally provides imprecise quantitative information. Thus, the imprecise quantitative information provided by qualitative requirement of users also needs to be processed when determining a precise value for the minimum support is required.

Third, the information provided by the pattern support distribution of a target dataset is used in the proposed algorithm in order to convert users' data-independent requirements to a data-dependent value for the minimum support. To get the accurate pattern support distribution in a target dataset is very computationally expensive. Therefore, usually an approximate pattern support distribution is used instead. However, although it costs less to get an approximate pattern support distribution, the information provided by the approximate pattern support distribution becomes imprecise. Fuzzy logic control is widely used in situations like this where ambigu-

ous and/or imprecise information needs to be handled in order to provide smooth control.

5.3 Overview of Fuzzy Logic Control

A fuzzy set as an extension of a crisp set (i.e., a classical bivalent set) was introduced by Zadeh [127]. A crisp set only allows elements to have full membership (i.e., belonging to the crisp set) or no membership at all (i.e., not belonging to the crisp set), and so a crisp membership function value is either 1 or 0. Unlike a crisp set, a fuzzy set allows its elements to have degrees of membership (i.e., partial membership). This can be described by a fuzzy membership function that can take real values in the closed interval $[0, 1]$. The fuzzy membership function value being 0 means that the corresponding element certainly does not belong to the fuzzy set, while the fuzzy membership function value being 1 means that the corresponding element belongs to the fuzzy set fully. Hence, crisp sets can be considered as a special case of fuzzy sets.

For example, in the theory of crisp sets, if anyone who is not shorter than 1.85 metres belongs to the set of 'tall people', then someone who is 1.8499 metres high does not belong to the set of 'tall people'. There is a sharp change between 1.8499 and 1.85, which is against our common sense. In the theory of fuzzy sets, the corresponding membership function value can indicate to what degree a person belongs to the set of tall people, which better fits our common sense.

Formally, let U be a collection of objects (a universe of discourse) denoted generically by x. Then a crisp set A in U is defined as:

$$A = \{x | \mu_A(x) = 1,\ x \in U\}\ , \tag{5.1}$$

where $\mu_A(x)$ is called the membership function for the crisp set A. $\mu_A(x)$ maps objects of U to a two-element set $\{0, 1\}$:

$$\mu_A : U \rightarrow \{0, 1\}\ . \tag{5.2}$$

And,

$$\forall x \in U,\ \mu_A(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}\ . \tag{5.3}$$

A fuzzy set B in U is defined as a set of ordered pairs:

$$B = \{(x, \mu_B(x)) | x \in U\} \ , \tag{5.4}$$

where $\mu_B(x)$ is called the membership function for the fuzzy set B. $\mu_B(x)$ maps objects of U to the closed interval $[0, 1]$:

$$\mu_B : U \rightarrow [0, 1] \ . \tag{5.5}$$

And,

$$\forall x \in U, \ 0 \leqslant \mu_B(x) \leqslant 1 \ . \tag{5.6}$$

The concept of fuzzy logic was conceived with the development of the fuzzy set theory by Zadeh [127]. Fuzzy logic is an extension of Boolean logic, i.e., classical bivalent logic. Fuzzy logic is based on fuzzy set theory. This allows fuzzy logic to describe vagueness as a continuous function between two extremes: true (i.e., 1) and false (i.e., 0). Thus, fuzzy logic provides a mathematical framework where imprecise and vague information can be analyzed by allowing an element to simultaneously belong to more than one set.

Fuzzy logic is often confused with stochastic probability. Fuzzy logic is used to deal with the uncertainty associated with vagueness and ambiguity, while probabilistic methods are used to handle the uncertainty derived from stochastic variability.

Fuzzy logic control refers to a non-linear machine control technique based on fuzzy logic, which can enable machines to reason in a manner similar to humans. The basic idea of fuzzy logic control was first proposed by Zadeh [128, 129]. Human beings can handle vague, imprecise information and qualitative decision-making problems, while machines and computers can only deal with binary computation and reasoning. By using fuzzy logic control techniques, machines and computers are able to be used to analyze vague, imprecise information and qualitative terms such as linguistic terms.

Fuzzy logic control generally involves a large number of fuzzy rules of the "IF-THEN" form, such as "if condition then consequence" and "if condition then action" where fuzzy terms, especially linguistic terms, are allowed in the expression of "condition", "consequence" and "action". Often in practice, the "condition" expression includes several logic subexpressions connected by logic operators such as NOT, AND and OR. Most of these rules are only related to some special conditions.

Thus, such rules are only activated by the occurrence of their related conditions. In this way, adding extra rules requires little additional computational cost. Therefore, the rules used by a fuzzy logic controller can remain stable and understandable, which can also lead to efficient coding and documentation.

Fuzzy logic control has another advantage, which is that the solution to a problem can be presented in terms that humans can understand. Consequently, human operators can more easily understand the results generated by a fuzzy logic controller. Meanwhile, the experience of human operators can be more easily used in designing the fuzzy logic controller as well.

The first fuzzy logic controller was reported by Mamdani and Assilian [77]. The basic structure of a fuzzy logic controller is shown in Figure 5.1.

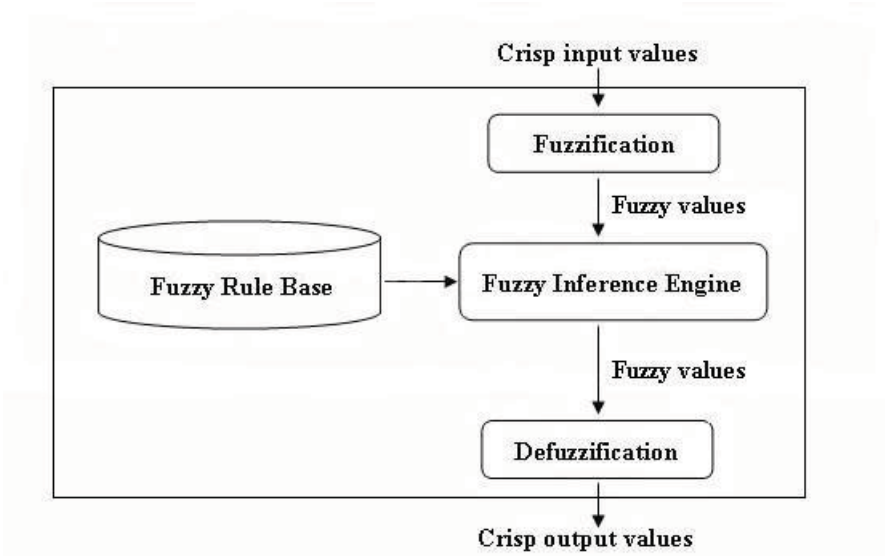


Figure 5.1: Basic structure of a fuzzy logic controller

In general, designing a fuzzy logic controller has several typical steps:

System analysis is the process of analyzing the expected fuzzy logic controller and determining the inputs and outputs. The ranges of the inputs and outputs need to be determined as well.

Setting fuzzy sets and membership functions is the process of setting the fuzzy sets and their corresponding membership functions in order to assign fuzzy sets to fuzzy variables. The universe of discourse (i.e., range) of a fuzzy variable (e.g., linguistic variable) is divided into fuzzy sets defined by the membership functions. Fuzzy sets represent the fuzzy value of the fuzzy variable for an input x in the range of the fuzzy variable.

For example, a fuzzy variable "the speed of a car" can be divided into three fuzzy sets, "Slow", "Moderate" and "Fast". The universe of discourse of the fuzzy variable can be $[20, 80]$ kilometres per hour. An input x in $[20, 80]$ belongs to the three fuzzy sets with different degrees of membership.

Although there are many fuzzy membership functions with different curves that can be used, triangular or trapezoidal shaped membership functions are the most common because they can be easily implemented.

Defining the fuzzy rule set is the process of determining and setting the IF-THEN rules based on the users' expectations and/or the experiments of experts in order to make decisions.

Fuzzification is the process of decomposing inputs into one or more fuzzy sets with their corresponding degrees of membership.

Inference and composition is the process of combining the logical products for the active rules into a single fuzzy set as a fuzzy output with its corresponding membership function value. There are two widely used inference-and-composition methods [14]: the max-min and max-product methods. The max-min method is the combination of MAX composition and MIN inference, while the max-product method is the combination of SUM composition and PRODUCT inference.

Defuzzification is the process of translating fuzzy outputs into crisp numerical values that best represent the corresponding fuzzy outputs. There are two commonly used defuzzification methods [103, 108]: the centroid and maximum methods. The centroid method is used to calculate the centre of gravity of the composite area representing the fuzzy output. The maximum method is used to compute the maximum membership function value of the fuzzy output in the defined fuzzy sets.

These typical steps will be followed to design a new fuzzy logic controller for our proposed approach to mining frequent patterns. The design of the new controller will be presented later in this chapter.

5.4 Related Work

There are lots of applications of fuzzy logic control in mining frequent patterns in a static transaction dataset. However, most of them, such as [22, 52, 53], aim at mining frequent patterns from the transaction datasets with quantitative values. The basic idea behind them is to define a set of linguistic labels with semantics given by fuzzy sets on the domain of the quantitative values in a target dataset, and to use them as a new domain. The quantitative values in each transaction can then be transformed to the labels in the new domain. In this way, the meaning of the information based on the labels is clearer. Also, the information based on the labels is not sensitive to small changes in the boundaries of the defined labels because they are fuzzy terms. This means that the mined results are more stable.

Some work in the literature touches on the issue of specifying a suitable value for the minimum support. For example, Silverstein et al. [109] proposed using the chi-squared significance test instead of specifying ad-hoc values for the minimum support. Hilderman [50] used a heuristic-based method for determining the usefulness of patterns. Piatetsky-Shapiro and Steingold pointed out that only the top 10% or 20% of the prospects with the highest score are usually discovered in database marketing [95]. For mining with temporal data, Roddick and Rice [102] suggested that the thresholds should be modified over time and be context dependent. Instead of asking users to specify the minimum support to identify interesting patterns, Sahar [104] proposed a different approach that asks users to identify several non-interesting rules and then uses these rules to eliminate other similar association rules as non-interesting rules in a target dataset. Some work considered only mining the top-k most frequent patterns instead [51, 82]. Some work [88, 115] attempted to design an alternative interest measure instead of the minimum support for mining association rules. These techniques endeavor to avoid asking users to set the minimum support to some extent. Moreover, they do not allow users to approximately express their expectations.

An approach called FARDIMS [136, 137] is credited for being the first to actually attempt to directly solve the problem of specifying a suitable minimum support value. FARDIMS uses fuzzy logic control, with the mean support of patterns and the 'lean' indicating the relationship between the mean support and the median support of patterns, in order to generate a suitable and data-dependent value for the minimum support based on a user-specified data-independent threshold. This is the most related work to our proposed approach.

Although the basic ideas behind the methods are similar, there are significant differences between our proposed approach and FARDIMS. First, the parameters used in the two approaches are different, and second, the mathematical models used for estimating the values of these parameters are different. Since the mathematical model used in FARDIMS does not present any knowledge about the pattern support distribution of a real transaction dataset, in general, the estimated results by using FARDIMS for real transaction datasets, in particular for real retail transaction datasets, may not be as accurate as the ones found by using our approach, which is based on the power-law relationship.

Third, FARDIMS does not offer a way to explicitly control the number of mined patterns. Although the 'lean' parameter of FARDIMS reflects the support of the majority of patterns in a target dataset, it has limited influence on generating a minimum support value to control the number of mined patterns.

5.5 Algorithm Design and Implementation

5.5.1 Design of a New Fuzzy Logic Controller

In this section, based on the typical steps of designing a fuzzy logic controller, which were reviewed in Section 5.3, we design a new fuzzy logic controller to convert user-specified reference values to a suitable minimum support value used to identify frequent patterns in a target dataset.

System Analysis

Let TD be a target transaction dataset, $PSD(TD)$ be the estimated power-law-based pattern support distribution of TD and $CPSD(TD)$ be the estimated power-law-based cumulative pattern support distribution of TD. Note that with the existence of noise, the power-law relationship in $CPSD(TD)$ cannot be accurately estimated based only on the estimated power-law relationship in $PSD(TD)$ in reality.

Our fuzzy logic controller takes two user-specified numerical reference values, i.e., the frequency ref_f and the number ref_s of expected patterns mined from a target dataset as inputs, as described in Section 5.2. ref_f can be any rational number with finite decimal representation in the interval $[0, 1]$. A larger value of ref_f indicates that expected patterns have higher frequency, i.e., larger support values. By simply assuming that users are not interested in the patterns whose support values are

less than the mean support value mean(sup_{all}) of all patterns in TD, ref_s can be any natural number in the interval $[n, m]$, where n is the number of patterns whose support values are expected to be not less than the maximum support value k_P covered by $PSD(TD)$ in TD and m is the number of patterns whose support values are expected to be not less than mean(sup_{all}). A larger value of ref_s indicates that users expect a larger number of patterns.

Based on the estimated $PSD(TD)$, the mean(sup_{all}) can be computed, i.e.,

$$\text{mean}(sup_{all}) = \frac{\sum_{i=1}^{k_P}(i^{-\alpha_P}e^{b_P} \times i)}{\sum_{i=1}^{k_P}(i^{-\alpha_P}e^{b_P})} = \frac{\sum_{i=1}^{k_P}(i^{1-\alpha_P})}{\sum_{i=1}^{k_P}(i^{-\alpha_P})} , \tag{5.7}$$

where k_P is the maximum support value covered by $PSD(TD)$, α_P is the scaling exponent of $PSD(TD)$ and b_P is the Y-intercept of $PSD(TD)$ in the natural log-log plot. Therefore, based on the estimated $CPSD(TD)$, n and m can be estimated as follows,

$$n = {k_P}^{-\alpha_{CP}}e^{b_{CP}} \tag{5.8}$$

and

$$m = \text{mean}(sup_{all})^{-\alpha_{CP}}e^{b_{CP}} , \tag{5.9}$$

where k_P is the maximum support value covered by $PSD(TD)$, α_{CP} is the scaling exponent of $CPSD(TD)$ and b_{CP} is the Y-intercept of $CPSD(TD)$ in the natural log-log plot. The value of m will be provided so that users can more easily specify the value of ref_s.

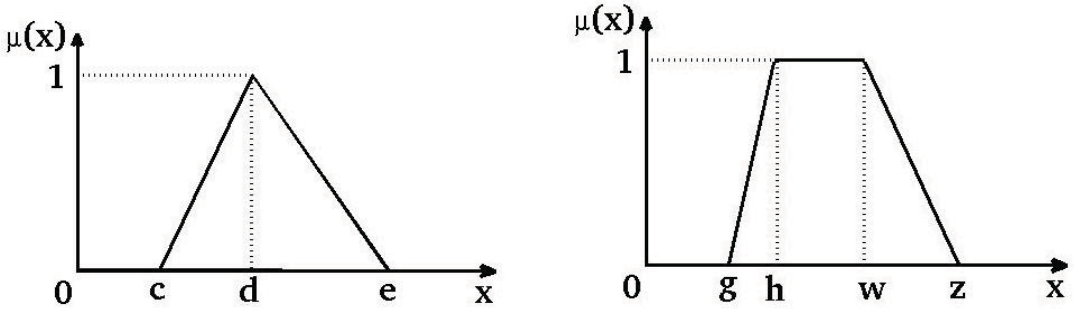
Correspondingly, our designed fuzzy logic controller generates a natural number in the interval [mean(sup_{all}), k_P] for the minimum support min_sup as an output.

Setting Fuzzy Sets and Membership Functions

For the sake of simplicity, in this chapter only five fuzzy sets are defined for each value range of inputs (i.e., ref_f and ref_s) and output (i.e., min_sup). Note that normally, with the increase of the number of fuzzy sets defined, smoother and better control of the output can be expected.

The sets of the fuzzy sets defined in the range of ref_f, ref_s and min_sup are {Very Low Frequency (VLF), Low Frequency (LF), Medium Frequency (MF), High Frequency (HF), Very High Frequency (VHF)}, {Very Small Size (VSS), Small Size (SS), Medium Size (MS), Large Size (LS), Very Large Size (VLS)} and {Very Small (VS), Small (S), Medium (M), Large (L), Very large (VL)} respectively.

The shapes used for the membership functions in this chapter are triangular and trapezoidal, as shown in Figure 5.2, since they can be easily computed and provide reasonable flexibility to the controller. Besides, the study in this chapter aims to present a fuzzy method to solve the minimum support issue and to demonstrate that the fuzzy method works even with the simplest-shaped fuzzy membership functions.



(a) A triangular shaped fuzzy membership function

(b) A trapezoidal shaped fuzzy membership function

Figure 5.2: Triangular and trapezoidal shaped fuzzy membership functions

The frequency level of expected patterns can be smoothly controlled by adjusting the minimum support value. Thus, the corresponding triangular and trapezoidal membership functions can be specified as follows:

$$\mu_{VLF}(ref_f) = \begin{cases} 1, & \text{if } ref_f \in [0, 0.1] \\ \frac{0.3 - ref_f}{0.3 - 0.1}, & \text{if } ref_f \in [0.1, 0.3] \\ 0, & \text{if } ref_f \in [0.3, 1] \end{cases}, \tag{5.10}$$

$$\mu_{LF}(ref_f) = \begin{cases} 0, & \text{if } ref_f \in [0, 0.1] \cup [0.5, 1] \\ \frac{ref_f - 0.1}{0.3 - 0.1}, & \text{if } ref_f \in [0.1, 0.3] \\ \frac{0.5 - ref_f}{0.5 - 0.3}, & \text{if } ref_f \in [0.3, 0.5] \end{cases}, \tag{5.11}$$

$$\mu_{MF}(ref_f) = \begin{cases} 0, & \text{if } ref_f \in [0, 0.3] \cup [0.7, 1] \\ \frac{ref_f - 0.3}{0.5 - 0.3}, & \text{if } ref_f \in [0.3, 0.5] \\ \frac{0.7 - ref_f}{0.7 - 0.5}, & \text{if } ref_f \in [0.5, 0.7] \end{cases}, \tag{5.12}$$

$$\mu_{HF}(ref_f) = \begin{cases} 0, & \text{if } ref_f \in [0, 0.5] \cup [0.9, 1] \\ \frac{ref_f - 0.5}{0.7 - 0.5}, & \text{if } ref_f \in [0.5, 0.7] \\ \frac{0.9 - ref_f}{0.9 - 0.7}, & \text{if } ref_f \in [0.7, 0.9] \end{cases}, \tag{5.13}$$

$$\mu_{VHF}(ref_f) = \begin{cases} 0, & \text{if } ref_f \in [0, 0.7] \\ \frac{ref_f - 0.7}{0.9 - 0.7}, & \text{if } ref_f \in [0.7, 0.9] \\ 1, & \text{if } ref_f \in [0.9, 1] \end{cases}. \tag{5.14}$$

The corresponding membership functions of the fuzzy sets {Very Low Frequency (VLF), Low Frequency (LF), Medium Frequency (MF), High Frequency (HF), Very High Frequency (VHF)} are shown in Figure 5.3.

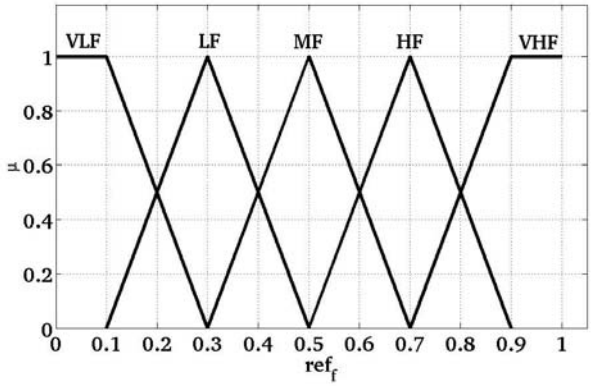


Figure 5.3: The corresponding membership functions of the fuzzy sets defined in the range of ref_f

Normally, there are a huge number of patterns in a real transaction dataset and the number of patterns grows exponentially with the decrease of the minimum support value. The distribution of patterns over support values can be quite different from one dataset to another. In addition, with the existence of noise in real datasets, the number of patterns whose support values are not less than a certain value in a transaction dataset cannot be accurately estimated from $PSD(TD)$. Thus, $CPSD(TD)$ needs to be estimated as well. As a result, the membership functions defined in the range of ref_s need to be determined based on the information provided by the estimated $PSD(TD)$ and $CPSD(TD)$.

Based on our experiments, there is a simple way to determine the fuzzy membership functions for the fuzzy sets defined in the value range of ref_s. One can evenly

divide the value range of min_sup into 10 units, i.e.,

$$q_i = \text{mean}(sup_{all}) + (k_P - \text{mean}(sup_{all}))/10 \times i \ , \tag{5.15}$$

where q_i is the ith cutting point, $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, mean($sup_{all}$) is the mean support value of all patterns in TD and k_P is the maximum support value covered by $PSD(TD)$. According to the estimated $CPSD(TD)$, the corresponding number p_i of patterns whose support values are not less than q_i can be computed as:

$$p_i = (q_i)^{-\alpha_{CP}} e^{b_{CP}} \ , \tag{5.16}$$

where $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, α_{CP} is the scaling exponent of $CPSD(TD)$ and b_{CP} is the Y-intercept of $CPSD(TD)$ in the natural log-log plot. The corresponding triangular and trapezoidal membership functions defined for ref_s can be specified based on p_1, p_3, p_5, p_7, and p_9, which are shown in Figure 5.4. In this way, the controller can have more control of the number of mined patterns with higher support values, which are usually those patterns that users are more interested in.

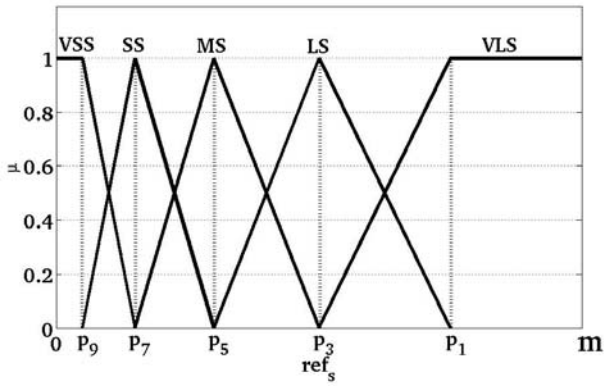


Figure 5.4: The corresponding membership functions of the fuzzy sets defined in the range of ref_s, where m is the number of patterns whose support values are expected to be not less than the mean support value mean(sup_{all}) of all patterns in a target dataset

The corresponding membership functions of the fuzzy sets {Very Small Size (VSS), Small Size (SS), Medium Size (MS), Large Size (LS), Very Large Size (VLS)} are shown in Figure 5.4.

Corresponding to the way that the fuzzy membership functions for ref_f and ref_s

are defined, one can evenly divide the value range of *min_sup* into 10 units:

$$q_i = \text{mean}(sup_{all}) + (k_P - \text{mean}(sup_{all}))/10 \times i \ , \tag{5.17}$$

where q_i is the ith cutting point, $i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, mean($sup_{all}$) is the mean support value of all patterns in TD and k_P is the maximum support value covered by $PSD(TD)$. The corresponding triangular and trapezoidal membership functions defined for *min_sup* can be specified based on q_1, q_3, q_5, q_7, and q_9, which are shown in Figure 5.5. This can reduce the chance that the number of patterns mined from TD is too far away from users' expectations.

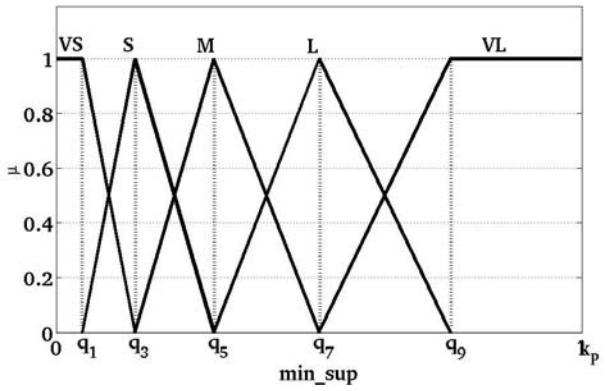


Figure 5.5: The corresponding membership functions of the fuzzy sets defined in the range of *min_sup*

The corresponding membership functions of the fuzzy sets {Very Small (VS), Small (S), Medium (M), Large (L), Very large (VL)} are shown in Figure 5.5.

Defining the Fuzzy Rule Set

The "IF-THEN" rules used in the controller have the form:

$$\text{IF } ref_f \text{ is } FS_1 \text{ and } ref_s \text{ is } FS_2 \text{ THEN } min\_sup \text{ is } FS_3$$

where FS_1 is a fuzzy set in {Very Low Frequency (VLF), Low Frequency (LF), Medium Frequency (MF), High Frequency (HF), Very High Frequency (VHF)}, FS_2 is a fuzzy set in {Very Small Size (VSS), Small Size (SS), Medium Size (MS), Large Size (LS), Very Large Size (VLS)}, and FS_3 is a fuzzy set in {Very Small (VS), Small (S), Medium (M), Large (L), Very large (VL)}.

Table 5.1 shows the "IF-THEN" rules adopted for our controller. Given a certain fuzzy set for ref_s and ref_f, there is an intersection that represents an output fuzzy set.

		ref_f				
		VLF	LF	MF	HF	VHF
ref_s	VSS	M	L	L	VL	VL
	SS	M	M	L	L	VL
	MS	S	M	M	L	L
	LS	S	S	M	M	L
	VLS	VS	S	S	M	M

Table 5.1: A table of fuzzy rules

For example, ref_f being LF and ref_s being SS leads to the fuzzy rule:

IF ref_f is LF and ref_s is SS THEN min_sup is M.

That means that if the user-specified frequency level belongs to "Low Frequency (LF)" and the number of expected patterns belongs to "Small Size (SS)", the output fuzzy set generated by the designed controller is "Medium (M)".

Fuzzification

Based on the fuzzy membership functions defined in the ranges of ref_f and ref_s, the values of ref_f and ref_s, which are specified by the user, are mapped into the fuzzy sets shown in Figures 5.3 and 5.4. After fuzzification, each of ref_f and ref_s are mapped into at most two fuzzy sets with non-zero membership function values.

Inference and Composition

Without loss of generality, the proposed controller adopts the max-min method, which is a widely used inference-and-composition method, to generate fuzzy outputs based on the mapped fuzzy sets and the defined fuzzy rules shown in Table 5.1.

Defuzzification

The proposed controller adopts the centroid method to convert fuzzy outputs to a crisp numerical value as the determined value for the minimum support. Note that the minimum support value has to be an integer. If the crisp numerical value determined by the designed controller is not an integer, the nearest integer that is not less than the determined crisp numerical value is generated as the output.

The following example is used to demonstrate the specific design process of our proposed fuzzy logic controller.

Assume that the target dataset be a transaction dataset TD and the power-law-

based pattern support distribution of TD is

$$\begin{aligned} PSD(x, TD) &= x^{-\alpha_P} e^{b_P} \\ &= x^{-\alpha_P} e^{\alpha_P \ln(k_P)}, \end{aligned} \tag{5.18}$$

where the scaling exponent α_P is 2.91, the maximum support value k_P covered by $PSD(TD)$ is 49 and the Y-intercept b_P of $PSD(TD)$ in the natural log-log plot is 11.33. The scaling exponent α_{CP} and the maximum support value k_{CP} covered by the power-law-based cumulative pattern support distribution $CPSD(TD)$ are 1.91 and 380 respectively. Based on the $PSD(TD)$, the mean support value (denoted as mean(sup_{all})) of all patterns in TD is 1.40 and the number of patterns m whose support values are not less than 1.40 is $22,512$.

Based on the methods described previously, the corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_f in the demonstrating example are shown in Figure 5.3, the corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_s are shown in Figure 5.6, and the corresponding fuzzy membership functions of the fuzzy sets defined in the range of min_sup are shown in Figure 5.7.

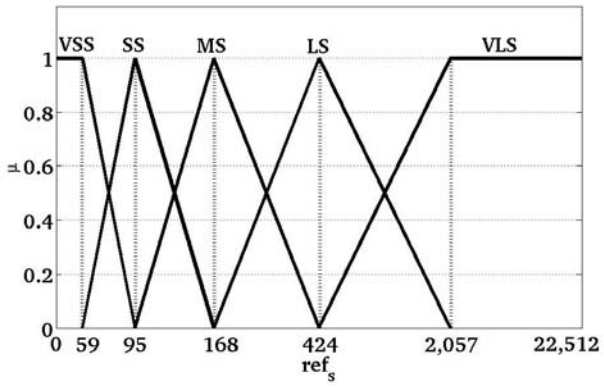


Figure 5.6: The corresponding membership functions of the fuzzy sets defined in the range of ref_s in the example

Assume that the two user-specified reference values ref_f and ref_s are 0.35 and $1,000$ respectively. After fuzzification, one can obtain

$$\begin{aligned} \mu(ref_f) &= \{\mu_{VLF}(0.35), \mu_{LF}(0.35), \mu_{MF}(0.35), \mu_{HF}(0.35), \mu_{VHF}(0.35)\} \\ &= \{0, 0.75, 0.25, 0, 0\} \end{aligned} \tag{5.19}$$

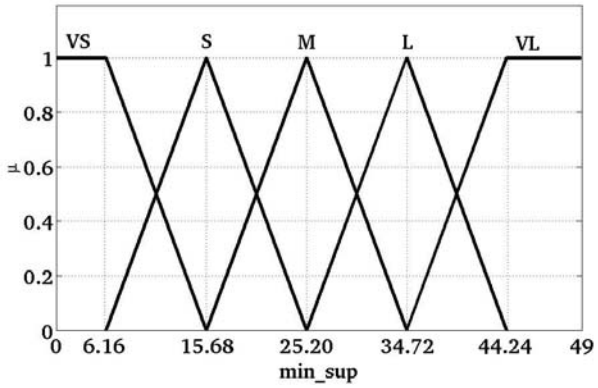


Figure 5.7: The corresponding membership functions of the fuzzy sets defined in the range of *min_sup* in the example

and

$$\begin{aligned} \mu(ref_s) &= \{\mu_{VSS}(1,000), \mu_{SS}(1,000), \mu_{MS}(1,000), \mu_{LS}(1,000), \mu_{VLS}(1,000)\} \\ &= \{0, 0, 0, 0.65, 0.35\} \ . \end{aligned} \tag{5.20}$$

Based on the fuzzy rules shown in Table 5.1, by using the max-min method, one can have the fuzzy output as

$$\begin{aligned} \mu(min\_sup) &= \{\mu_{VS}, \mu_S, \mu_M, \mu_L, \mu_{VL}\} \\ &= \{0, 0.65, 0.25, 0, 0\} \ . \end{aligned} \tag{5.21}$$

By using the centroid method, the crisp numerical value is determined as

$$\frac{0.65 \times 15.68 + 0.25 \times 25.20}{0.65 + 0.25} = 18.32 \ . \tag{5.22}$$

Thus, the *min_sup* value generated by our proposed controller is 19, which is the nearest integer that is not less than the determined crisp numerical value 18.32, when the two user-specified reference values ref_f and ref_s are 0.35 and $1,000$ respectively.

5.5.2 Approximate Estimation of Power-law-based Pattern Support Distributions and their cumulative distributions

When the power-law-based $PSD(TD)$ in a target dataset is unknown, for the sake of computational efficiency, especially in this chapter, the very accurate $PSD(TD)$

may not be necessary, and the $PSD(TD)$ can be roughly estimated by using the following method.

One can randomly draw a small number of transactions from TD as a small sample dataset SD. During the sampling process, the item support distribution of TD and its best-fit power-law relationship can be discovered inexpensively. Since SD is small, it also costs less to compute the item support distribution, the pattern support distribution of SD and their best-fit power-law relationships.

The influence of the sampling process on the item and pattern support distributions are similar, since it affects both the item and pattern support distributions of TD and the item support distribution is a special kind of pattern support distribution. Thus, from our informal tests, the following approximations can be considered,

$$\frac{\alpha_{I'}}{\alpha_I} \approx \frac{\alpha_{P'}}{\alpha_P} \tag{5.23}$$

and

$$\frac{\ln(k_{I'})}{\ln(k_I)} \approx \frac{\ln(k_{P'})}{\ln(k_P)} \,, \tag{5.24}$$

where $\alpha_{I'}$, α_I, $\alpha_{P'}$, and α_P represent the scaling exponents of the item support distributions in the sample and TD, and the pattern support distributions of the sample and TD respectively, $k_{I'}$, k_I, $k_{P'}$, and k_P represent the maximum support values covered by the item support distributions of the sample and TD, and the pattern support distributions of the sample and TD respectively.

Theoretically, the scaling exponent value of a cumulative distribution of a power-law distribution is approximately equal to that of the power-law distribution minus one. Thus, in the same support range where the pattern support distribution of TD follows a power-law relationship, the scaling exponent α_{CP} of the power-law relationship in the cumulative pattern support distribution $CPSD(TD)$ of TD approximately satisfies:

$$\alpha_{CP} \approx \alpha_P - 1 \,, \tag{5.25}$$

where α_P is the scaling exponent of the power-law relationship in the pattern support distribution of TD.

Note that here the maximum support value k_{CP} covered by the power-law relationship in the cumulative pattern support distribution of TD cannot be estimated based on α_{CP}, α_P and k_P calculated above, since α_{CP}, α_P and k_P calculated based on the approximations above are estimated results. If directly using them to estimated k_{CP}, the estimated k_{CP} can be far away from the real one.

From our informal tests, k_{CP} can be estimated as follows:

$$\frac{k_{CP'}}{|SD|} \approx \frac{k_{CP}}{|TD|}\ , \tag{5.26}$$

where $|SD|$ is the number of transactions in SD, $|TD|$ is the number of transactions in TD, and $k_{CP'}$ is the maximum support value covered by the power-law relationship, whose scaling exponent value is equal to $\alpha_{P'} - 1$, in $CPSD(TD)$.

Hence, the power-law-based pattern support distribution and the power-law-based cumulative pattern support distribution of TD can be estimated from a sample dataset of TD. Then, the estimated two distributions can be used to compute the fuzzy membership functions used in the proposed fuzzy logic controller.

5.5.3 Implementation

The proposed approach is outlined below:

Input: A transaction dataset TD, the frequency ref_f and the number ref_s of user-expected patterns mined from TD.

Output: The frequent patterns that satisfy the users' expectations represented via ref_f and ref_s.

Method:

1. *If the power-law-based $PSD(TD)$ is unknown, quickly estimate $PSD(TD)$ and $CPSD(TD)$.*
2. *Based on the estimated power-law-based $PSD(TD)$ and $CPSD(TD)$, set the membership functions for the fuzzy sets defined in the ranges of ref_f, ref_s and min_sup in order to create a fuzzy logic controller.*
3. *Show users the total number of patterns above the mean support of all patterns in TD, which can be estimated based on the power-law-based $CPSD(TD)$, and ask users to specify the two reference values ref_f and ref_s.*
4. *Taking ref_f and ref_s as inputs, the controller outputs a crisp numerical value.*

5. *Taking the crisp numerical value as the minimum support value, mine and output all patterns whose support values are not less than the minimum support value.*

5.6 Experimental Evaluation

This section describes a set of experiments to demonstrate the effectiveness and efficiency of our proposed approach by comparing the performance of our proposed approach with that of the FP-growth algorithm. The performance of our proposed approach refers to the performance of the processes of approximately estimating the PSD and the CPSD of a target transaction dataset, determining a minimum support value *min_sup* by using our proposed fuzzy logic controller with two user-specified reference values ref_f and ref_s, and then mining frequent patterns by using FP-growth with *min_sup*. The performance of FP-growth refers to the performance of the process of using FP-growth directly with a determined *min_sup* to mine frequent patterns from the target dataset.

There are two real transaction datasets (i.e., the Retail and BMS-POS datasets introduced in Chapter 3) used in these experiments. All the experiments were performed on a 3.20GHz Intel Pentium(R) 4 PC machine with 1GB of available main memory, running the Windows XP professional operating system.

Tables 5.2 and 5.3 shows the distribution of patterns in the Retail and BMS-POS datasets against the value of *min_sup*. In Table 5.2 the value of *min_sup* linearly increases from 1 to $|TD|$ with a step of $[\frac{|TD|-1}{10}]$, while in Table 5.3 the value of *min_sup* exponentially increases from 1 to $|TD|$.

It can be clearly seen in Tables 5.2 and 5.3 that the number of patterns grows exponentially with the support value. However, there is no simple equation to accurately indicate this relationship. Thus, in general, without detailed knowledge of their target datasets, users have to use guesses and trials before a suitable minimum support value is found. Moreover, with noise and statistical fluctuation, the number of patterns grows with the increase of the support value, but does not precisely follow any known distribution. This phenomenon varies from one dataset to another, which makes it difficult to guess suitable minimum support values for applications. Obviously, that is inconvenient for users.

The next set of experiments tests the performance of our proposed fuzzy approach. To quickly estimate the power-law-based pattern support distribution, in

Retail			BMS-POS		
min_sup	FP-growth		*min_sup*	FP-growth	
	patterns	time(s)		patterns	time(s)
1		$\gg$ 3,600	1		$\gg$ 3,600
8,817	9	2.07	51,561	13	30.70
17,633	3	2.01	103,120	3	29.95
26,449	3	2.01	154,680	1	26.55
35,265	2	2.00	206,239	1	26.55
44,082	1	1.89	257,799	1	26.55
52,898	0	1.16	309,359	0	10.35
61,714	0	1.16	360,918	0	10.35
70,530	0	1.16	412,478	0	10.35
79,346	0	1.16	464,037	0	10.35
88,162	0	1.16	515,597	0	10.35

Table 5.2: Some experimental results with respect to the Retail and BMS-POS datasets to show the inconvenience when specifying minimum support values without knowing enough about the target datasets. Note that the value of *min_sup* linearly increases in these experiments.

Retail			BMS-POS		
min_sup	FP-growth		*min_sup*	FP-growth	
	patterns	time(s)		patterns	time(s)
1		$\gg$ 3,600	1		$\gg$ 3,600
3	20,647,331	496.72	4		$\gg$ 3,600
10	189,400	7.16	14		$\gg$ 3,600
30	33,130	3.79	52	31,669,064	627.85
95	6,949	2.83	193	1,147,814	65.84
297	1,150	2.34	718	59,706	38.96
927	149	2.17	2,675	3,908	33.93
2,895	25	2.10	9,968	328	31.72
9,041	8	2.06	37,141	28	31.62
28,233	3	2.01	138,382	1	26.55
88,162	0	1.16	515,597	0	10.35

Table 5.3: Other experimental results with respect to the Retail and BMS-POS datasets to show the inconvenience when specifying minimum support values without knowing enough about the target datasets. Note that the value of *min_sup* exponentially increases in these experiments.

all the experiments only 100 transactions as a sample dataset were randomly drawn from a target dataset.

Table 5.4 lists the approximately-estimated values of the scaling exponents and

the maximum support values covered by $PSD(Retail)$ and $CPSD(Retail)$.

Retail	α	k
$PSD(Retail)$	2.63	$1,115$
$CPSD(Retail)$	1.63	$19,836$

Table 5.4: The approximately-estimated values of the parameters of the best-fit power-law-based pattern support distribution and cumulative pattern support distribution of the Retail dataset from its sample datasets by using the methods in Section 5.5.2

The corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_f in the Retail dataset are shown in Figure 5.3, the corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_s are shown in Figure 5.8, and the corresponding fuzzy membership functions of the fuzzy sets defined in the range of min_sup are shown in Figure 5.9.

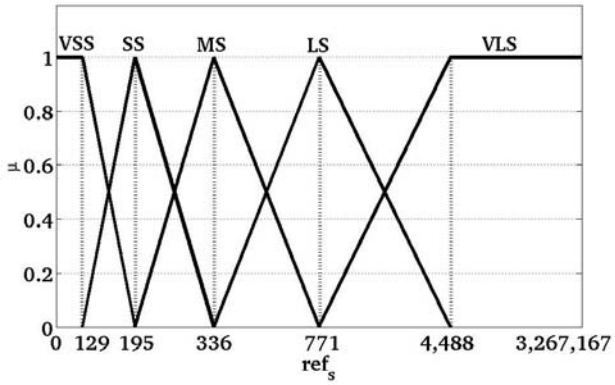


Figure 5.8: The corresponding membership functions of the fuzzy sets defined in the range of ref_s in the Retail dataset

Table 5.5 lists the approximate values of the scaling exponents and the maximum support values covered by $PSD(BMS\text{-}POS)$ and $CPSD(BMS\text{-}POS)$.

BMS-POS	α	k
$PSD(BMS\text{-}POS)$	3.06	$14,742$
$CPSD(BMS\text{-}POS)$	2.06	$157,257$

Table 5.5: The approximately-estimated values of the parameters of the best-fit power-law-based pattern support distribution and cumulative pattern support distribution of the BMS-POS dataset from its sample datasets by using the methods in Section 5.5.2

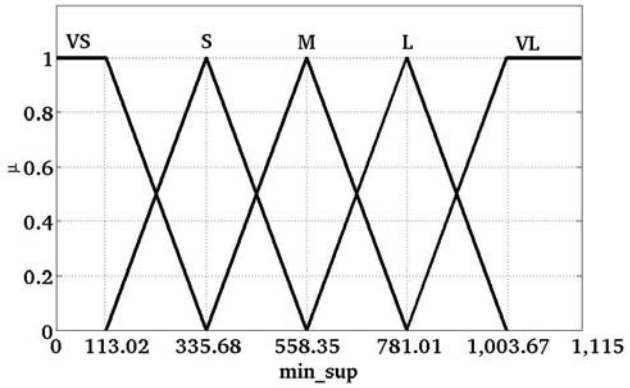


Figure 5.9: The corresponding membership functions of the fuzzy sets defined in the range of *min_sup* in the Retail dataset

The corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_f in the BMS-POS dataset are shown in Figure 5.3, the corresponding fuzzy membership functions of the fuzzy sets defined in the range of ref_s are shown in Figure 5.10, and the corresponding fuzzy membership functions of the fuzzy sets defined in the range of *min_sup* are shown in Figure 5.11.

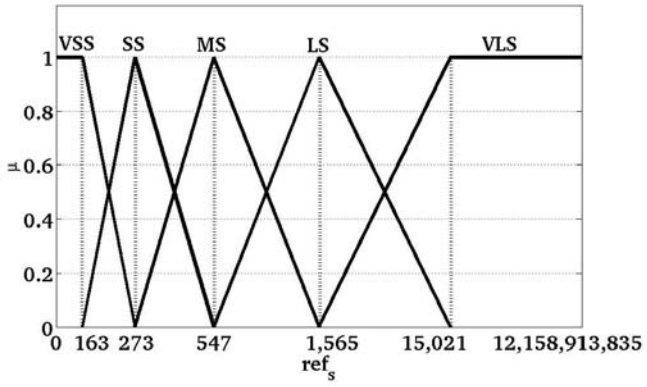


Figure 5.10: The corresponding membership functions of the fuzzy sets defined in the range of ref_s in the BMS-POS dataset

Tables 5.6 and 5.7 show the experimental results with respect to the Retail and BMS-POS datasets respectively. In each table, the first and second columns indicate two user-specified reference values ref_f and ref_s respectively. The third column indicates the execution time (in seconds) of approximately estimating the PSD and the $CPSD$ of a target dataset. The fourth column indicates the minimum support values determined by our proposed fuzzy logic controller according to a pair of ref_f and ref_s and the fifth column indicates the corresponding execution

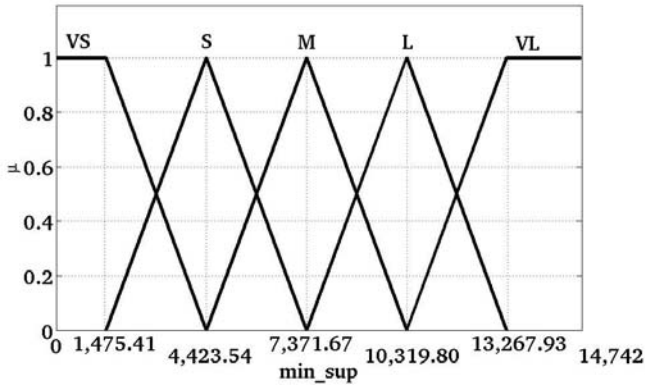


Figure 5.11: The corresponding membership functions of the fuzzy sets defined in the range of *min_sup* in the BMS-POS dataset

time (in seconds). The last two columns respectively present the number of mined patterns and the corresponding execution time (in seconds) by using FP-growth with a determined *min_sup*.

ref_f	ref_s	Our proposed approach with Retail				
		Estimating the PSD and the CPSD	Determining a *min_sup*		Mining FPs (FP-growth)	
		time(s)	*min_sup*	time(s)	patterns	time(s)
0.9	600	52.33	782	0.04	199	2.17
0.7	600	52.33	646	0.04	292	2.25
0.5	600	52.33	559	0.04	378	2.27
0.3	600	52.33	423	0.04	625	2.30
0.1	600	52.33	336	0.04	915	2.32
0.25	129	52.33	726	0.04	231	2.19
0.25	195	52.33	559	0.04	378	2.27
0.25	336	52.33	503	0.04	465	2.27
0.25	771	52.33	336	0.04	915	2.32
0.25	4,488	52.33	281	0.04	1,265	2.35
0.6	250	52.33	684	0.04	260	2.19
0.2	1,482	52.33	275	0.04	1,312	2.35
0.5	336	52.33	559	0.04	378	2.27
0.1	4,488	52.33	114	0.04	5,282	2.65

Table 5.6: The experimental results with respect to the Retail dataset

Each pair of ref_f in Column 1 and ref_s in Column 2, which are randomly selected from the ranges of ref_f and ref_s, has its corresponding value of *min_sup* determined by our proposed fuzzy logic controller.

ref_f	ref_s	Our proposed approach with BMS-POS				
		Estimating the PSD and the CPSD	Determining a *min_sup*		Mining FPs (FP-growth)	
		time(s)	*min_sup*	time(s)	patterns	time(s)
0.9	10,000	617.22	8,463	0.33	450	32.55
0.7	10,000	617.22	7,372	0.33	559	32.73
0.5	10,000	617.22	5,515	0.33	953	33.02
0.3	10,000	617.22	4,424	0.33	1,460	33.45
0.1	10,000	617.22	2,567	0.33	4,266	34.11
0.25	163	617.22	9,583	0.33	352	32.45
0.25	273	617.22	7,372	0.32	559	32.73
0.25	547	617.22	6,635	0.33	679	32.80
0.25	1,565	617.22	4,424	0.33	1,460	33.45
0.25	15,021	617.22	3,687	0.33	2,089	33.77
0.6	376	617.22	9,047	0.33	404	32.49
0.2	3,607	617.22	3,744	0.33	2,024	33.75
0.3	1,565	617.22	4,424	0.33	1,460	33.45
0.1	15,021	617.22	1,476	0.33	13,143	35.33

Table 5.7: The experimental results with respect to the BMS-POS dataset

Rows 1–5 in Tables 5.6 and 5.7 show the experimental results when fixing the value of ref_s and varying the value of ref_f. Rows 6–10 show the experimental results when varying the value of ref_s and fixing the value of ref_f. As ref_f changes linearly or ref_s changes exponentially, the determined value of *min_sup* is expected to change linearly and the number of mined patterns is expected to change exponentially. These experimental results clearly show that the changes of the determined value of *min_sup* and the number of mined patterns roughly follow our expectation with varying values of ref_s or ref_f. Thus, these experimental results have demonstrated that our proposed approach of determining *min_sup* based on ref_f and ref_s works effectively.

As shown in the experimental results in Rows 1–10 in Tables 5.6 and 5.7, there is some instability in the changes of the determined value of *min_sup* and the number of mined patterns when the value of ref_s or ref_f varies. The instability in the determined value of *min_sup* is mainly caused by the small number of fuzzy sets defined in the fuzzy logic controller used in these experiments, while the instability in the determined value of *min_sup* can cause further instability in the number of mined patterns. Note that there are only five fuzzy sets defined in each value range of the inputs and the output in these experiments. With the increase of the

number of the fuzzy sets defined in each value range, the amount of instability in the determined value of *min_sup* and the number of mined patterns can be reduced.

Rows 11–14 in Tables 5.6 and 5.7 show the experimental results when the expected minimum support value referred by ref_f is the exactly same to the one referred by the corresponding ref_s, according to the estimated PSD and CPSD of the Retail/BMS-POS dataset. Rows 11–12 give the experimental results when the expected minimum support value is not a vertex of two fuzzy membership functions defined in the value range of *min_sup*, while Rows 13–14 give the experimental results when the expected minimum support value is a vertex of two fuzzy membership functions defined in the value range of *min_sup*.

By comparing the expected number of patterns in Column 2 between Row 11 and Row 14 with the real number of mined patterns in Column 6 between Row 11 and Row 14, there is a difference between the expected number of patterns and the real number of patterns mined by using our proposed algorithm. The difference increases with the decrease of ref_f and/or the increase of ref_s.

Note that the number of patterns mined from a target dataset is decided by the value of *min_sup*. When projecting the difference between the expected and real number of outputted patterns on the level of *min_sup*, the difference can be much smaller. For example, in Table 5.7, when $ref_f = 0.1$ and $ref_s = 15,021$, the determined value of *min_sup* is $1,476$. The difference between the expected and real number of outputted patterns is $15,021 - 13,143 = 1,878$. By mining the BMS-POS dataset, one can find that the expected value of *min_sup* is $1,386$. And the difference between the expected and determined value of *min_sup* is $90 = 1,476 - 1,386$, which is much smaller than the difference between the expected and real number of outputted patterns.

The difference may be caused by three things. The first is the existence of noise in real transaction datasets, where the pattern support distribution does not perfectly follow a power-law distribution. The second is the uncertainty of random sampling. To reduce the computational cost of estimating the PSD and the CPSD of a target dataset, our proposed approach estimates the two distributions from the random sample datasets of the target dataset, which affects the accuracy of the estimated results. The last is the defined fuzzy membership functions in the value ranges of ref_f, ref_s and *min_sup*. Our proposed approach just adopts the simplest shapes to define fuzzy membership functions, since the study in this chapter is to demonstrate that the fuzzy method of solving the minimum support problem works.

Our proposed approach is efficient, although the total execution time of our proposed approach is much larger than that of a mining process that uses FP-growth directly with a determined minimum support value. The most expensively computational process in our proposed approach is the one of approximately estimating the PSD and the CPSD of a target dataset. When the PSD of a target dataset is known, one can avoid the estimation process. The total execution time of our proposed approach can be reduced to be very close to that of using FP-growth directly with a determined minimum support value, as shown in the experimental results in Tables 5.6 and 5.7.

While the estimation process is relatively expensive, it is only required once for a dataset. In reality, a dataset is rarely mined only once, since there can be many different user requirements and applications with the dataset. When the PSD and the CPSD of a target dataset are unknown, after the PSD and the CPSD of the dataset are estimated by running our proposed approach once, they can be saved for later.

When considering the uncertainty of random sampling, to obtain more reliable estimated results of the PSD and the CPSD of a dataset, one can compute the mean results by running our proposed approach with the dataset a number of times. After a number of times of running our proposed approach, the estimated results of the PSD and the CPSD of the dataset become stable, and then the expensive estimating process can be avoided.

Hence, in general, the total computational cost of mining a dataset by using our proposed approach can be close to that of using FP-growth directly with a determined minimum support value.

In conclusion, the experimental results and corresponding discussions in this section have demonstrated that our proposed approach is effective and efficient.

5.7 Summary

In this chapter, the problem of how to help users specify the minimum support value is studied. Normally, mining frequent patterns in a target dataset requires a suitable, data-independent and precise minimum support value to be specified by users. To find this minimum support value, users need detailed knowledge of the features of their target dataset. However, this knowledge is rarely known to users before they actually process their target dataset.

To deal with the problem, a new fuzzy approach was developed. Based on the power-law-based pattern support distribution of a target dataset and fuzzy logic control techniques, the proposed fuzzy approach can automatically convert user-specified data-independent reference values to a suitable, data-dependent and precise minimum support value that can be used to mine frequent patterns in the target dataset.

There are two user-specified reference values required by our proposed approach. One indicates the frequency of expected patterns, while the other indicates the number of expected patterns. In this way, users do not need to struggle to specify a suitable and data-dependent value for the minimum support. Besides, users are able to explicitly control the number of expected patterns.

We experimentally studied the performance of the proposed fuzzy approach with different transaction datasets and various users' expectation. The experimental results demonstrate the effectiveness and efficiency of the proposed fuzzy approach.

The aim of this study was to present a fuzzy approach to solving the minimum support problem and demonstrate that the proposed fuzzy approach works even with the simplest-shaped fuzzy membership functions. Thus, the shapes used for the membership functions in the proposed approach are triangular and trapezoidal. Also, only five fuzzy membership functions are defined in each value range of the inputs and output of the proposed fuzzy logic controller. Note that the mined results can be closer to the users' expectation with more complex-shaped fuzzy membership functions and/or more fuzzy membership functions. Naturally, this increases the execution cost at the same time.

To improve the performance of our proposed approach is in our future plan. We also plan to bring more constraints into our proposed fuzzy approach so that users can have more power to control mined results.

Chapter 6

Summary and Conclusions

The end is the crown of any work.

— Russian Proverb

This thesis has investigated the issues relating to frequent pattern mining in transaction datasets and power-law-based pattern support distributions of real static transaction datasets. Several new methods to overcome the problems identified have been proposed, developed, evaluated, and compared with existing methods.

This chapter summarizes the work detailed in each chapter and suggests some directions that have opened up for further exploration.

6.1 An Overview of the Research

As introduced in Chapter 1, with the development of Information Technology, an enormous quantity of data is accumulated every day. A great challenge is to find useful knowledge from this huge amount of data. Data mining appears as an effective research direction to meet this challenge. There are many types of useful knowledge hidden in various data, of which common patterns are one. Each pattern in a dataset has a support value, which is the frequency of its appearance in the dataset, and the distribution of patterns against their corresponding supports is the pattern support distribution of the dataset.

In this thesis, we mainly focus on the issues of mining frequent patterns (i.e., those with high support values) and exploring power-law relationships in pattern support distributions of transaction datasets. Frequent pattern mining plays a core role in many data mining tasks, since frequent patterns can be easily applied in many real world applications for better decision support. Mining frequent patterns

has been studied extensively in the field of data mining, but most previous studies concentrate on conventional frequent pattern mining, which was initially developed for mining patterns in static datasets, where data is not expected to vary with time. However, datasets can be updated on a regular basis, and so frequent patterns in these datasets evolve.

The study in Chapter 2 explored how to use the incremental method to extend existing mining algorithms to efficiently handle dynamic datasets. With this aim, we investigated the problem of effectively and efficiently mining frequent closed patterns in a dynamic transaction dataset, with a user-specified minimum frequency value. Frequent closed patterns are a condensed representation of frequent patterns. We propose a modification of an existing method called CLOSET+, which is one of the most efficient algorithms for mining frequent closed patterns in static transaction datasets, to efficiently mine frequent closed patterns in dynamic transaction datasets.

It is also very valuable to learn the features of the target dataset in order to achieve better algorithmic efficiency and better understanding of the mined results. In Chapter 3, power-law relationships and self-similarity phenomena in the pattern support distributions of real transaction datasets were examined with a large number of experimental studies and/or theoretical analyses. We found that the existence of power-law relationships in the pattern support distributions of real retail transaction datasets cannot be ruled out at the level of basic distributions.

Accurately identifying the power-law-based pattern support distribution of a dataset is not trivial, since it requires all patterns and their corresponding support values in the target dataset. Sampling can be a potential solution to this problem. Thus, in Chapter 4, we investigated the influence of sampling on estimating the power-law-based pattern support distribution of a static transaction dataset and proposed a cost-effective approach to estimating the power-law-based pattern support distribution of a real static transaction.

Most conventional mining algorithms for frequent patterns need a user-specified precise data-dependent value called the minimum support as an input parameter to distinguish frequent and infrequent patterns in a target dataset, because the assumption behind these algorithms is that patterns that appear frequently (i.e., have high support values) are the interesting ones. To specify a suitable minimum support value, users need to have detailed knowledge of the target dataset. In most cases, users are unlikely to have such knowledge before they actually specify the

minimum support value and mine the target dataset. To conquer this problem, in Chapter 5, based on the power-law-based pattern support distribution and fuzzy logic control techniques, a fuzzy approach is presented to automatically determine a suitable value for the minimum support in order to mine frequent patterns in a target dataset.

6.2 Mapping Achievements to Aims and Objectives

The previous section briefly summarized the research undertaken in each chapter of this thesis. This section aims to verify that our research aims and objectives, set in Chapter 1, have been accomplished.

The main goal of this research is to study and improve frequent pattern mining in transaction datasets. To achieve the main goal, we set two aims: designing effective and efficient algorithms, and learning the features of datasets. These two aims are further divided into three objectives.

1. *Investigating how to extend conventional mining approaches of frequent pattern to be used in dynamic transaction datasets.*

   A modification of the CLOSET+ algorithm is proposed to incrementally mine frequent closed patterns in dynamic transaction datasets. Based on the theoretical analyses on how the support of a pattern in a dynamic transaction dataset changes with an update (i.e., adding new transactions, deleting existing transactions, or modifying existing transactions), all frequent patterns in a target dynamic transaction dataset can be successfully discovered with a data-dependent minimum support value specified by users after each update.

   Several sets of experiments were conducted to measure the performance of the proposed algorithm with the number of transactions in an update and the number of transactions in the base dataset. The experimental results show that the proposed algorithm is more efficient than rerunning CLOSET+ in the updated dataset, when the number of transactions changed by the dataset update is relatively small by comparing with that of unchanged transactions in the dataset.

2. *Exploring the power-law-based pattern support distributions of real static transaction datasets.*

The pattern support distribution has been studied extensively in this thesis. By using qualitative appraisals, statistical goodness-of-fit tests, the bootstrap method, and the universality of power-law relationships, the power-law relationship has been discovered in the pattern support distributions of real static transaction datasets.

To accurately identify the power-law-based pattern support distribution in a dataset is generally extremely computationally expensive. To estimate the power-law-based pattern support distribution inexpensively, it seems reasonable to use sampling methods. Consequently, we identify and analyze the influence of sampling on estimating power-law-based pattern support distributions.

Eventually, we propose two estimation methods (an approximate estimation and a numerical estimation) of the power-law-based pattern support distribution in a transaction dataset. The theoretical discussion and the demonstrating example show that the proposed methods can inexpensively estimate the power-law-based pattern support distribution in a target transaction dataset with relatively high accuracy.

3. *Investigating how to enhance conventional mining approaches of frequent pattern to be used without a precise and data-dependent minimum support value.*

   A novel fuzzy approach is proposed to deal with the issue of the user-specified minimum support value. The proposed approach only requires users to specify two approximate and data-independent reference values instead of a precise and data-dependent minimum support value needed by most conventional mining algorithms of frequent pattern. Thus, the proposed approach does not need users to have detailed knowledge of their target datasets. Moreover, one of the user-specified reference values indicates the degree of the number of expected patterns. Users are able to explicitly control the number of expected patterns.

   The performance of the proposed fuzzy approach has been examined with different static transaction datasets and various user expectations. The experimental results demonstrate the effectiveness and efficiency of the proposed fuzzy approach, which only uses a few simple-shaped fuzzy membership functions.

6.3 Open Questions and Future Work

As shown in Section 6.1, this thesis has made contributions to the research area of frequent pattern mining in transaction datasets. However, the insights gained have also opened up some areas of future research. This section describes some of these areas.

- Approximate frequent pattern mining is worthy of study. The study presented in this thesis has focused on precise support values. However, in some situations, where multiple scans on a whole target dataset is not possible, such as a data stream, or users emphasize the execution time of a mining process, mining patterns with precise support values may not be applicable. Approximating the support values of patterns in a target dataset instead of precisely counting the support values of patterns can avoid multiple scans on the whole target dataset and reduce the execution time of the mining process. Although the mined results of approximate frequent pattern mining are not very precise, approximate frequent pattern mining can still be a very promising solution for such situations where precise results are not strictly required.

- The user-specified minimum frequency value may change during the process of mining frequent patterns in a dynamic transaction dataset. For instance, users may want to mine the top $1,000$ patterns with highest support values after each update. In a dynamic dataset, each update can change the features of the dataset. Thus, the minimum frequency value meeting users' expectations may need to be changed after each update. Obviously, it is not practical that users manually change the minimum frequency value after each update. A combination of our proposed approaches in Chapter 2 and 5 can be a potential solution. However, it is still challenging to develop an efficient algorithm.

- Theoretical analyses of the issues regarding the power-law-based pattern support distributions of transaction datasets needs to be further studied. For instance, as shown in Chapter 4, our theoretical analyses on the influence of sampling on power-law-based pattern support distributions meets the hypergeometric function with general arguments, which does not currently have a theoretical solution. Therefore, advanced mathematical development and/or different approaches on such issues are required. The outcomes of such theoretical analyses will not only provide the theoretical insight into power-law-based

pattern support distributions of transaction datasets, but also help in dealing with similar issues in other research fields.

- It could be interesting to examine whether the power-law relationship also exists in other pattern-related distributions of real datasets, such as frequent closed pattern distributions and frequent sequential patterns. Although we have investigated the power-law relationship existing in pattern support distributions of real datasets, there are still many other pattern-related distributions. The benefits of identifying the power-law relationship existing in pattern support distributions can clearly be seen in this thesis. Thus, it is very valuable to examine whether the power-law relationship also exists in other pattern-related distributions. That could reveal more detailed knowledge of target datasets, bring more ideas to create more effective and efficient algorithms and help users to understand their mined results.

- Research into frequent pattern mining continues in various types of data, not only transaction data. There are lots of other types of data including unstructured text, video and audio, which play an increasingly important role in our daily lives. Extending and developing mining techniques to extract useful knowledge from all types of data is a promising research direction.

Bibliography

[1] Cater general presentation. `http://www.cater-ist.org/docs/CATER_General_Presentation.pdf`, March 2009.

[2] Martha L. Abell, James P. Braselton, and John Arthur Rafter. *Statistics with Mathematica*, volume 1. Academic Press, 1999.

[3] Lada Adamic and Bernardo A. Huberman. The nature of markets in the world wide web. *Quarterly Journal of Electronic Commerce*, 1:5–12, 2000.

[4] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining associations between sets of items in massive databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington D.C., USA, May 1993.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *Proceedings of the Twentieth International Conference on Very Large DataBases (VLDB)*, pages 487–499, Santiago, Chile, September 12–15 1994.

[6] R. Askey. Ramanujan and hypergeometric and basic hypergeometric series. *Russian Mathematical Surveys*, 45(1), 1990.

[7] F. Auerbach. Das gesetz der bevölkerungskonzentration. *Petermanns Geographische Mitteilungen*, 59:74–76, 1913.

[8] Yonatan Aumann, Ronen Feldman, Orly Lipshtat, and Heikki Manilla. Borders: An efficient algorithm for association generation in dynamic databases. *Journal of Intelligent Information Systems*, 12(1):61–73, April 1999.

[9] N. F. Ayan, A. U. Tansel, and E. Arkun. An efficient algorithm to update large itemsets with early pruning. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[10] Jay Ayres, Johannes Gehrke, Tomi Yiu, and Jason Flannick. Sequential pattern mining using bitmaps. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.

[11] Sikha Bagui. An approach to mining crime patterns. *International Journal of Data Warehousing and Mining*, 2(1):50–80, 2006.

[12] Wilfrid Norman Bailey. *Generalized Hypergeometric Series*. Camrbridge University Press, the first edition, 1935.

[13] Roberto J. Bayardo. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 85–93, Seattle, Washington, USA, 1998. ACM.

[14] Rezaul Begg and Marimuthu Palaniswami. *Computational Intelligence for Movement Sciences: Neural Networks and Other Emerging Techniques*. Computational Intelligence and its Applications (COMIA) Book Series. IGI Publishing, March 2006.

[15] Jérémy Besson, Christophe Rigotti, Ieva Mitašiunaite, and Jean-François Boulicaut. Parameter tuning for differential mining of string patterns. In *Proceedings of the 2008 IEEE International Conference on Data Mining Workshops (ICDMW'08)*, pages 77–86, Pisa, Italy, December 2008. IEEE Computer Society.

[16] Frits Beukers. Gauss' hypergeometric function. Technical report, Utrecht University, April 12 2005.

[17] Christian Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the First International Workshop on Open Source Data Mining (OSDM'05)*, pages 1–5, Chicago, Illinois, USA, 2005. ACM Press.

[18] Tom Brijs, Gilbert Swinnen, Koen Vanhoof, and Geert Wets. Using association rules for product assortment decisions: A case study. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 254–260, San Diego, California, USA, 1999. ACM.

[19] S. Brin, R. Motwani, and C. Silverstein. Beyond market basket: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM-SIGMOD*

International Conference Management of Data (SIGMOD97), pages 265–276, Tucson, Arizona, USA, May 1997.

[20] Douglas Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the Seventeenth International Conference on Data Engineering*, pages 443–452, Washington, D.C., USA, 2001. IEEE Computer Society.

[21] C. Chang, Y. Li, and J. Lee. An efficient algorithm for incremental mining of association rules. In *Proceedings of the Fifteenth International Workshop on Research Issues on Data Engineering: Stream Data Mining and Applications (RIDE-SDMA 2005)*, pages 3–10, 2005.

[22] Yen-Liang Chen and Cheng-Hsiung Weng. Mining fuzzy association rules from questionnaire data. *Knowledge-Based Systems*, 22(1):46–56, January 2009.

[23] D. W. Cheung, S. D. Lee, and Kao B. A general incremental technique for maintaining discovered association rules. In *Proceedings of the Fifth International Conference On Database Systems For Advanced Applications*, pages 185–194, Melbourne, Australia, March 1997.

[24] David Wai-Lok Cheung, Jiawei Han, Vincent Ng, and C. Y. Wong. Maintenance of discovered association rules in large databases: An incremental updating technique. In *Proceedings of the Twelfth International Conference on Data Engineering (ICDE'96)*, pages 106–114, New Orleans, Louisiana, USA, February 1996. IEEE Computer Society.

[25] William Cheung and Osmar R. Zaïane. Incremental mining of frequent patterns without candidate generation or support constraint. In *Proceedings of the Seventh International Database Engineering and Applications Symposium*, pages 111–116, July 2003.

[26] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, 2004.

[27] P. B. Chou, E. Grossman, D. Gunopulos, and P. Kamesam. Identifying prospective customers. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456. ACM Press, 2000.

[28] Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. On exploring the power-law relationship in the itemset support distribution. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *Advances in Database Technology—EDBT 2006*, volume 3896 of *Lecture Notes in Computer Science*, pages 682–699, Munich, Germany, March 26–31 2006. Springer Berlin / Heidelberg.

[29] Kun-Ta Chuang, Jiun-Long Huang, and Ming-Syan Chen. Power-law relationship and self-similarity in the itemset support distribution: analysis and applications. *The International Journal on Very Large Data Bases*, 17(5):1121–1141, August 2008.

[30] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, November 2009.

[31] John Derbyshire. *Prime Obsession: Bernhard Riemann and the Greatest Unsolved Problem in Mathematics*. John Henry Press, Washington, DC, 2003.

[32] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 1999 International Conference Knowledge Discovery and Data Mining (KDD99)*, pages 43–52, San Diego, California, USA, August 1999.

[33] Jacques Dutka. The early history of the hypergeometric function. *Archive for History of Exact Sciences*, 31(1):15–34, 1984.

[34] M. Raafat El-Gewely. *Biotechnology Annual Review*, volume 1 of *Biotechnology Annual Review Series*. Elsevier, 1995.

[35] B.S. Everitt and A. Skrondal. *The Cambridge Dictionary of Statistics*. Cambridge University Press, 2010.

[36] Christie I. Ezeife and Yue Su. Mining incremental association rules with generalized fp-tree. *Lecture Notes in Computer Science (LNCS)*, 2338:147–160, 2002. http://www.springerlink.com/content/ubdckuurthekwtm7/.

[37] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Proceedings of the Second International Conference on Knowledge Discovery and Data*

Mining (KDD-96), pages 82–88, Portland, Oregon, USA, August 1996. AAAI Press.

[38] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.

[39] R. Feldman, Y. Aumann, A. Amir, and H. Mannila. Efficient algorithms for discovering frequent sets in incremental databases. In *Proceedings of the 1997 SIGMOD Workshop on DMKD*, Tucson, Arizona, USA, May 1997.

[40] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. *Knowledge Discovery in Databases*, chapter Knowledge Discovery in Databases: An Overview, pages 1–30. AAAI/MIT Press, 1991.

[41] H. Fu and E. Mephu Nguifo. Mining frequent closed itemsets for large data. In *Proceedings of the 2004 International Conference on Machine Learning and Applications (ICMLA04)*, 2004.

[42] Karolien Geurts, Geert Wets, Tom Brijs, and Koen Vanhoof. Profiling high frequency accident locations by using association rules. In *Proceedings of the Eighty-second Annual Transportation Research Board*, number 1840, pages 123–130, Washington D.C., USA, January 12–16 2003. Transportation Research Board.

[43] M.L. Goldstein, S.A. Morris, and G.G. Yen. Problems with fitting to the power-law distribution. *The European Physical Journal B*, 41(2):255–258, September 2004.

[44] Gösta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Proceedings of the IEEE ICDM Workshop in Frequent Itemset Mining Implementations*, pages 123–132, Melbourne, Florida, USA, 2003.

[45] Michael Hahsler. A model-based frequency constraint for mining associations from transaction data. *Data Mining and Knowledge Discovery*, 13(2):137–166, September 01 2006.

[46] Jia-Wei Han, Jian Pei, and Xi-Feng Yan. From sequential pattern mining to structured pattern mining: a pattern-growth approach. *Journal of Computer Science and Technology*, 19(3):257–279, 2004.

[47] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pages 1–12, Dallas, Texas, USA, May 2000. ACM.

[48] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 211–218, Maebashi, Japan, 2002. IEEE Computer Society.

[49] Jiawei Han, Yiwen Yin, and Guozhu Dong. Efficient mining of partial periodic patterns in time series database. In *Proceedings of the Fifteenth International Conference on Data Engineering (ICDE'99)*, 1999.

[50] Robert J. Hilderman. Assessing the interestingness of discovered knowledge using a principled objective approach. In *Proceedings of the Workshop on Utility-Based Data Mining (UBDM'06), the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 38–47, Philadelphia, USA, August 2006.

[51] Yu Hirate, Eigo Iwahashi, and Hayato Yamana. TF^{2}P-growth: An efficient algorithm for mining frequent patterns without any thresholds. In *Proceedings of IEEE ICDM 2004 Workshop on Alternative Techniques for Data Mining and Knowledge Discovery*, Brighton, the UK, November 01 2004. http://elvex.ugr.es/icdm2004/pdf/hirate.pdf.

[52] Tzung-Pei Hong, Chan-Sheng Kuo, and Shyue-Liang Wang. A fuzzy aprioritid mining algorithm with reduced computational time. *Applied Soft Computing*, 5(1):1–10, December 2004.

[53] Tzung-Pei Hong and Yeong-Chyi Lee. *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, chapter An Overview of Mining Fuzzy Association Rules, pages 397–410.

[54] Kuo-Yu Huang and Chia-Hui Chang. Efficient mining of frequent episodes from complex sequences. *Information Systems*, 33(1):96–114, 2008.

[55] Adolf Hurwitz. Some properties of the Dirichlet function $F(x) = \Sigma \left(\frac{D}{n}\right) \frac{1}{n^s}$, which occurs in the determination of the class numbe of binary quadratic

forms. (Einige Eigenschaften der Dirichlet'schen Functionen $F(x) = \Sigma\left(\frac{D}{n}\right)\frac{1}{n^s}$, die bei der Bestimmung der Klassenzahlen binärer quadratischer Formen auftreten.). *Z. für Math. und Physik*, pages 86–102, 1882.

[56] Saidat Adebukola Ibrahim, Olusegun Folorunso, and Olutayo Bamisele Ajayi. Knowledge discovery of closed frequent calling patterns in a telecommunication database. In *Proceedings of the 2005 Informing Science and IT Education Joint Conference*, pages 137–148, Flagstaff, Arizona, USA, June 16–19 2005.

[57] Roberto J. Bayardo Jr. Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 85–93, New York, NY, USA, 1998. ACM.

[58] Peter Kennedy. *A guide to econometrics*. The MIT Press, Cambridge, Massachusetts, USA, the fifth edition, 2003.

[59] Peter Eris Kloeden, Eckhard Platen, and Henri Schurz. *Numerical Solution of SDE Through Computer Experiments*. Springer, the first edition, 1994.

[60] Ron Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207, Portland, Oregon, USA, August 2-4 1996.

[61] Ron Kohavi, Carla Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000. http://www.ecn.purdue.edu/KDDCUP.

[62] Walter A. Kosters and Wim Pijls. Apriori, a depth first implementation. In B. Goethals FL and M. J. Zaki, editors, *IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, Melbourne, Australia, 2003. IEEE Press.

[63] Hare Krishna and Pramendra Singh Pundir. Discrete Maxwell distribution. *InterStat*, (3), November 2007.

[64] Reiner Kühnau. *Handbook of complex analysis: geometric function theory*. Handbook of Complex Analysis. North Holland/Elsevier, 2004.

[65] Chang-Hung Lee, Cheng-Ru Lin, , and Ming-Syan Chen. Sliding-window filtering: An efficient algorithm for incremental mining. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, pages 263–270, New York, NY, USA, 2001. ACM.

[66] Chang-Hung Lee, Cheng-Ru Lin, and Ming-Syan Chen. Sliding window filtering: an efficient method for incremental mining on a time-variant database. *Information Systems*, 30(3):227–244, May 2005.

[67] Carson Leung, Quamrul Khan, Zhan Li, and Tariqul Hoque. Cantree: a canonical-order tree for incremental frequent-pattern mining. *Knowledge and Information Systems*, 11(3):287–311, 2007. 10.1007/s10115-006-0032-8.

[68] L. Lhote, F. Rioult, and A. Soulet. Average number of frequent (closed) patterns in Bernoulli and Markovian databases. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM05)*, pages 713–716, November 2005.

[69] Deyi Li and Yi Du. *Artificial Intelligence with Uncertainty*. Chapman & Hall, the first edition, September 27 2007.

[70] Xueming Li, Dongxia Qin, and Cun Yu. ACCF: Associative classification based on closed frequent itemsets. In *Proceedings of the fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'08)*, volume 2, pages 380–384, October 2008.

[71] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 337–341, San Diego, California, USA, 1999.

[72] Sharon L. Lohr. *Sampling: Design and Analysis*. Duxbury Press, 1999.

[73] Alfred J. Lotka. The frequency distribution of scientific productivity. *Journal of the Washington Academy of Sciences*, 16(12):317–324, 1926.

[74] Claudio Lucchese, Salvatore Orlando, and Raffaele Perego. DCI_Closed: A fast and memory efficient algorithm to mine frequent closed itemsets. In Roberto Bayardo, Bart Goethals, and Mohammed J. Zaki, editors, *Proceedings of the*

IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2004), Brighton, the UK, November 204.

[75] Y. Ma, B. Liu, C. K. Wong, P. S. Yu, and S. M. Lee. Targeting the right students using data mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 457–464. ACM Press, 2000.

[76] Neal Noah Madras. *Monte Carlo methods*, volume 26 of *Fields Institute Monographs*. AMS Bookstore, 2000.

[77] E. H. Mamdani and S. Assilian. A case study on the application of fuzzy set theory to automatic control. In *Proceedings of the IFAC Stochastic Control Symposium*, pages 643–649, Budapest, Hungary, September 1974.

[78] Benoit Mandelbrot. How Long Is the Coast of Britain? Statistical Self-Similarity and Fractional Dimension. *Science*, 156(3775):636–638, 1967.

[79] H. Mannila, H. Toivonen, and A.I. Verkamo. Discovery of frequent episodes in event sequences. In *Data Mining and Knowledge Discovery*, volume 1, pages 259–289, 1997.

[80] Paul Meakin. *Fractals, scaling and growth far from equilibrium*, volume 5 of *Cambridge Nonlinear Science Series*. Cambridge University Press, January 15 1998.

[81] Shu-Ching Chen Mei-Ling Shyu and R. L. Kashyap. Generalized affinity-based association rule mining for multimedia database queries. *Knowledge and Information Systems*, 3(3), August 2001.

[82] Quang Tran Minh, Shigeru Oyanagi, and Katsuhiro Yamazaki. *Intelligent Data Engineering and Automated Learning IDEAL 2006*, volume 4224/2006 of *Lecture Notes in Computer Science*, chapter Mining the K-Most Interesting Frequent Patterns Sequentially, pages 620–628. Springer Berlin / Heidelberg, 2006. http://www.springerlink.com/content/p7g83378824x4430.

[83] Sasuke Miyazima, Youngki Lee, Tomomasa Nagamine, and Hiroaki Miyajima. Power-law distribution of family names in Japanese societies. *Physica A: Statistical Mechanics and its Applications*, 278:282–288, April 2000.

[84] A. Mueller. Fast sequential and parallel algorithms for association rule mining: a comparison. Technical Report CS-TR-3515, University of Maryland, College Park, August 1995.

[85] R. M. W. Musson, T. Tsapanos, and C. T. Nakas. A power-law function for earthquake interarrival time and magnitude. *Bulletin of the Seismological Society of America*, 92(5):1783–1794, June 2002.

[86] G. Neukum and B. A. Ivanov. *Hazards Due to Comets and Asteroids*, chapter Crater Size Distributions and Impact Probabilities on Earth from Lunar, Terrestrial-planet, and Asteroid Cratering Data, pages 359–416. Space Science Series. University of Arizona Press, Tucson, Arizona, USA, 1994.

[87] Miroslav Michal Novak. *Emergent nature: patterns, growth and scaling in the sciences*. World Scientific, 2002.

[88] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, January 2003.

[89] T. Oyama, K. Kitano, K. Satou, and T. Ito. Mining association rules related to protein-protein interactions. *Genome Informatics*, pages 358–359, 2000.

[90] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. *ACM SIGMOD Record*, 24(2):175–186, May 1995.

[91] Srinivasan Parthasarathy. Efficient progressive sampling for association rules. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, pages 354–361, Washington, D.C., USA, 2002. IEEE Computer Society.

[92] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory*, 1999.

[93] J. Pei, J. Han, and R. Mao. CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000.

[94] Gregory Piatetsky-Shapiro and William J. Frawley. *Knowledge Discovery in Databases*. AAAI/MIT Press, Cambridge, Massachusetts, USA, 1991.

[95] Gregory Piatetsky-Shapiro and Sam Steingold. Measuring lift quality in database marketing. *ACM SIGKDD Explorations Newsletter*, 2(2):76–80, December 2000.

[96] J. H. Pollard. *A Handbook of Numerical and Statistical Techniques: with Examples Mainly from the Life Sciences*. CUP Archive, 1979.

[97] Karl R. Popper. *The logic of scientific discovery*. Routledge, London, the UK, 1959.

[98] S.-L. Qiu and M. Vuorinen. Landen inequalities for hypergeometric functions. *Nagoya Mathematical Journal*, 154:31–56, 1999.

[99] Ganesh Ramesh, William A. Maniatty, and Mohammed J. Zaki. Feasible itemset distributions in data mining: theory and application. In *Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'03)*, pages 284–295, San Diego, California, USA, 2003. ACM.

[100] Inder K. Rana. *An introduction to measure and integration*. Graduate Studies in Mathematics. American Mathematical Society, 2002.

[101] Bernhard Riemann. Ueber die anzahl der primzahlen unter einer gegebenen grösse. *Monatsberichte der Berliner Akademie*, November 1859.

[102] John F. Roddick and Sally Rice. What's interesting about cricket?: on thresholds and anticipation in discovered rules. *ACM SIGKDD Explorations Newsletter*, 3(1):1–5, July 2001.

[103] T. A. Runkler. Selection of appropriate defuzzification methods using application specific properties. *IEEE Transactions on Fuzzy Systems*, 5(1):72–79, February 1997.

[104] Sigal Sahar. On incorporating subjective interestingness into the mining process. In *Proceedings of the Second IEEE International Conference on Data Mining (ICDM'02)*, pages 681–684, Los Alamitos, California, USA, December 09–12 2002. IEEE Computer Society. http://doi.ieeecomputersociety.org/10.1109/ICDM.2002.1184028.

[105] N. L. Sarda and N. V. Srinivas. An adaptive algorithm for incremental mining of association rules. In *Proceedings of the ninth International Workshop on Database and Expert Systems Applications*, pages 240–245. IEEE Computer Society, 1998.

[106] Ashoka Savasere, Edward Omiecinski, and Shamkant Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the Twenty-first International Conference on Very Large Data Bases*, pages 432–444, San Francisco, California, USA, September 1995. Morgan Kaufmann Publishers Incorporated.

[107] Masakazu Seno and George Karypis. Lpminer: An algorithm for finding frequent itemsets using length decreasing support constraint. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 505–512, San Jose, California, USA, 2001.

[108] Ian S. Shaw. *Fuzzy control of industrial systems: theory and applications*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1998.

[109] Craig Silverstein, Sergey Brin, and Rajeev Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, January 1998.

[110] Lucy Joan Slater. *Generalized Hypergeometric Functions*. Cambridge University Press, 2 January 1966.

[111] Didier Sornette. *Encyclopedia of Complexity and Systems Science*, chapter Probability Distributions in Complex Systems, pages 7009–7024. Springer New York, 2009. `http://dx.doi.org/10.1007/978-0-387-30440-3_418`.

[112] Gerd Stumme, Rafik Taouil, Yves Bastide, Nicolas Pasquier, and Lotfi Lakhal. Computing iceberg concept lattices with titanic. *Data & Knowledge Engineering*, 42(2):189–222, August 2002.

[113] Yudho Giri Sucahyo, Raj P. Gopalan, and Amit Rudra. *AI 2003: Advances in Artificial Intelligence*, chapter Efficiently Mining Frequent Patterns from Dense Datasets Using a Cluster of Computers, pages 233–244. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.

[114] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Cp-tree: a tree structure for single-pass frequent pattern mining. In Takashi Washio, Einoshin Suzuki, Kai Ting, and Akihiro Inokuchi, editors, *Proceedings of the Twelfth Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD'08)*, volume 5012 of *Lecture Notes in Computer Science*, pages 1022–1027, Osaka, Japan, 2008. Springer Berlin / Heidelberg. `http://dx.doi.org/10.1007/978-3-540-68125-0_108`.

[115] Nikolaj Tatti. Maximum entropy based significance of itemsets. *Knowledge and Information Systems*, 17(1):57–77, October 2008.

[116] Shiby Thomas, Sreenath Bodagala, Khaled Alsabti, and Sanjay Ranka. An efficient algorithm for the incremental updation of association rules in large datbases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 263–266, New Port Beach, California, USA, August 1997.

[117] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the Twenty-second International Conference on Very Large Data Bases (VLDB'96)*, pages 134–145, Mumbay, India, September 1996.

[118] Kuei-Ying Lin Tzung-Pei Hong and Been-Chian Chien. Mining fuzzy multiple-level association rules from quantitative data. *Applied Intelligence*, 18, January 2003.

[119] Quang H. Vuong. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57(2):307–333, March 1989.

[120] Jianyong Wang, Jiawei Han, and Jian Pei. Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 236–245, Washington, D.C., USA, August 2003. ACM.

[121] Lusheng Wang, Hao Zhao, Guozhu Dong, and Jianping Li. On the complexity of finding emerging patterns. *Theoretical Computer Science*, 335(1):15–27, 2005.

[122] T. Wetjen. Discovery of frequent gene patterns in microbial genomes. TZI-Report, Technologie Zentrum Informatik (TZI) 27, University of Bremen, Germany, 2002.

[123] Dietmar Wolfram. *Applied Informetrics for Information Retrieval Research*. New Directions in Information Management. Greenwood Publishing Group, July 2003.

[124] Guizhen Yang. The complexity of mining maiximal frequent itemsets and maximal frequent patterns. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 344–353, Seattle, Washington, USA, August 2004.

[125] Jiong Yang, Wei Wang, and Philip S. Yu. Infominer+: mining partial periodic patterns with gap penalties. In *Proceedings of the Second IEEE International Conference on Data Mining (ICDM02)*, pages 725–728. IEEE Press, 2002.

[126] Man Lung Yiu and Nikos Mamoulis. Frequent-pattern based iterative projected clustering. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, pages 689–692, Washington, D.C., USA, November 2003. IEEE Computer Society.

[127] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[128] Lotfi A. Zadeh. A rationale for fuzzy control. *Journal of Dynamic Systems, Measurements and Control*, 34:3–4, 1972.

[129] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(1):28–44, January 1973.

[130] M. Zaki and C. Hsiao. Charm: an efficient algorithm for closed association rule mining. Technical report 99–10, Computer Science, Rensselaer Polytechnic Institute, 1999.

[131] Mohammed J. Zaki. Generating non-redundant association rules. In *Proceedings of Sixth International Conference of Knowledge Discovery and Data Mining*, pages 34–43, New York, NY, USA, 2000. ACM.

[132] Mohammed J. Zaki and Karam Gouda. Fast vertical mining using diffsets. Technical report 01-1, Computer Science Department, Rensselaer Polytechnic Institute, March 2001.

[133] Mohammed J. Zaki and Ching-Jui Hsiao. CHARM: An efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining*, Arlington, VA, USA, April 2002.

[134] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Wei Li, and Mitsunori Ogihara. Evaluation of sampling for data mining of association rules. In *Proceedings of the Seventh International Workshop on Research Issues in Data Engineering*, pages 42–50, April 1997.

[135] Chengqi Zhang and Shichao Zhang. *Association rule mining: models and algorithms*, volume 2307 of *Lecture Notes in Artificial Intelligence*. Springer Berlin / Heidelberg, New York, NY, USA, 2002.

[136] Shichao Zhang, Jingli Lu, and Chengqi Zhang. A fuzzy logic based method to acquire user threshold of minimum-support for mining association rules. *Information Sciences*, 164(1–4):1–16, September 2004.

[137] Shichao Zhang, Xindong Wu, Chengqi Zhang, and Jingli Lu. Computing the minimum-support for mining frequent patterns. *Knowledge and Information Systems*, 15(2):233–257, May 2008.

[138] Zhongnan Zhang, Weili Wu, and Yaochun Huang. Mining dynamic interdimension association rules for local-scale weather prediction. In *Proceedings of the Twenty-eighth Annual International Computer Software and Applications Conference (COMPSAC '04) - Workshops and Fast Abstracts*, pages 146–149, Washington, D.C., USA, 2004. IEEE Computer Society.

[139] Zhongmei Zhou, Zhaohui Wu, Chunshan Wang, and Yi Feng. *Fuzzy Systems and Knowledge Discovery*, volume 4223, chapter Efficiently Mining Both Association and Correlation Rules, pages 369–372. Springer Berlin / Heidelberg, 2006.

[140] Lixing Zhu and Lixing Zhu. *Nonparametric Monte Carlo tests and their applications*, volume 182 of *Lecture Notes in Statistics*. Springer, 2005.

[141] George K. Zipf. *The psycho-biology of language: an introduction to dynamic philology.* Houghton Mifflin company, Boston, 1935.

[142] George K. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, Cambridge, 1949.